STABILITY ANALYSIS OF RECURRENT NEURAL-BASED CONTROLLERS

By
REZA JAFARI

Master of Science in Mechatronics
American University of Sharjah
Sharjah
2005

Master of Science in Mathematics
Oklahoma State University
Stillwater, Oklahoma
2011

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfilment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2012

STABILITY ANALYSIS OF RECCURRENT

NEURAL-BASED CONTROLLERS

Dissertation Approved:

Dr. Martin Hagan

Dissertation Adviser

Dr. Carl D. Latino

Dr. George Scheets

Dr. Anthony Kable

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Overview

In the last two decades, more attention has been given to Recurrent Neural Networks (RNNs) and their applications. The ability of RNNs to solve complex problems in control, system identification, signal processing, communication, pattern recognition, etc. is well understood, and more research is taking place in this area. Although RNNs are more powerful than feedforward networks, this comes at the expense of more difficult training and the potential for instabilities. It has become more and more important to have efficient methods for determining the stability of RNNs.

RNNs are capable of handling severe nonlinearities, and their feedback connections bring memory to the network and empower RNNs to solve complex nonlinear control problems. Many studies have been devoted to the application of RNNs to solving control problems. Hagan *et al* [HaDe02] designed a NARMA-L2 controller, which is an RNN-based controller, for a magnetic levitation system. Magnetic levitation is an example of a highly nonlinear system, and the NARMA-L2 controller obtained excellent performance for this system. The performance of the NARMA-L2 controller has been verified through various

simulations for the magnetic levitation system. Despite the good performance, the potential instability of the overall closed-loop with a NARMA-L2 controller is the main disadvantage. The potential instability of RNNs also makes the training phase of RNNs challenging. The spurious valleys in the training error surface, which were studied in [JeHa09], are an immediate consequence of the potential instability of RNNs.

The problem of stability in RNNs has been studied by several other researchers. Wang and Xu [WaZo06] presented a set of sufficient and necessary conditions for Global Exponential Stability (GES) of a class of generic discrete-time RNNs. Fang and Kincaid [FaKi96] obtained sufficient conditions for local exponential stability of neural networks using the matrix measure technique. Suykens *et al* [SuVa96], [SuVa97-1] and [SuVa97-2] offered a new approach to the stability analysis of RNNs. They analyzed a specific form of RNN, which they called NL*q*. They derived a criteria for demonstrating the global stability of the origin. Their method, which is a Lyapunov based method, does not cover the case when an RNN has nonzero biases. Setting biases to zero severely limits the mapping capabilities of the RNN and also limits the stable ranges for the network weights. Another approach for the stability analysis of RNNs was introduced by Tanaka [Tana96]. Tanaka proves Global Asymptotic Stability (GAS) of the origin via Linear Matrix Inequalities (LMIs). Barabanov and Prokharov [BaPr02] also demonstrated GAS of the origin using LMIs. They claimed that the stability criteria derived in [SuVa97-2] and [Tana96] are special cases of their method. Liu [Liu07] derived a sufficient condition for stability of Delayed Standard Neural Network Model (DSNNM), which is more general than the model considered in [BaPr02].

The stability or instability of equilibrium points of RNNs depends on the network parameters: weights and biases. Denote M as a *space of stability* for a specific RNN architecture, and let it be defined as the set of all parameter values for which a given RNN is stable. The best stability analysis method would be the one that can identify the largest possible subset of M. Except in special cases, the exact determination of the full set M is not possible.

We will say that a stability criterion is restrictive, or conservative, to the extent that it does not identify the full set M. The criteria derived in all the existing methods are conservative, because many stable systems can be found where none of these criteria can be satisfied. To the best of our knowledge, the method of Reduction of Dissipativity Domain (RODD) introduced in [BaPr03] has the potential to provide the least restrictive stability criterion. Due to the importance of the RODD method, we devote Chapter 6 and Chapter 7 of this study to describing this method and to explaining how it might be modified.

Next, we will briefly explain the main objectives of this research.

Objectives

The main contribution of this research is to develop a new stability analysis method that can identify the largest possible subset of M in the shortest possible time, and then to design stable control systems with the developed methods.

The outline of this study is given in the following section.

Outline

This research can be divided into three phases: 1) Modeling, 2) Stability Analysis and 3) Testing. Chapter 2, Chapter 3 and Chapter 4 are devoted to the modeling phase,

Chapter 5, Chapter 6 and Chapter 7 are devoted to the stability analysis phase and Chapter 8 and Chapter 9 investigate the efficiency of the proposed algorithm with some problems.

In Chapter 2, we introduce the most general dynamical system representations: the state-space representation and the I/O representation. The choice of representations depends on the application. In this chapter, the necessary and sufficient conditions for the transformation of an observable state-space model to an I/O model will be derived.

In Chapter 3, we concentrate on the modeling of RNNs for the purpose of stability analysis. The state-space model is a suitable representation for stability analysis, hence we introduce different models that are examples of the state-space model, i.e. the NL_q and Lur'e models. After that, the BPRNN [BaPr02] will be introduced as a special type of RNN. This network will be the main topic of Chapter 4 and Chapter 5 in the derivation of the absolute stability criteria. The BPRNN is not in state-space form, so it will be transferred to the Lur'e model form using a special technique called the state-space extension method.

In Chapter 4, we prove that the transformed BPRNN is in the Lur'e model form by showing that it consists of a linear part in the forward path and a nonlinear part in the feedback path. We also demonstrate that the nonlinear part has a finite sector bound and satisfies a sector condition. After showing that the BPRNN model is in the Lur'e model form, we then study the absolute stability.

In Chapter 5, the results from the previous two chapters will be merged together to derive the final criteria for the absolute stability of the BPRNN. The final criteria, which is a Lyapunov based criteria, will be derived using the S-procedure method. The Lyapunov condition and the sector condition from Chapter 4 are merged together using the S-proce-

dure method. The derived criteria in this chapter is conservative, and there is a need to derive a less conservative criteria. This will be addressed in the next chapter.

In Chapter 6, we introduce a new approach for the stability analysis of RNNs ([BaPr03]). The RODD-LB1 method, which is believed to approximate the largest possible space of stable parameters, will be investigated in this chapter. The RODD-LB1 method gives a less conservative stability criteria, but it suffers from a slow rate of convergence. This makes the algorithm impractical for large networks. Hence, there is a need to fix this problem and develop a more efficient algorithm.

In Chapter 7, the RODD-LB2 method, which is an extension to the RODD-LB1 method, is developed. The RODD-LB2 method fixes the slow rate of convergence in the RODD-LB1 method by introducing a more accurate optimization. RODD-LB2 is an efficient version of RODD-LB1 that can detect stability in a shorter time. Both RODD-LB1 and RODD-LB2 use a linear approximation to the reachable sets. However, the reachable sets can also be approximated by quadratic functions. In the second section of this chapter, RODD-EB, which uses a quadratic approximation to the reachable sets, will be introduced. RODD-EB is an efficient algorithm in detecting stability for any dynamical system with elliptical reachable sets. In the third section of this chapter, RODD-Hybrid, which is a combination of RODD-LB2 and RODD-EB, will be proposed as the most efficient algorithm. The factors which affect the efficiency of RODD-EB are given at the end of this chapter.

In Chapter 8, several test problems will be given to investigate the performance of the proposed RODD algorithms from Chapter 7. The test problems are divided into stable

and unstable groups. In each group, we provided some systems in Lure form and some systems in non-Lure form.

In Chapter 9, two RNN-based controllers, MRAC and NARMA-L2, will be investigated. The RNN-based plant model and RNN-based controllers are cascaded together and make a closed loop system. The stability of the overall closed loop system will be investigated using the proposed methods from Chapter 7.

In Chapter 10, a summary of the research and potential future work are presented.

2 GENERAL NON-LINEAR SYSTEMS

In this chapter we will introduce the most common system representations: the state-space representation and the Input Output (I/O) representation. One may choose either model, depending on the application. In this study, we focus our attention on systems that are nonlinear, discrete-time, deterministic, time-invariant and have finite dimension. Choosing the right system representation will ease the job of stability analysis. The necessary and sufficient conditions for going back and forth between the state-space model and the I/O model will be described in this chapter.

The state-space representation is one of the important system representations. A Single-Input-Single-Output (SISO) discrete-time, deterministic, time-invariant, finite dimensional nonlinear state space model can be expressed as

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), u(k)) \\ y(k) &= h(\mathbf{x}(k)) \end{aligned} \tag{2.1}$$

where $\mathbf{x}(k) \in \mathbf{R}^n$ is the state of the system, $u(k) \in \mathbf{R}$ is the system input, $y(k) \in \mathbf{R}$ is the system output, $f : \mathbf{R}^n \times \mathbf{R} \to \mathbf{R}^n$, $h : \mathbf{R}^n \to \mathbf{R}$, f and h belong to C^∞ and are smooth functions. It is also assumed that $(\mathbf{0}, 0)$ is an equilibrium point of (2.1), hence $f(\mathbf{0}, 0) = 0$, and $h(\mathbf{0}) = 0$. If the origin is not the equilibrium point of the system above, then without loss of generality we can always transfer the non-zero equilibrium point to the origin via a change of variable.

A second standard system model is the Input-Output (I/O) map. A discrete-time deterministic, time-invariant, finite dimensional I/O map model has the form

$$y(k+d) = g(y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)) \tag{2.2}$$

where $g : \mathbf{R}^{2n} \to \mathbf{R}$.

In the following sections we will derive the conditions under which the state space model is equivalent to the I/O model.

State-Space Model to I/O Model

In this section we will show that under certain conditions an I/O model can be derived directly from a state space model.

Conditions for Conversion (Controllability-Observability)

An I/O model can be derived by eliminating states of the state model given in (2.1). The difficult part of this elimination is the nonlinearity involved in the state model. In the

general case, the conversion can be made by applying the *implicit function theorem.* The theorem is stated as follows**.**

Theorem 2.1: *(Implicit function Theorem): Let* n, m *be positive integer and* $\Psi_1, \ldots, \Psi_n$ *be differentiable functions* $\boldsymbol{\Psi} : \mathbf{R}^{n+m} \to \mathbf{R}^n$ *on a neighborhood of the point* $(\mathbf{x}_0,\mathbf{z}_0) = \left[x_1^0\ x_2^0\ \ldots\ x_n^0\ z_1^0\ z_2^0\ \ldots\ z_m^0\right]^T \in \mathbf{R}^{n+m}$ *if*

$$\begin{aligned} \Psi_1(\mathbf{x}_0,\mathbf{z}_0) &= 0 \\ \Psi_2(\mathbf{x}_0,\mathbf{z}_0) &= 0 \\ &\cdots \\ \Psi_n(\mathbf{x}_0,\mathbf{z}_0) &= 0 \end{aligned} \tag{2.3}$$

If the $n \times n$ *matrix*

$$\begin{bmatrix} \dfrac{\partial \Psi_1}{\partial x_1} & \dfrac{\partial \Psi_1}{\partial x_2} & \cdots & \dfrac{\partial \Psi_1}{\partial x_n} \\ \dfrac{\partial \Psi_2}{\partial x_1} & \dfrac{\partial \Psi_2}{\partial x_2} & \cdots & \dfrac{\partial \Psi_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \dfrac{\partial \Psi_n}{\partial x_1} & \dfrac{\partial \Psi_n}{\partial x_2} & \cdots & \dfrac{\partial \Psi_n}{\partial x_n} \end{bmatrix} \tag{2.4}$$

is non-singular at $(\mathbf{x}_0,\mathbf{z}_0)$*, then there is a neighborhood* $\mathbf{U}$ *of the point* $\mathbf{z}_0$ *in* $\mathbf{R}^m$*, there is a neighborhood* $\mathbf{V}$ *of the point* $\mathbf{x}_0$ *in* $\mathbf{R}^n$ *and there is a unique mapping* $\varphi : \mathbf{U} \to \mathbf{V}$ *such that* $\mathbf{x}_0 = \varphi(\mathbf{z}_0)$ *and*

$$\begin{aligned} \Psi_1(\varphi((\mathbf{z}), \mathbf{z})) &= 0 \\ \Psi_2(\varphi((\mathbf{z}), \mathbf{z})) &= 0 \\ &\cdots \\ \Psi_n(\varphi((\mathbf{z}), \mathbf{z})) &= 0 \end{aligned} \tag{2.5}$$

for all $\mathbf{z}$ *in* $\mathbf{U}$. *Further more,* φ *is differentiable. In other words if*

$\varphi(\mathbf{z}) = \left[g_1(\mathbf{z})\ g_1(\mathbf{z})\ \ldots\ g_n(\mathbf{z})\right]^T$ *where* $g_1, \ldots, g_n$ *are differentiable functions on* $\mathbf{U}$,

then

$$\begin{aligned} x_1 &= g_1(z_1, \ldots, z_m) \\ &\ldots \\ x_n &= g_n(z_1, \ldots, z_m) \end{aligned} \tag{2.6}$$

is the unique solution to the system (2.5) which satisfies $\mathbf{x}_0 = \varphi(\mathbf{z}_0)$ *near* $\mathbf{z}_0$. *Equation (2.6) can be rewritten in matrix format as follow*

$$\mathbf{x} = \varphi(\mathbf{z}) \tag{2.7}$$

In order to derive the I/O map from the state model, we need to evaluate system (2.1) sequentially for different time steps. The procedures are as follow

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), u(k)) \\ \mathbf{x}(k+2) &= f(\mathbf{x}(k+1), u(k+1)) \\ &= f(f(\mathbf{x}(k), u(k)), u(k+1)) \\ &\equiv f^2(\mathbf{x}(k), u(k), u(k+1)) \\ &\ldots \\ \mathbf{x}(k+n) &= f(\mathbf{x}(k+n-1), u(k+n-1)) \\ &= f(f\ldots(f(\mathbf{x}(k), u(k)), u(k+1)), \ldots, u(k+n-2)), u(k+n-1)) \\ &\equiv f^n(\mathbf{x}(k), u(k), u(k+1), \ldots, u(k+n-2), u(k+n-1)) \end{aligned} \tag{2.8}$$

where $\mathbf{x}(k) = \left[x_1(k)\ x_2(k)\ \ldots\ x_n(k)\right]^T$. By defining new variable

$\mathbf{v}(k) = \left[u(k)\ u(k+1)\ \ldots\ u(k+n-1)\right]^T$ equation (2.8) can be written in closed form as follows

$$\mathbf{x}(k+n) = \mathbf{f}^n(\mathbf{x}(k), \mathbf{v}(k)) \tag{2.9}$$

where $\mathbf{f}$ is a vector function with the i^{th} element

$$f^i(\mathbf{x}((k), \mathbf{v}(k))) = \underbrace{f(f \ldots (f}(\mathbf{x}(k), u(k)), u(k+1)), \ldots, u(k+i-1)) \;,\; 1 \le i \le n$$

Similarly, by evaluating the output equation $y(k)$ in (2.1) for different time steps the outcome is

$$\begin{aligned}
y(k) &= h(\mathbf{x}(k)) \\
y(k+1) &= h(\mathbf{x}(k+1)) \\
&\equiv h(f^1(\mathbf{x}(k), u(k)) \\
y(k+2) &= h(f(\mathbf{x}(k+1), u(k+1)) \\
&= h(f(f(\mathbf{x}(k), u(k)), u(k+1)) \\
&\equiv h(f^2(\mathbf{x}(k), u(k), u(k+1)) \\
&\cdots \\
y(k+n-1) &= h(f(\mathbf{x}(k+n-2), u(k+n-2)) \\
&= h(f^{n-1}(\mathbf{x}(k), u(k), u(k+1), \ldots, u(k+n-2))
\end{aligned} \tag{2.10}$$

Stacking all the outputs at different time steps together and defining new variable $\mathbf{w}(k) = \left[y(k)\ y(k+1)\ \ldots\ y(k+n-1)\right]^T$ we may rewrite (2.10) in a matrix format as follows

$$\begin{bmatrix} y(k) \\ y(k+1) \\ \cdots \\ y(k+n-1) \end{bmatrix} = \begin{bmatrix} h(\mathbf{x}(k)) \\ h(f^1(\mathbf{x}(k), u(k)) \\ \cdots \\ h(f^{n-1}(\mathbf{x}(k), u(k), u(k+1), \ldots, u(k+n-2)) \end{bmatrix} \tag{2.11}$$

which can be rewritten in closed form as

$$\mathbf{w}(k) = \mathbf{h}^n(\mathbf{x}((k), \mathbf{v}(k))) \tag{2.12}$$

Let us take into consideration the equation (2.10) again and move all the terms on the right hand side to the left hand side. Then

$$\begin{aligned}
y(k)-h(\mathbf{x}(k)) &= 0 \\
y(k+1)-h(\mathbf{x}(k+1)) &= y(k+1)-h(f(\mathbf{x}(k), u(k))) = 0 \\
y(k+2)-h(\mathbf{x}(k+2)) &= y(k+2)-h(f(\mathbf{x}(k+1), u(k+1))) \\
&= y(k+2)-h(f(f(\mathbf{x}(k), u(k)), u(k+1))) = 0 \\
&\quad \ldots \\
y(k+n-1)-h(\mathbf{x}(k+n-1)) &= y(k+n-1)-h(f(\mathbf{x}(k+n-2), u(k+n-2))) \\
&= y(k+n-1)-h(f(f \ldots f(\mathbf{x}(k), u(k)), u(k+1)), \ldots \\
&\quad u(k+n-3)), u(k+n-2))) \\
&= 0
\end{aligned} \tag{2.13}$$

Denote the left hand side of the above equations with the new functions $\Psi_1, \Psi_2, \ldots, \Psi_n$ where $\Psi_i : \mathrm{R}^{3n} \rightarrow \mathrm{R}$. Considering the new notation, the system of equations given in (2.13) can be rewritten as follow:

$$\begin{aligned}
&\Psi_1(x_1(k), \ldots, x_n(k), u(k), \ldots, u(k+n-1), y(k), \ldots, y(k+n-1)) = 0 \\
&\Psi_2(x_1(k), \ldots, x_n(k), u(k), \ldots, u(k+n-1), y(k), \ldots, y(k+n-1)) = 0 \\
&\ldots \\
&\Psi_n(x_1(k), \ldots, x_n(k), u(k), \ldots, u(k+n-1), y(k), \ldots, y(k+n-1)) = 0
\end{aligned} \tag{2.14}$$

The critical question here is if (2.14) can be solved for $\mathbf{x}(k)$ in terms of $\mathbf{w}(k)$. This question will be answered by applying Implicit function theorem. In order to apply the implicit function theorem, a definition of nominal point is needed. Consider the nominal point as the equilibrium point of system (2.1) given by

$$\begin{aligned}
\mathbf{x}^0(k) &= \left[x_1^0(k)\ x_2^0(k)\ \ldots\ x_n^0(k)\right]^T = \mathbf{0} \\
\mathbf{v}^0(k) &= \left[u^0(k)\ u^0(k+1)\ \ldots\ u^0(k+n-1)\right]^T = \mathbf{0} \\
\mathbf{w}^0(k) &= \left[y^0(k)\ y^0(k+1)\ \ldots\ y^0(k+n-1)\right]^T
\end{aligned} \tag{2.15}$$

Considering the assumptions $f(\mathbf{0}, 0) = 0$, $h(\mathbf{0}) = 0$ and evaluating system (2.1) at the nominal point we will get

$$\begin{aligned}
y^0(k) &= h(\mathbf{x}^0(k)) = h(\mathbf{0}) = 0 \\
y^0(k+1) &= h(\mathbf{x}^0(k+1)) = h(\mathbf{0}) = 0 \\
&\cdots \\
y^0(k+n-1) &= h(\mathbf{x}^0(k+n-1)) = h(\mathbf{0}) = 0
\end{aligned} \tag{2.16}$$

The equation above shows that the nominal point given in (2.15) satisfies the system of equation (2.14). In other words,

$$\begin{aligned}
\Psi_1(\mathbf{x}^0(k),\mathbf{v}^0(k), \mathbf{w}^0(k)) &= 0 \\
\Psi_2(\mathbf{x}^0(k),\mathbf{v}^0(k), \mathbf{w}^0(k)) &= 0 \\
&\cdots \\
\Psi_n(\mathbf{x}^0(k),\mathbf{v}^0(k), \mathbf{w}^0(k)) &= 0
\end{aligned} \tag{2.17}$$

This equation is identical to (2.3) considering $\mathbf{z}^0 = \left[\mathbf{v}^0(k)\ \mathbf{w}^0(k)\right]^T$, hence, the first condition of the implicit function theorem has been satisfied. The second condition that needs to be satisfied is the non singularity of (2.4), evaluated at the nominal point (2.15). Then by applying the implicit function theorem, there is a neighborhood $\mathbf{U}$ of the points $(\mathbf{v}^0(k), \mathbf{w}^0(k))$ in $\mathbf{R}^{2n}$, there is neighborhood $\mathbf{V}$ of the points $\mathbf{x}^0(k)$ in $\mathbf{R}^n$ and there is

unique mapping $\varphi : \mathbf{U} \to \mathbf{V}$ such that $\mathbf{x}^0(k) = \varphi(\mathbf{v}^0(k), \mathbf{w}^0(k))$ and

$$\begin{aligned} \psi_1(\varphi((\mathbf{v}, \mathbf{w}), \mathbf{v}, \mathbf{w})) &= 0 \\ \psi_2(\varphi((\mathbf{v}, \mathbf{w}), \mathbf{v}, \mathbf{w})) &= 0 \\ &\cdots \\ \psi_n(\varphi((\mathbf{v}, \mathbf{w}), \mathbf{v}, \mathbf{w})) &= 0 \end{aligned} \tag{2.18}$$

which means that (2.12) can be solved for $\mathbf{x}(k)$ in terms of $\mathbf{w}(k)$ and $\mathbf{v}(k)$. In other words,

$$\mathbf{w}(k) = \mathbf{h}^n(\varphi(\mathbf{w}(k), \mathbf{v}(k)), \mathbf{v}(k)) \tag{2.19}$$

Equation (2.19) is the solution of block output equations of system (2.1) for $\mathbf{x}(k)$ in terms of future input and future output. Similarly, solving $\mathbf{x}(k)$ in terms of $\mathbf{v}(k)$ and $\mathbf{w}(k)$ in the first equation of system (2.1) yields,

$$\begin{aligned} \mathbf{x}(k+n) &= \mathbf{f}^n(\varphi(\mathbf{w}(k), \mathbf{v}(k)), \mathbf{v}(k)) \\ &= \bar{\mathbf{g}}(y(k), \ldots, y(k+n-1), u(k), \ldots u(k+n-1)) \end{aligned} \tag{2.20}$$

where $\bar{\mathbf{g}} : \mathbf{R}^{2n} \to \mathbf{R}^n$ is a smooth vector function. Evaluating the output equation given in (2.1) at $k+n$ and (2.1) and the substitution $\mathbf{x}(k+n)$ derived in (2.20) the following input-output model is formed,

$$y(k+n) = h(\bar{\mathbf{g}}(y(k), \ldots, y(k+n-1), u(k), \ldots u(k+n-1)))$$

By changing the time index we will get

$$y(k) = g(y(k-n), y(k-n+1), \ldots, y(k-1), u(k-n), u(k-n+1), \ldots u(k-1)) \tag{2.21}$$

where $g : \mathbf{R}^{2n} \to \mathbf{R}$ is a scalar function which is the composition function $ho\bar{\mathbf{g}}(\bullet)$

Equation (2.21) can either be called the I/O model representation or the Non-linear-Auto-Regressive with eXogenous inputs (NARX) model. The NARX model given in (2.21) is an

exact representation of the system (2.1) in a neighborhood of origin. This model is derived under the condition that $rank\left(\frac{\partial \mathbf{h}^n}{\partial \mathbf{x}}\right) = n$ around the neighborhood of the origin. This condition guarantees the solvability of (2.12) for $\mathbf{x}(k)$ in terms of $\mathbf{w}(k)$ and $\mathbf{v}(k)$ in the neighborhood of the origin. In fact, a system with this property is called *locally first-order observable*. Therefore, any locally first-order observable system with the state model given in (2.1) is guaranteed to have an I/O representation (2.21) in the neighborhood of the origin.

Relative Degree

An important concept for system model is relative degree. Relative degree is defined as follows:

Definition 2.1: *Relative Degree[CaNa95] Let consider f_e denote the state dynamics $f(\ \cdot\ , 0)$ and f_e^i the i-times iterated composition of f_e for the system (2.1). The system has relative degree d at $(\mathbf{x}, u) = (\mathbf{0}, 0)$ if there exists a neighborhood $\chi_d \times U_d \subset \chi \times U$ of the equilibrium state $(\mathbf{0}, 0)$ for which the following conditions are satisfied*

$$\begin{aligned} &\frac{\partial (hof_e^k of(\mathbf{x}, u))}{\partial u} = 0 \quad for \quad 0 \le k \le d-2 \\ &\frac{\partial (hof_e^{d-1} of(\mathbf{x}, u))}{\partial u} \neq 0 \end{aligned} \tag{2.22}$$

where $hof_e^k of(\mathbf{x}, u)$ is in fact the output of system (2.1) at the time $k+1$. If $\frac{\partial (hof_e^k of(\mathbf{x}, u))}{\partial u} = 0$ for all k in some neighborhood $\chi_\infty \times U_\infty \subset \chi \times U$ we say that the relative degree of (2.1) at $(\mathbf{0}, 0)$ is $+\infty$. If there exists a $\bar{k} \ge 1$ and a neighborhood

$\chi_* \times U_* \subset \chi \times U$ *such that*

$$\left.\frac{\partial(hof_e^{\bar{k}}of(\mathbf{x}, u))}{\partial u}\right|_{\mathbf{0}, 0} = 0$$
$$\frac{\partial(hof_e^{\bar{k}}of(\mathbf{x}, u))}{\partial u} \neq 0, (x, u) \in \chi_* \times U_* - \{(\mathbf{0}, 0)\} \tag{2.23}$$

we say that the relative degree of (2.1) at $(\mathbf{0}, 0)$ *does not exist, or equivalently, that (2.1) doesn't have a well-defined relative degree at* $(\mathbf{0}, 0)$.

Qualitatively, relative degree implies that an input at instant k with the initial condition in the neighborhood of the equilibrium state affects the output only d units of time later. Consequently, when the relative degree is defined, it represents the delay of the system (2.1) between input $u(k)$ and output $y(k)$.

It was shown that if the state space model of a system given in (2.1) is first order observable, then the I/O representation can be derived as in (2.21). If system (2.1) has relative degree 1 ($d = 1$), then by the definition of relative degree

$\frac{\partial(hof_e^0of(\mathbf{x}(k), u(k)))}{\partial u(k)} = \frac{\partial(y(k+1))}{\partial u(k)} \neq 0$. Furthermore $y(k+1)$ does locally depend on $u(k)$. If $d \neq 1$ then $\frac{\partial(y(k+1))}{\partial u(k)} = 0$ which means that $y(k+1)$ does not depend on $u(k)$ around the neighborhood of the origin. For the case when $d \neq 1$ we may eliminate $u(k)$ in the input argument of (2.21) and rewrite it like the following equation

$$y(k+1) = \bar{g}_1(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), u(k-2), \ldots u(k-n+1)) \tag{2.24}$$

where $\bar{g}_1 : \mathbf{R}^{2n-1} \rightarrow \mathbf{R}$. This elimination is only valid locally around the neighborhood of the origin. Another iteration in (2.24) yields,

$$y(k+2) = \bar{g}_1(y(k+1), y(k), \ldots, y(k-n+2), u(k), u(k-1), \ldots u(k-n+2)) \tag{2.25}$$

Substitute $y(k+1)$ in (2.24) into (2.25) and the outcome will be as follows

$$y(k+2) = \bar{g}_1(\bar{g}_1(y(k), y(k-1), \ldots, y(k-n+1)), y(k), \ldots, y(k-n+2), u(k), u(k-1), \ldots, u(k-n+2)) \tag{2.26}$$

which can be rewritten as

$$y(k+2) = g_2(y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots u(k-n+1)) \tag{2.27}$$

where $g_2 : \mathbf{R}^{2n} \rightarrow \mathbf{R}$. If $d = 2$ then $\frac{\partial(y(k+2))}{\partial u(k)} \neq 0$. These guarantee the dependency of $y(k+2)$ on $u(k)$. Otherwise, if $d \neq 2$, then $\frac{\partial(y(k+2))}{\partial u(k)} = 0$. In this case, when $d \neq 2$, we may eliminate $u(k)$ in the input argument of (2.27) and rewrite it as the following

$$y(k+2) = \bar{g}_2(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), u(k-2), \ldots u(k-n+1))$$

where $\bar{g}_2 : \mathbf{R}^{2n-1} \rightarrow \mathbf{R}$. Again this elimination is only valid locally around the neighborhood of the origin. Continuing in the same manner and assuming that the system (2.1) has a finite relative degree d, the output at time $k+d$ is written as

$$y(k+d) = g_d(y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots u(k-n+1)) \tag{2.28}$$

where $g_d : \mathbf{R}^{2n} \rightarrow \mathbf{R}$ and $\frac{\partial(y(k+d))}{\partial u(k)} \neq 0$. The relative degree of d for the system (2.1) guarantees the dependency of $y(k+d)$ on $u(k)$ in (2.28). In the following section the I/O

model given in (2.28) will be used to derive the NARMA-L1 and NARMA-L2 models.

The I/O model given in (2.28) is not convenient for the sake of control because of non-linear dependence of all past input control signals on the desired reference signal. (For the case when the non-linear functions are too complex, computation of control signal will be tedious.) For the same reason, Narendra [NaMu97] proposed two approximations to the NARX model called NARMA-L1 and the NARMA-L2 models. The main feature of these models is that the control input $u(k)$ occurs linearly in the input-output model. Hence, the control signal can be solved in terms of the previous inputs and the reference signal. For the derivation of NARMA-L1, check .

NARMA-L2

The NARMA-L2 model is an approximation of the NARX model using Taylor series expansion. Denote

$$\mathbf{m}(k) = \left[y(k)\ y(k-1)\ \ldots\ y(k-n+1)\ u(k-1)\ u(k-2)\ \ldots\ u(k-n+1)\right]^T \in \mathbf{R}^{2n-1} \tag{2.29}$$

then (2.28) may be rewritten as

$$y(k+d) = g_d(\mathbf{m}(k), u(k)) \tag{2.30}$$

Expanding (2.30) using the Taylor series about the nominal point $u(k) = 0$ yields

$$y(k+d) = g_d(\mathbf{m}((k), 0)) + \frac{\partial}{\partial u(k)} g_d(\mathbf{m}(k), u(k))\Big|_{u(k)=0} u(k) + \mathbf{R}_1(\mathbf{m}(k), u(k)) \tag{2.31}$$

The second term on the right hand side of (2.31) is guaranteed to be non-zero. This is due to the fact that system (2.1) with I/O representation derived in (2.28) is assumed to have the relative degree d. By the Taylor Remainder theorem [Buck78], the last term on the right

hand side of (2.31), $\mathbf{R}_1(\mathbf{m}(k), u(k))$, yields

$$\mathbf{R}_1(\mathbf{m}(k), u(k)) = \frac{\partial^2}{\partial u^2(k)} g_d(\mathbf{m}(k), u(k))\Big|_{u(k)=\tau} \times \frac{u^2(k)}{2} \tag{2.32}$$

where τ is an appropriately chosen point between 0 and $u(k)$. Let M_2 be the maximum value of $\frac{\partial^2}{\partial u^2(k)} g(\mathbf{m}(k), u(k))\Big|_{u(k)=\tau}$; therefore we may rewrite equation (2.32) as the following inequality

$$\left|\mathbf{R}_1(\mathbf{m}(k), u(k))\right| \le \frac{M_2 u(k)^2}{2} \tag{2.33}$$

Since a continuous function attains a maximum in a compact set, the value of M_2 is bounded so the approximation can be made as accurately as desired by decreasing the amplitude of the input $u(k)$. This magnitude can be large, but in practice the magnitude of $\frac{\partial^2}{\partial u^2(k)} g(\mathbf{m}(k), u(k))$ is usually small. Considering a small upper bound for the remainder, (2.31) may be approximated as follows,

$$\begin{aligned} y(k+d) \cong & f_0(y(k), y(k-1), \ldots, y(k-n+1) + u(k-1), \ldots, u(k-n+1)) \\ & + g_0(y(k), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1))u(k) \end{aligned} \tag{2.34}$$

where $f_0, g_0 : \mathbf{R}^{2n-1} \to \mathbf{R}$. Equation (2.34) is an approximation of the NARX model that is called the NARMA-L2 model. The advantage of this model over the original NARX model is that the control signals at each time step can be solved linearly with respect to the reference points. Similar to NARMA-L1(see Appendix A), the NARMA-L2 model is a

good approximation of the NARX model only about the origin ($u(k) = 0$). For the cases where system (2.1) is not operating around the origin, a different method may be used to approximate the NARX model.

I/O Model to State-Space Model (Realization)

The previous section described how to develop an I/O model from a state-space model. This section will describe the reverse operation developing a state-space model from an I/O model which is called the *Realization* problem.

In this section, we will investigate an algorithm by Sadegh [Nade01] to derive the necessary and sufficient conditions for existence of a locally observable state-space representation of the I/O map in (2.21).

Consider the following state model given in (2.1) which is locally observable around the neighborhood of the origin. In (2.9) and (2.12) it is shown that the block state space representation can be derived as

$$\begin{aligned} \mathbf{x}(k+n) &= \mathbf{f}^n(\mathbf{x}(k), \mathbf{v}(k)) \\ \mathbf{w}(k) &= \mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k)) \end{aligned} \tag{2.35}$$

where $\mathbf{v}(k)$ is the future input vector and $\mathbf{w}(k)$ is the future output vector. For the case when the state model is locally observable around the neighborhood of the origin, the conditions of the implicit function theorem will be satisfied. Hence, the output equation in (2.35) can be solved for $\mathbf{x}(k)$ in terms of $\mathbf{w}(k)$ and $\mathbf{v}(k)$ (equation. (2.19)). Substituting (2.19) for $\mathbf{x}(k)$ into the first equation of (2.35) yields

$$\begin{aligned} \mathbf{x}(k+n) &= \mathbf{f}^n(\varphi(\mathbf{w}(k), \mathbf{v}(k)), \mathbf{v}(k)) \\ &= \Theta(\mathbf{w}(k), \mathbf{v}(k)) \end{aligned} \tag{2.36}$$

Substituting $\mathbf{x}(k)$ in (2.36) (by changing of the time index form $k+n \rightarrow k$) into the output equation in (2.35) yields,

$$\begin{aligned} \mathbf{w}(k) &= \mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k)) \\ &= \mathbf{h}^n(\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)), \mathbf{v}(k)) \end{aligned} \tag{2.37}$$

The above equation presents the block I/O map of the state model given in (2.1) under the condition that the state model is locally observable.

The other way to construct the block I/O model is to begin with NARX model derived in (2.21). Write this equation as

$$\begin{aligned} y(k) &= g(y(k-n), y(k-n+1), \ldots, y(k-1), u(k-n), u(k-n+1), \ldots u(k-1)) \\ &= g_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{aligned} \tag{2.38}$$

By iterating one step ahead in the above equation, $y(k+1)$ may be derived as

$$\begin{aligned} y(k+1) &= g(y(k-n+1), y(k-n+2), \ldots, y(k), u(k-n+1), u(k-n+2), \ldots u(k)) \\ &= g_2(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{aligned} \tag{2.39}$$

Substituting $y(k)$ in (2.38) with (2.39) proves that $y(k+1)$ is a function of $\mathbf{w}(k-n)$, $\mathbf{v}(k-n)$ and $\mathbf{v}(k)$. ($y(k+1)$ depends on $y(k)$, and $y(k)$ depends on $\mathbf{w}(k-n)$, $\mathbf{v}(k-n)$ and $\mathbf{v}(k)$). By iterating one step ahead in (2.39), $y(k+2)$ can be derived as

$$\begin{aligned} y(k+2) &= g(y(k-n+2), y(k-n+3), \ldots, y(k+1), u(k-n+2), u(k-n+3), \ldots u(k+1)) \\ &= g_3(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{aligned} \tag{2.40}$$

$y(k+2)$ is function of $\mathbf{w}(k-n)$, $\mathbf{v}(k-n)$ and $\mathbf{v}(k)$ because $y(k)$ and $y(k+1)$ are functions of $\mathbf{w}(k-n)$, $\mathbf{v}(k-n)$ and $\mathbf{v}(k)$. Similarly, the output sequence at the time $k+n-1$ can be derived as

$$\begin{aligned} y(k+n-1) &= g(y(k-1), y(k), \ldots, y(k+n-2), u(k-1), u(k), \ldots u(k+n-2)) \\ &= g_n(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{aligned} \tag{2.41}$$

By stacking all the output sequences together, the following block I/O map will be formed

$$\begin{bmatrix} y(k) \\ y(k+1) \\ \cdots \\ y(k+n-1) \end{bmatrix} = \begin{bmatrix} g_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \\ g_2(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \\ \cdots \\ g_n(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{bmatrix} \tag{2.42}$$

which can be rewritten in matrix format as

$$\mathbf{w}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \tag{2.43}$$

where $g : \mathbf{R}^{2n} \to \mathbf{R}$, $g_i : \mathbf{R}^{3n} \to \mathbf{R}$ for $1 \le i \le n$ $\mathbf{G} : \mathbf{R}^{2n} \to \mathbf{R}^n$ and the i^{th} row of (2.43) denoted by

$$\begin{aligned} g_i(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) = g(&y(k-n+i-1), y(k-n+i), \ldots, y(k-1), \\ &g_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)), g_2(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)), \ldots \\ &g_{i-1}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)), u(k-n+i-1), u(k-n+i), \ldots \\ &u(k-1), \ldots, u(k), u(k+1), \ldots, u(k+i-2)) \end{aligned}$$

By iterating one step ahead in (2.41) the $y(k+n)$ can be derived as follows

$$\begin{aligned} y(k+n) &= g(y(k), y(k+1), \ldots, y(k+n-1), u(k), u(k+1), \ldots u(k+n-1)) \\ &= g_{n+1}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \end{aligned} \tag{2.44}$$

Observable State Space Input Output Map (OSSIOM)

In this section we carefully define the conditions under which an I/O map model and a state variable model are equivalent.

Definition 2: *The I/O map*

$y(k) = g(y(k-n), \ldots, y(k-1), u(k-n), \ldots u(k-1))$ *is said to be Observable State Space I/O Map (OSSIOM) if it has an equivalent* n *dimensional observable state space realization.*

Lemma 2.1: *Let* $\mathbf{G}$ *in (2.43) be a block OSSIOM. Then* $D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))$ *is non-singular and* $\{D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))\}^{-1}\{D_{v(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))\}$ *is independent of the third variable* $\mathbf{v}(k)$ *on a neighborhood of the origin.*

Proof: Let

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), u(k)) \\ y(k) &= h(\mathbf{x}(k)) \end{aligned} \tag{2.45}$$

be an equivalent observable state space realization of an I/O map $y(k) = g(y(k-n), \ldots, y(k-1), u(k-n), \ldots u(k-1))$. Then the block I/O map (2.37) derived from the block state equation and the block I/O map derived in (2.43) are identical. Hence,

$$\mathbf{w}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) = \mathbf{h}^n(\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)), \mathbf{v}(k)) \tag{2.46}$$

Differentiating the above equation with respect to $\mathbf{w}(k-n)$ and using the chain rule yields

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) &= \frac{\partial}{\partial \mathbf{x}(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))\frac{\partial}{\partial \mathbf{w}(k-n)}\mathbf{x}(k) \\ &= D_{x(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))D_{w(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)) \end{aligned} \tag{2.47}$$

Similarly, differentiating (2.46) with respect to $\mathbf{v}(k-n)$ yields,

$$\begin{aligned} \frac{\partial}{\partial \mathbf{v}(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) &= \frac{\partial}{\partial \mathbf{x}(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))\frac{\partial}{\partial \mathbf{v}(k-n)}\mathbf{x}(k) \\ &= D_{x(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))D_{v(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)) \end{aligned} \tag{2.48}$$

Rewriting (2.19) for the time index $k-n$ and differentiating both sides with respect to $\mathbf{w}(k-n)$ will be as follows

$$\begin{aligned} I_{n\times n} &= \frac{\partial}{\partial \mathbf{x}(k-n)}\mathbf{h}^n(\mathbf{x}(k-n), \mathbf{v}(k-n))\frac{\partial}{\partial \mathbf{w}(k-n)}\varphi(\mathbf{w}(k-n), \mathbf{v}(k-n)) \\ &= D_{x(k-n)}\mathbf{h}^n(\mathbf{x}(k-n), \mathbf{v}(k-n))D_{w(k-n)}\varphi(\mathbf{w}(k-n), \mathbf{v}(k-n)) \end{aligned} \tag{2.49}$$

Differentiating (2.36) with respect to $\mathbf{w}(k-n)$ yields

$$\begin{aligned} D_{w(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)) &= D_{x(k-n)}\mathbf{f}^n(\mathbf{x}(k-n), \mathbf{v}(k-n)) \\ & D_{w(k-n)}\varphi(\mathbf{w}(k-n), \mathbf{v}(k-n)) \end{aligned} \tag{2.50}$$

By the lemma assumption, the state equation needs to be observable, so $D_{x(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))$ is non-singular. By (2.49), $D_{w(k-n)}\varphi(\mathbf{w}(k-n), \mathbf{v}(k-n))$ should be an invertible matrix. Also, assuming that the state equation is invertible guarantees non-singularity of $D_{x(k-n)}\mathbf{f}^n(\mathbf{x}(k-n), \mathbf{v}(k-n))$. Then the two pieces on the right hand side of (2.50) are invertible, as is their product. This is enough to say that $D_{w(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n))$ is non-singular. Similarly, the two pieces on the right hand side of (2.47) will be non-singular so $D_{w(k-n)}(\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)))$ is also non-singular. Solving $D_{x(k)}\mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))$ in (2.47) and substituting in (2.48) yields

$$\begin{aligned} D_{v(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) &= D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) \\ \{D_{w(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n))\}^{-1} & D_{v(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)) \end{aligned}$$

Rearranging the above equation yields

$$\{D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))\}^{-1} D_{v(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) = \{D_{w(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n))\}^{-1} D_{v(k-n)}\Theta(\mathbf{w}(k-n), \mathbf{v}(k-n)) \tag{2.51}$$

The lemma is proved, because the right hand side of the above equation is independent of $\mathbf{v}(k)$. □

The above lemma proves the sufficient conditions for an I/O map to be OSSIOM. After defining the state vector, the necessary conditions will be stated in lemma 2.

Let the state vector candidate be defined as

$$\mathbf{x}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{0}) \tag{2.52}$$

This means that the future inputs, $\mathbf{v}(k)$, should set equal to zero in the block I/O map. Iterate the state vector candidate one step ahead of time

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{G}(\mathbf{w}(k-n+1), \mathbf{v}(k-n+1), \mathbf{0}) \\ &\equiv \mathbf{G}_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}^*(k)) \end{aligned} \tag{2.53}$$

where

$$\begin{aligned} \mathbf{G}_1 &= (g_2(\ ), g_3(\ ), \ldots, g_{n+1}(\ ))^T \\ \mathbf{v}^*(k) &= (u(k), 0, \ldots, 0)^T \end{aligned} \tag{2.54}$$

From the definition of the output and previous input vector, the last element of $\mathbf{w}(k-n+1)$ and $\mathbf{v}(k-n+1)$ is $y(k)$ and $u(k)$ accordingly. Also by (2.21) the output $y(k)$ depends on $\mathbf{w}(k-n)$ and $\mathbf{v}(k-n)$, so (2.53) can be rewritten in terms of new function called $\mathbf{G}_1$ that depends on $\mathbf{w}(k-n)$, $\mathbf{v}(k-n)$ and $u(k)$. Assuming that the derivative of $\mathbf{G}$ with respect to $\mathbf{w}(k-n)$ is non-singular around in the neighborhood of the origin, then by the implicit function theorem there exists a unique function $\mathbf{G}_w^{-1}$ such that

$\mathbf{w}(k-n)$ can be written in terms of $\mathbf{x}(k)$ and $\mathbf{v}(k-n)$. Let's rewrite (2.53) with the inverse function $\mathbf{G}_w^{-1}$ as follows

$$\mathbf{x}(k+1) = \mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}(k), \mathbf{v}(k-n), \mathbf{0}), \mathbf{v}(k-n), \mathbf{v}^*(k)) \tag{2.55}$$

Lemma 2.2: *Suppose that the block I/O map (2.43) is such that* $D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))$ *is non-singular and* $\{D_{w(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))\}^{-1}\{D_{v(k-n)}\mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k))\}$ *is independent of the third variable* $\mathbf{v}(k)$ *on a neighborhood of the origin. Defining the state vector* $\mathbf{x}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}((k-n), \mathbf{0}))$ *then*

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) = \mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}(k), \mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{v}^*(k)) \\ y(k) &= h(\mathbf{x}(k)) = x_1(k) \end{aligned} \tag{2.56}$$

Proof: The equation given in (2.55) is the state equation equivalent to (2.43) if we could prove that the right hand side does not depend on $\mathbf{v}(k-n)$. In the other words, it needs to be proved that

$$\begin{aligned} \frac{\partial}{\partial \mathbf{v}(k-n)}\mathbf{x}(k+1) &= \frac{\partial}{\partial \mathbf{w}(k-n)}\mathbf{G}_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}^*(k))\frac{\partial}{\partial \mathbf{v}(k-n)}\mathbf{G}_w^{-1}(\mathbf{x}(k), \mathbf{v}(k-n), \mathbf{0}) \\ &\quad + \frac{\partial}{\partial \mathbf{v}(k-n)}\mathbf{G}_1(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}^*(k)) \\ &= 0 \end{aligned}$$

Ignore the time index in (2.52) and rewrite it as follows

$$\mathbf{x} = \mathbf{G}(\mathbf{w}, \mathbf{u}, \mathbf{v}) \tag{2.57}$$

where $\mathbf{w}$, $\mathbf{u}$ and $\mathbf{v}$ are the place holders for previous outputs, previous inputs and future inputs. Notice that for the case when $\mathbf{v} = \mathbf{0}$ (2.57) is identical to the state vector. Since the

derivative of $\mathbf{G}$ with respect to $\mathbf{w}$ is assumed to be non-singular in the neighborhood of the origin, then applying the implicit function theorem yields

$$\mathbf{x} = \mathbf{G}(\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}), \mathbf{u}, \mathbf{v}) \tag{2.58}$$

Differentiating both sides with respect to $\mathbf{u}$ yields,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{u}} = D_w\mathbf{G}(\mathbf{w}, \mathbf{u}, \mathbf{v})D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}) + D_u\mathbf{G}(\mathbf{w}, \mathbf{u}, \mathbf{v})$$

Both the left hand side and the right hand side are independent of $\mathbf{u}$. Solving for $D_u\mathbf{G}_w^{-1}$ yields

$$D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = -\{D_w\mathbf{G}(\mathbf{w}, \mathbf{u}, \mathbf{v})\}^{-1}D_u(\mathbf{G}(\mathbf{w}, \mathbf{u}, \mathbf{v})) \tag{2.59}$$

By the second condition of the lemma, the right hand side of above equation is independent of $\mathbf{v}$, and so is the left hand side. Since $D_u\mathbf{G}_w^{-1}$ is independent of $\mathbf{v}$ the following equation is corrected

$$D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}) = D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{0}) \tag{2.60}$$

By the definition of G_1 (2.53) may be written in terms of place holder as follows

$$\mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}), \mathbf{u}, \mathbf{v}) = (x_2, x_3, \ldots, g_{n+1}(\mathbf{x}, \mathbf{v})) \tag{2.61}$$

Since the above equation does not depend on $\mathbf{u}$, its derivative with respect to $\mathbf{u}$ is zero.

$$D_w\mathbf{G}_1(\mathbf{w, u, v})D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v}) + D_u\mathbf{G}_1(\mathbf{w, u, v}) = \mathbf{0}$$

Substituting (2.60) in above equation for $D_u\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{v})$ yields

$$D_u\mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{0}), \mathbf{u}, \mathbf{v}) = \mathbf{0} \tag{2.62}$$

Since the derivative of above equation with respect to previous inputs is equal to zero,

$$\mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{u}, \mathbf{0}), \mathbf{u}, \mathbf{v}) = \mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}, \mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{v}) \tag{2.63}$$

Comparing (2.55) and (2.63) yields

$$\begin{aligned} \mathbf{x}(k+1) &= f(x(k), u(k)) = \mathbf{G}_1(\mathbf{G}_w^{-1}(\mathbf{x}(k), \mathbf{0}, \mathbf{0}), \mathbf{0}, \mathbf{v}^*(k)) \\ y(k) &= x_1(k) \end{aligned} \tag{2.64}$$

This proves the first part of the lemma. Considering that $y(k)$ only depends on previous outputs and previous inputs, and by the definition of the state vector, the output is always identical to the first element of the state vector. For the second part of the lemma, it needs to be proven that (2.64) is the first order observable state space equation. The block state space equation is equivalent to the block I/O map given in the lemma

$$\mathbf{w}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{v}(k)) = \mathbf{h}^n(\mathbf{x}(k), \mathbf{v}(k))$$

Considering the definition of the state vector one will get

$$\mathbf{x}(k) = \mathbf{G}(\mathbf{w}(k-n), \mathbf{v}(k-n), \mathbf{0}) = \mathbf{h}^n(\mathbf{x}(k), \mathbf{0}) \tag{2.65}$$

Differentiating both sides with respect to $\mathbf{x}(k)$ around the neighborhood of the origin yields

$$I_{n \times n} = \frac{\partial^n}{\partial \mathbf{x}} \mathbf{h}(\mathbf{0}, \mathbf{0})$$

which guarantees the absorbability condition.

In the following example, we will demonstrate the conversion of the I/O maps to state-space models.

Example 3.1 *Consider the following I/O map*

$$y(k) = y^2(k-2) + u(k-1) + u^2(k-2). \tag{2.66}$$

The first step is to find the block I/O map. Iterating one step ahead in the above equation yields

$$y(k+1) = y^2(k+1) + u(k) + u^2(k+1)$$

Furthermore, the block I/O map is

$$\begin{bmatrix} y(k) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} y^2(k-2) + u(k-1) + u^2(k-2) \\ y^2(k-1) + u(k) + u^2(k-1) \end{bmatrix} \tag{2.67}$$

By (2.52) the state equation is

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \begin{bmatrix} y^2(k-2) + u(k-1) + u^2(k-2) \\ y^2(k-1) + u^2(k-1) \end{bmatrix} \tag{2.68}$$

The next step is to find $\mathbf{G}_1$ *by iterating one step ahead in (2.68).*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} y^2(k-1) + u(k) + u^2(k-1) \\ \{y^2(k-2) + u(k-1) + u^2(k-2)\}^2 + u^2(k) \end{bmatrix} \tag{2.69}$$

where we plug (2.66) for $y(k)$ *in the above equation. Also,* $\mathbf{G}_w^{-1}$ *needs to be found, where (2.68) should be solved for* $y(k-1)$ *and* $y(k-2)$ *in terms of the other variables. Considering (2.56),* $y(k-1)$ *and* $y(k-2)$ *are given by*

$$\begin{aligned} y(k-2) &= \pm\sqrt{x_1(k)} \\ y(k-1) &= \pm\sqrt{x_2(k)} \end{aligned} \tag{2.70}$$

Using lemma 2 to derive the state equation, we need to plug (2.70) in to (2.69) by ignoring the inputs except the current input. Consequently, the following equation is formed

$$
\begin{aligned}
x_1(k+1) &= x_2(k) + u(k) \\
x_2(k+1) &= x_1^2(k) + u^2(k) \\
y(k) &= x_1(k)
\end{aligned}
\tag{2.71}
$$

This is the state-space equivalent of the I/O map of (2.66).

Conclusion

In this chapter, two major system representations, the state-space model and the I/O model, were discussed. We showed that under certain conditions these two models are identical. The modelling of a physical system is an important process, because the form of a model can ease its analysis. Since the two models are identical, we can study the behavior of a system under the model that is easier to analyze. In Chapter 4, we will show that the Barabanov-Prokharov model is not easy to use for stability analysis, hence the equivalent state-space model will be used.

In the next chapter, different models of recurrent neural networks will be studied.

3 NEURAL NETWORK MODEL OF NONLINEAR SYSTEM

In the previous chapter two different representations (state-space model and input-output model) of non-linear dynamic systems were derived. The previous chapter was mainly devoted to finding the sufficient conditions for going from a state model to an I/O model and back. The subject of this chapter is to introduce different Recurrent Neural Network (RNN) models and to investigate a method of transforming the original network to another equivalent network which is convenient for stability analysis. The state model introduced in the previous chapter will be used as a convenient form for the stability analysis of certain types of RNNs.

Modeling of Recurrent Neural Network (RNN)

RNNs are examples of non-linear dynamic systems. The inherent feedback properties of RNNs, make these type of dynamic networks more difficult to analyze than static neural networks. The RNN has memory, which makes it more powerful than static net-

works. RNNs are widely used in different areas of engineering such as control, pattern recognition, signal processing, etc. The main concern when applying an RNN for any application is stability. The stability analysis of the RNN will be addressed in detail in this research.

Generally a neural network which has a feedback connection is an RNN. There is no single standard representation for these type of networks. However, they could be categorized as fully-connected RNNs or semi-connected RNNs. In fully-connected RNNs, each neuron is connected to every neuron (including itself) through a delay. In semi-connected RNNs, some of these connections are missing. The following sections will present some standard RNN frameworks, for semi-connected RNNs.

Barabanov-Prokhorov (BP) RNN Framework

One special type of semi-connected RNN model was introduced by Barabanov and Prokhorov in [BaPr03]. The Barabanov-Prokhorov (BP) RNN model is a network with global feedback connections. The BPRNN model can be represented by

$$\begin{aligned}
\mathbf{a}^1(k+1) &= \mathbf{f}^1(\mathbf{LW}^{1,1}\mathbf{a}^1(k) + \mathbf{LW}^{1,M}\mathbf{a}^M(k) + \mathbf{b}^1) \\
\mathbf{a}^2(k+1) &= \mathbf{f}^2(\mathbf{LW}^{2,2}\mathbf{a}^2(k) + \mathbf{LW}^{2,1}\mathbf{a}^1(k+1) + \mathbf{b}^2) \\
\mathbf{a}^3(k+1) &= \mathbf{f}^3(\mathbf{LW}^{3,3}\mathbf{a}^3(k) + \mathbf{LW}^{3,2}\mathbf{a}^2(k+1) + \mathbf{b}^3) \\
&\cdots \\
\mathbf{a}^M(k+1) &= \mathbf{f}^M(\mathbf{LW}^{M,M}\mathbf{a}^M(k) + \mathbf{LW}^{M,M-1}\mathbf{a}^{M-1}(k+1) + \mathbf{b}^M)
\end{aligned} \tag{3.1}$$

In this model M is the number of layers, S_j is the number of neurons in layer j, $\mathbf{f}^j : \mathbf{R}^{S_j} \to \mathbf{R}^{S_j}$ is the vector of activation functions for the layer j, $\mathbf{a}^j(k) \in \mathbf{R}^{S_j}$ for all $j = 1, \ldots, M$ is the output vector of layer j at time step k, $\mathbf{LW}^{j,k} \in \mathbf{R}^{S_j \times S_k}$ is a constant

matrix of weights connecting layer k to layer j, and $\mathbf{b}^j \in \mathbf{R}^{S_j}$ is a fixed vector of biases. The schematic view of the network (3.1) is shown in Figure (3.1). The figure illustrates that every layer of this network is has a local feedback connection. In Chapter 5, the sufficient condition for the absolute stability of this network will be derived.

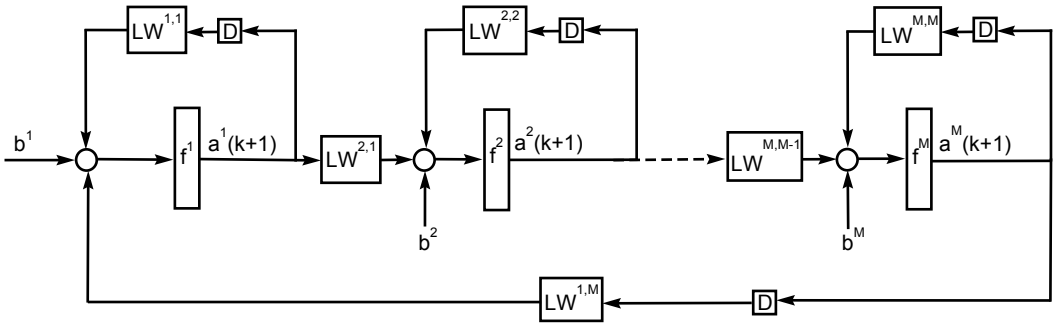

Figure (3.1) Schematic view of system (3.1)

NLq RNN Framework

Another general framework that has been proposed for RNNs is the NLq system. NLq neural networks are discrete time nonlinear dynamic systems in the state space form that contain q layers with alternating linear and nonlinear operators satisfying a sector condition (The sector condition will be introduced in Chapter 4). The NLq system concept was first introduced in [SuVa96]. The NLq system with constant external inputs can be represented as

$$\mathbf{a}(k+1) = \mathbf{f}_1(\mathbf{LW}_1\mathbf{f}_2(\mathbf{LW}_2 \ldots \mathbf{f}_M(\mathbf{LW}_M\mathbf{a}(k) + \mathbf{B}_M) \ldots + \mathbf{B}_2) + \mathbf{B}_1) \qquad (3.2)$$

where $\mathbf{a}(k) \in \mathbf{R}^M$ is the state vector at time step k, $\mathbf{LW}_j$ are constant matrices $j = 1, \ldots, M$ and $\mathbf{f}_j$ is a vector function consisting of activation functions for layer j. All

the elements of vector function $\mathbf{f}_j$ are defined by the identity map, $1(s) = s$, except the $M - j + 1$ element, which is defined by and $\mathbf{f}^i$. The system (3.1) can be transformed into the form of (3.2) using the following matrices:

$$\mathbf{LW}_1 = \begin{bmatrix} \mathbf{I} & 0 & 0 & \dots & 0 \\ 0 & \mathbf{I} & 0 & \dots & 0 \\ \dots & & & & \\ 0 & \dots & & \mathbf{I} & 0 \\ 0 & \dots & \mathbf{LW}^{M,M-1} & & \mathbf{LW}^{M,M} \end{bmatrix}, \dots, \mathbf{LW}_M = \begin{bmatrix} \mathbf{LW}^{1,1} & 0 & \dots & \mathbf{LW}^{1,M} \\ 0 & \mathbf{I} & 0 \dots & 0 \\ \dots & & & \\ 0 & \dots & \mathbf{I} & 0 \\ 0 & \dots & & \mathbf{I} \end{bmatrix}$$

$$\mathbf{B}_1 = \begin{bmatrix} 0 & 0 & \dots & \mathbf{b}_M \end{bmatrix}^T, \dots, \mathbf{B}_{M-1} = \begin{bmatrix} 0 & \mathbf{b}_2 & \dots & 0 \end{bmatrix}^T, \mathbf{B}_M = \begin{bmatrix} \mathbf{b}_1 & 0 & \dots & 0 \end{bmatrix}^T$$

$$\mathbf{f}_1 = \begin{bmatrix} 1 & 1 & \dots & \mathbf{f}^M \end{bmatrix}^T, \dots, \mathbf{f}_{M-1} = \begin{bmatrix} 1 & \mathbf{f}^2 & \dots & 1 \end{bmatrix}^T, \mathbf{f}_M = \begin{bmatrix} \mathbf{f}^1 & 1 & \dots & 1 \end{bmatrix}^T \tag{3.3}$$

Suykens derived sufficient conditions for global asymptotic stability of system (3.2) when the biases are assumed to be zeros. The derived criterion by Suykens was not practical because of the lack of consideration of the biases. Ignoring the biases not only severely limits the mapping capabilities of neural networks but also makes the stability criterions more difficult to be satisfied. The stability analysis of a general case when the biases are non-zero will be investigated in Chapter 5.

Generalized Lur'e Model

Another general framework that can represent certain types of RNNs is called the Generalized Lur'e Model. For instance, the BPRNN model given in (3.1) can be transformed into a generalized Lur'e model. This transformation can be done using the state space extension method (explained in the next section). The generalized Lur'e model can be represented as

$$\begin{aligned}\mathbf{y}(k+1) &= \mathbf{A}\mathbf{y}(k) + \mathbf{B}\boldsymbol{\eta}(k) \\ \mathbf{n}(k) &= \mathbf{W}\mathbf{y}(k) + \mathbf{b} = \left[n_1(k), n_2(k) \ \ldots \ n_l(k)\right]^T \\ \boldsymbol{\eta}(k) &= \mathbf{f}(\mathbf{n}(k))\end{aligned} \tag{3.4}$$

where $\mathbf{y}(k) \in \mathbf{R}^n$, $\boldsymbol{\eta}(k) \in \mathbf{R}^s$, $\mathbf{A} \in \mathbf{R}^{n \times n}$, $\mathbf{B} \in \mathbf{R}^{n \times s}$, $\mathbf{W} \in \mathbf{R}^{s \times n}$, $\mathbf{b} \in \mathbf{R}^s$, $\mathbf{n} \in \mathbf{R}^s$ and $f_i(s)$ is assumed to be $\tanh(s)$ for all the activation functions (n is the number of states, s is the number of inputs and outputs).

Transformation of the BPRNN model into the generalized Lur'e model requires $\mathbf{A}$, $\mathbf{B}$, $\mathbf{W}$ and $\mathbf{b}$ to follow certain structures. This transformation can be done using the state space extension method. The state space extension method converts the BPRNN model to a higher order generalized Lur'e model that is equivalent in terms of stability. Both systems are equivalent in terms of stability because the solution set of the BPRNN is the sub-sampled solution of the generalized Lur'e model.

The equilibrium point of (3.4) is not necessarily located at the origin, because the vector $\mathbf{b}$ is not necessarily zero. To be consistent, we always transfer make a change of variable that transfer the non-zero equilibrium point to the origin. We can then derive the stability criterion for the origin. Consider (3.4), where $\mathbf{b} \neq 0$ and $\mathbf{z}$ is the equilibrium point, then the equilibrium point equation can be rewritten as

$$\mathbf{z} = \mathbf{A}\mathbf{z} + \mathbf{B}\tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) \tag{3.5}$$

Consider the affine transformation $\mathbf{x} = \mathbf{y} - \mathbf{z}$ and denote $\mathbf{c} = \mathbf{W}\mathbf{z} + \mathbf{b}$, then we may re-write (3.4) with the new variable as follows

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{a}(k) \\ \boldsymbol{\sigma}(k) &= \mathbf{W}\mathbf{x}(k) \\ \mathbf{a}(k) &= \mathbf{f}(\boldsymbol{\sigma}(k)) = \tanh((\boldsymbol{\sigma}(k)) + \mathbf{c}) - \tanh(\mathbf{c}) \end{aligned} \tag{3.6}$$

where the origin is the equilibrium point. Transforming of the non-zero equilibrium point of system (3.4) to a zero equilibrium point of a system given in (3.4) changes the activation function from $\tanh(s)$ to $\tanh(s+c) - \tanh(c)$. We will derive the sector bound of the new activation function in Chapter 4.

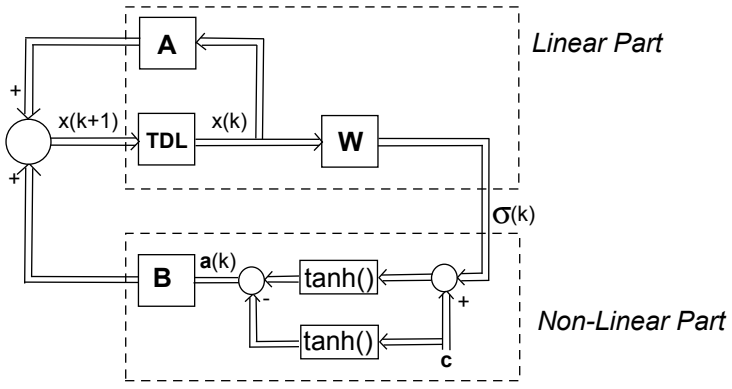

Figure (3.2) Generalized Lur'e system

Figure (3.2) is the block diagram representation of system (3.6). This system consists of a linear part and a non-linear part. The linear part, which is located in the feed-forward path, has the following transfer function

$$\frac{\boldsymbol{\sigma}(z)}{\mathbf{a}(z)} = \mathbf{W}(z\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$$

If the nonlinear part, which is located in the feedback path of system (3.6), satisfies certain sector conditions, then this system can be analyzed using the theory of absolute stability. A special class of functions which satisfy sector conditions will be discussed in detail in the following chapter, and the absolute stability criterion will be derived in Chapter 5. The

schematic view of the generalized Lur'e model is given in Figure (3.3).

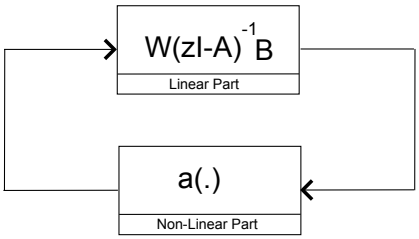

Figure (3.3) Simplified view of Generalized Lur'e system

In the following section, we will present a method of transferring system (3.1) into an equivalent system which is convenient for stability analysis. The equivalent model is in the form of system (3.4). The relationship between the solutions of the original model and the equivalent model will also investigated.

State Space Extension Method [BaPr02]

The state space extension method is a technique we can use to transform the BPRNN model (3.1) into a generalized Lur'e model. The solutions of the transformed system are just repetitions of the solutions of the original system with some time delays. So, convergence of the solutions of the transformed system guarantees the convergence of solutions of the original system. In other words, both systems are equivalent in terms of stability.

Consider the BPRNN given in (3.1). Applying the state space extension method to this system will transform system (3.1) into a new system. The dynamics of the l^{th} layer of the new system can be represented by

$$
\begin{aligned}
\mathbf{y}_1^l(k+1) &= \mathbf{f}^l(\mathbf{LW}^{l,l}\mathbf{y}_M^l(k) + \mathbf{LW}^{l,l-1}\mathbf{y}_1^{l-1}(k) + \mathbf{b}^l) \\
\mathbf{y}_2^l(k+1) &= \mathbf{y}_1^l(k) \qquad\qquad l = 1, \ldots, M \\
\mathbf{y}_3^l(k+1) &= \mathbf{y}_2^l(k) \\
&\cdots \\
\mathbf{y}_M^l(k+1) &= \mathbf{y}_{M-1}^l(k)
\end{aligned}
\tag{3.7}
$$

Writing (3.7) for all layers gives a generalized Lur'e system (3.4) with

$s = S_1 + S_2 + \ldots + S_M$ and $n = M \cdot s$ where $\mathbf{f} : \mathbf{R}^s \to \mathbf{R}^s$, $\mathbf{f} = \left[\mathbf{f}^1\ \mathbf{f}^2\ \ldots\ \mathbf{f}^M\right]$,

$\mathbf{f}^j : \mathbf{R}^{S_j} \to \mathbf{R}^{S_j}, \mathbf{f}^j = \left[f_1^j\ f_2^j\ \ldots\ f_{S_j}^j\right]$ $f_d^j : \mathbf{R} \to \mathbf{R}$, $(1 \le d \le S_j, \quad 1 \le j \le M)$

By comparing the original system, Figure (3.1), and the transformed system, Figure (3.4), it can be observed that the transformed system has $M - 1$ more delays in every layer.

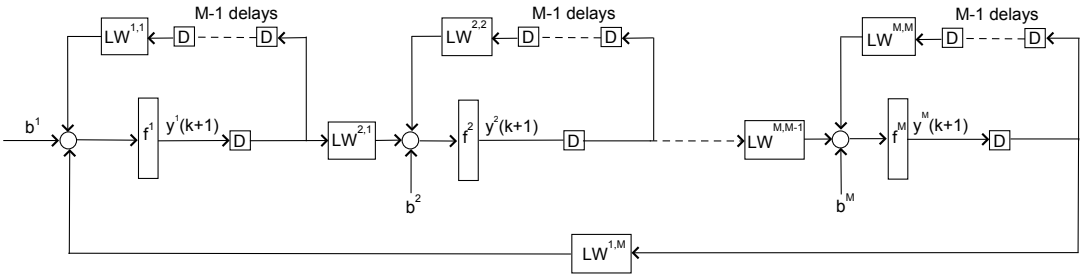

Figure (3.4) Extended system (3.7)

In the following example we will illustrate the special structure of the matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{W}$ and $\mathbf{b}$.

Example 3.1 *Consider the two-layer BPRNN given in (3.1) with three neurons in each layer ($M = 2, S_1 = 2, S_2 = 3$). This network can be represented by*

$$\begin{aligned}\mathbf{a}^1(k+1) &= \mathbf{f}^1(\mathbf{LW}^{1,1}\mathbf{a}^1(k) + \mathbf{LW}^{1,2}\mathbf{a}^2(k) + \mathbf{b}^1)\\ \mathbf{a}^2(k+1) &= \mathbf{f}^2(\mathbf{LW}^{2,2}\mathbf{a}^2(k) + \mathbf{LW}^{2,1}\mathbf{a}^1(k+1) + \mathbf{b}^2)\end{aligned} \tag{3.8}$$

By rewriting (3.7) for $l = 1, 2$ the following new system will be formed

$$\begin{aligned}\mathbf{y}_1^1(k+1) &= \mathbf{f}^1(\mathbf{LW}^{1,1}\mathbf{y}_2^1(k) + \mathbf{LW}^{1,2}\mathbf{y}_1^2(k) + \mathbf{b}^1)\\ \mathbf{y}_2^1(k+1) &= \mathbf{y}_1^1(k)\\ \mathbf{y}_1^2(k+1) &= \mathbf{f}^2(\mathbf{LW}^{2,2}\mathbf{y}_2^2(k) + \mathbf{LW}^{2,1}\mathbf{y}_1^1(k) + \mathbf{b}^2)\\ \mathbf{y}_2^2(k+1) &= \mathbf{y}_1^2(k)\end{aligned} \tag{3.9}$$

Then (3.9) can be rewritten in the generalized Lur'e form as follows

$$\begin{bmatrix}\mathbf{y}_1^1(k+1)\\ \mathbf{y}_2^1(k+1)\\ \mathbf{y}_1^2(k+1)\\ \mathbf{y}_2^2(k+1)\end{bmatrix} = \begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{I}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{I}&\mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{y}_1^1(k)\\ \mathbf{y}_2^1(k)\\ \mathbf{y}_1^2(k)\\ \mathbf{y}_2^2(k)\end{bmatrix} + \begin{bmatrix}\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}\\ \mathbf{0}&\mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{v}^1(k)\\ \mathbf{v}^2(k)\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{n}^1(k)\\ \mathbf{n}^2(k)\end{bmatrix} = \begin{bmatrix}\mathbf{0}&\mathbf{LW}^{1,1}&\mathbf{LW}^{1,2}&\mathbf{0}\\ \mathbf{LW}^{2,1}&\mathbf{0}&\mathbf{0}&\mathbf{LW}^{2,2}\end{bmatrix}\begin{bmatrix}\mathbf{y}_1^1(k)\\ \mathbf{y}_2^1(k)\\ \mathbf{y}_1^2(k)\\ \mathbf{y}_2^2(k)\end{bmatrix} + \begin{bmatrix}\mathbf{b}^1\\ \mathbf{b}^2\end{bmatrix} \tag{3.10}$$

where $\mathbf{y}(k) = \left[\mathbf{y}_1^1(k)\ \mathbf{y}_2^1(k)\ \mathbf{y}_1^2(k)\ \mathbf{y}_2^2(k)\right]^T \in \mathbf{R}^{10\times 1}$, $\mathbf{a}(k) = \left[\mathbf{v}^1(k)\ \mathbf{v}^2(k)\right]^T \in \mathbf{R}^{5\times 1}$,

$\mathbf{n}(k) = \left[\mathbf{n}^1(k)\ \mathbf{n}^2(k)\right]^T \in \mathbf{R}^{5\times 1}$, $\mathbf{b} = \left[\mathbf{b}^1\ \mathbf{b}^2\right]^T \in \mathbf{R}^{5\times 1}$ and $\mathbf{v}^1(k), \mathbf{v}^2(k)$ are defined by

$$\mathbf{a}(k) = \mathbf{f}(\mathbf{n}(k)) = \begin{bmatrix} \mathbf{v}^1(k) \\ \mathbf{v}^2(k) \end{bmatrix} = \begin{bmatrix} \mathbf{f}^1(\mathbf{n}^1(k)) \\ \mathbf{f}^2(\mathbf{n}^2(k)) \end{bmatrix} \quad \mathbf{f}^1 : \mathbf{R}^2 \to \mathbf{R}^2, \mathbf{f}^2 : \mathbf{R}^3 \to \mathbf{R}^3 \qquad (3.11)$$

The solution of the original network (3.1) is a sub-sampled version of the solution of the extended network (3.4). This is enough to say that convergence of the solutions of (3.4) guarantee the convergence of the solutions of (3.1). In other words, the two systems are equivalent in the sense of stability.

Conclusion

This chapter was devoted to introducing some existing frameworks for representing RNNs. Advantages and disadvantages of certain architectures for representing RNNs were also discussed. We introduced a transformation technique to convert BPRNN models into generalized Lur'e models through the state space extension method. The Lur'e model has been chosen as a convenient form for stability analysis. In the following chapter the sector condition and quadratic form will represented. These are required for the derivation of the absolute stability criterion for the system (3.1), which will be investigated in Chapter 5.

4 SECTOR CONDITIONS AND QUADRATIC FORM

In the previous chapter we showed that, by the state space extension method, the BPRNN model could be converted to a generalized Lur'e model, which is more convenient for stability analysis. The two models proved to be equivalent in the sense of stability. After the transformation, the new model with a non-zero equilibrium point could be converted to another model with zero equilibrium point through a change of variable. The process of transforming the non-zero equilibrium point to a zero equilibrium point changes the activation function. The original model (3.1) has the tangent hyperbolic activation function, while the transformed model with zero equilibrium point has the transfer function given in (3.6). The new transfer function belongs to a class of functions which satisfy a certain condition called the sector condition.

The main contribution of this chapter is to define the sector condition and derive quadratic forms for the activation function of the transformed system given in (3.6). These quadratic forms will be used in the next chapter to derive a criterion for stability of the

BPRNN model. In the following section, the definition of the sector condition will be given in more detail.

Sector Condition

In general, activation functions of standard neural networks are infinitely differentiable and bounded. They can be categorized as a special class of functions, if they satisfy a certain condition called the *sector condition.* Let $a : \mathbf{R} \to \mathbf{R}$ be a bounded smooth function satisfying the following condition

$$l \le \frac{a(s)}{s} \le u \qquad s \in (-\infty, \infty) \ , \ s \neq 0, \quad l, u \in \mathbf{R} \tag{4.1}$$

Then $a(s)$ is said to belong to the sector $[l, u]$ where the l is the lower sector bound and u is the upper sector bound. An example of a function that belongs to the sector $[l, u]$ is illustrated schematically in Figure (4.1).

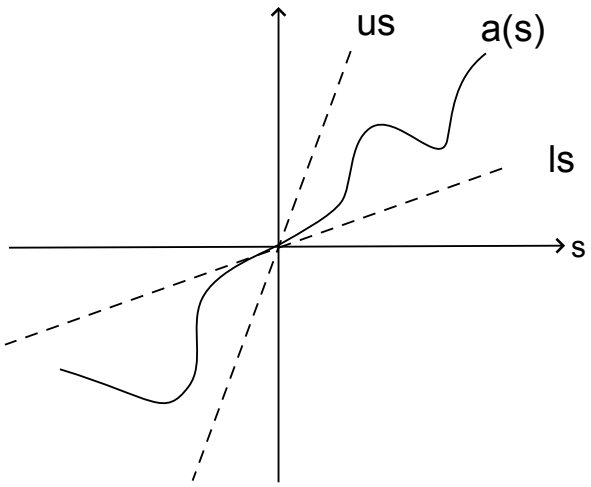

Figure (4.1) Linear Sector Condition

Sector Condition Constraint

The sector condition given in (4.1) can be decomposed into two separate inequalities. Multiply both sides of the inequality (4.1) by $s > 0$, which will be

$$ls \le a(s) \le us$$

and for $s < 0$

$$ls \ge a(s) \ge us$$

If $s > 0$ then

$$a(s) - ls \ge 0 \qquad \text{and} \qquad us - a(s) \ge 0 \tag{4.2}$$

and if $s < 0$ then

$$ls - a(s) \ge 0 \qquad \text{and} \qquad a(s) - us \ge 0 \tag{4.3}$$

The condition (4.2) or (4.3) imply the following inequality

$$[a(s) - ls][us - a(s)] \ge 0 \tag{4.4}$$

The inequality given in (4.4) is a sufficient condition that (4.1) be true, but it is not a necessary condition.

The network under investigation for stability (3.1) uses $\tanh(s)$ as an activation function for all the layers, so it is important to study the behavior of this function. It is easy to show that $f(s) = \tanh(s)$ satisfies the following conditions

1. f is a of class C^2
2. f is an odd function, which means $f(-s) = -f(s)$ and $f(0) = 0$
3. $\lim_{s \to \infty} f(s) = 1$ and $\lim_{s \to -\infty} f(s) = -1$
4. $f'(s) > 0$ for $s \in R$

$$5.\ f''(s)\begin{cases} positive & ;\ s<0 \\ negative & ;\ s>0 \\ 0 & ;\ s=0 \end{cases}$$

To determine the sector that $\tanh(s)$ belongs to, we must investigate $\frac{\tanh(s)}{s}$. By inspection of Figure (4.2), we can see that $\frac{\tanh(s)}{s}$ is bounded and lies in the interval $[0, 1]$.

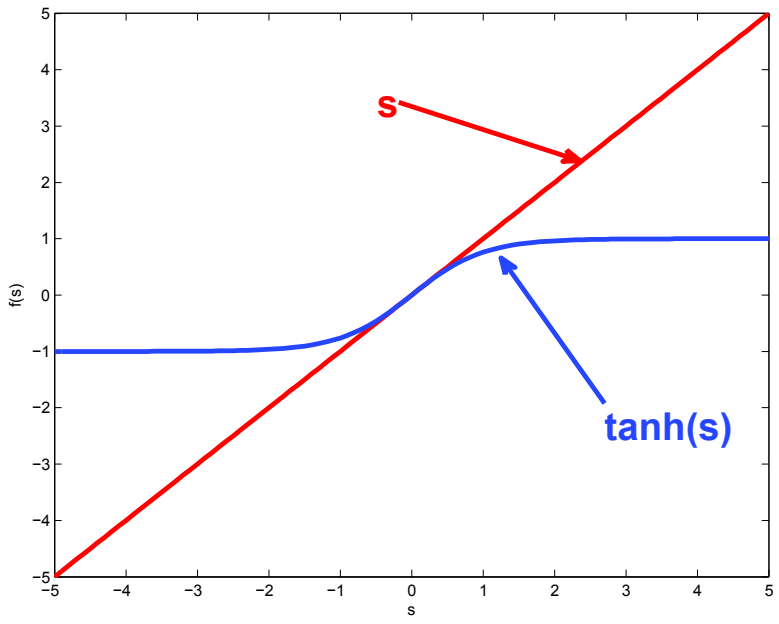

Figure (4.2) Hyperbolic tangent function

In other words

$$0 \le \frac{\tanh(s)}{s} \le 1 \tag{4.5}$$

This result is also shown in Figure (4.3). As $s \to -\infty$ or $s \to \infty$, $\frac{\tanh(s)}{s}$ goes to zero, because the denominator becomes extremely large while the numerator is bounded. Since the

maximum slope of $\tanh(s)$ occurs at the origin and is equal to one, so the maximum of $\frac{\tanh(s)}{s}$ occurs at the origin and is equal to one.

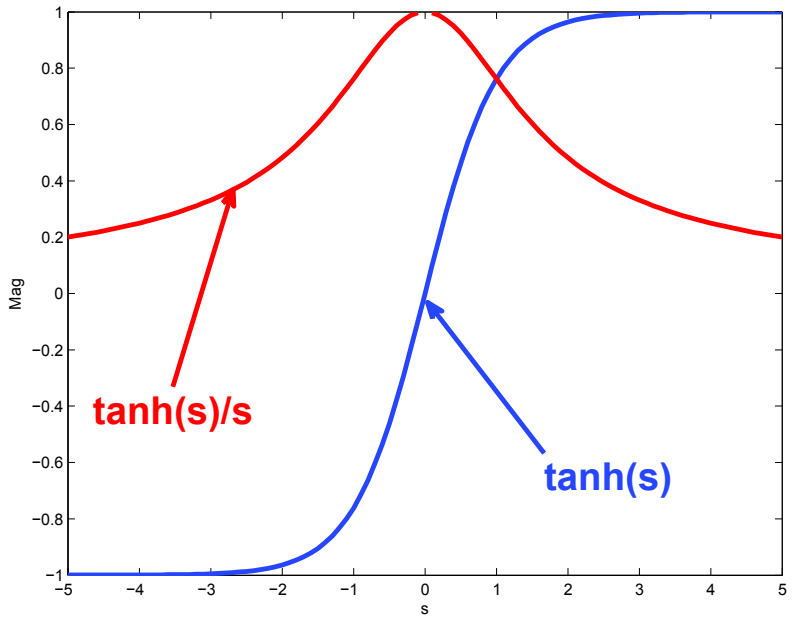

Figure (4.3) $\tanh(s)$ and its sector bound

The affine transformation $\mathbf{x} = \mathbf{y} - \mathbf{z}$ transformed model (3.4) with non-zero equilibrium point into a new model (3.6) with zero equilibrium point. Moreover, the activation function changed from $\tanh(s)$ to $\tanh(s+c) - \tanh(c)$ for $c \neq 0$. For the absolute stability analysis of (3.6) (explained in the next chapter) it is crucial to show that

$$a(s) = \tanh(s+c) - \tanh(c) \tag{4.6}$$

has a finite sector bound. Figure (4.3) illustrates the largest sector width, 1, which occurs when $c = 0$. Figure (4.4) illustrates a case when $c \neq 0$, where the width of the sector is less than unity

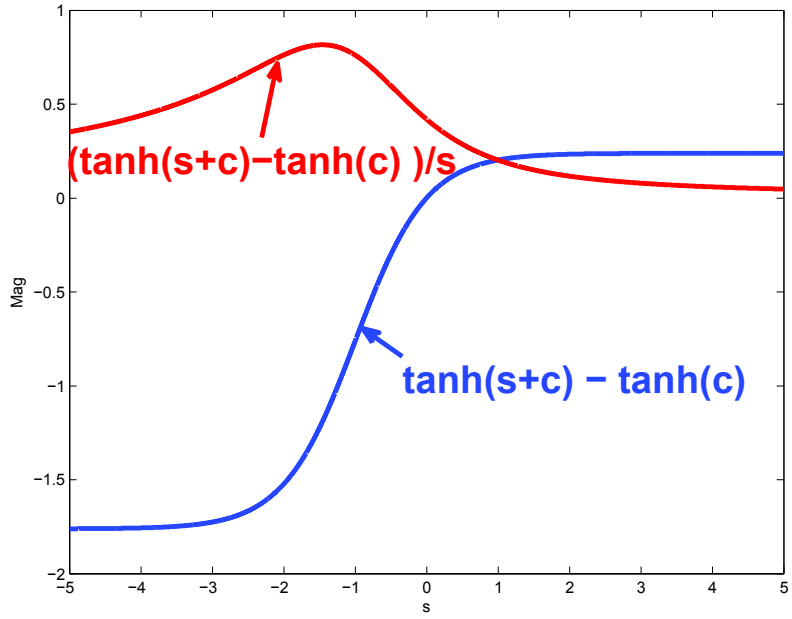

Figure (4.4) $a(s)$ and its sector bound when $c = 1$

The function $a(s)$ defined in (4.6) becomes the effective activation function for neural networks with non-zero biases. The sector of $a(s)$ is defined by

$$g_c(s) = \begin{cases} \dfrac{\tanh(s+c) - \tanh(c)}{s} & s \neq 0 \\ 1 - \tanh^2(c) & s = 0 \end{cases} \tag{4.7}$$

The upper and lower limits for $g_c(s)$ will define the sector for $a(s)$. Define $s' = s + c$. Then a shifted version of $g_c(s)$ would be

$$g_c^{sh}(s') = g_c(s' - c) = \begin{cases} \dfrac{\tanh(s') - \tanh(c)}{s' - c} & s' \neq c \\ 1 - \tanh^2(c) & s' = c \end{cases} \tag{4.8}$$

The upper and lower limits of this function are the same as those of $g_c(s)$, and can also be

used to define the sector of $a(s)$. Now consider Figure (4.5), which shows a plot of $\tanh(s')$. The slope of the chord shown in the figure is equal to $g_c^{sh}(s')$. The figure shows the $s' = s^*$ location that corresponds to the maximum slope, and therefore the upper limit of the sector of $a(s)$. From the figure, we can see that this slope can not be larger that 1, and it would only be equal to 1 for $c = 0$.

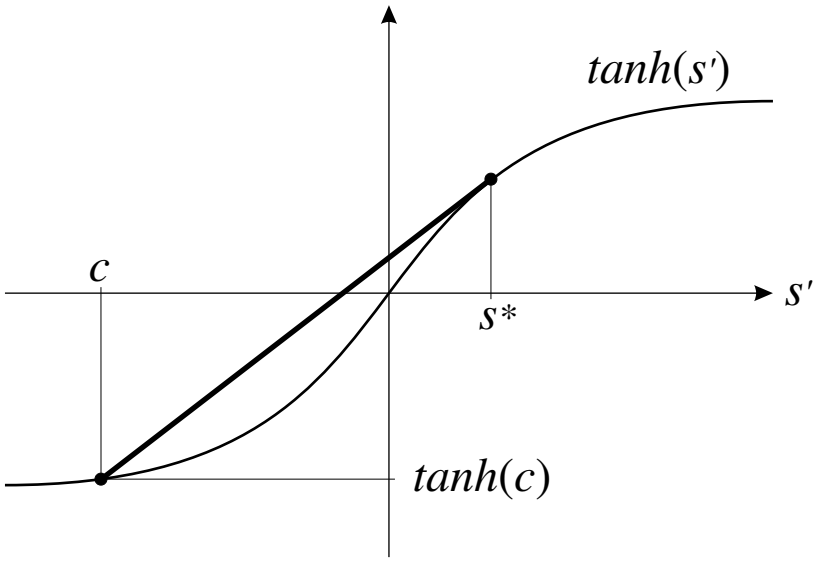

Figure (4.5) Illustration of the sector condition

In the next theorem we will list some of the important properties of the function g_c^{sh}, which will be used later in this section.

Theorem 4.1 *The function g_c^{sh} in (4.8) satisfies the following properties:*

1. $g_c^{sh}(s)$ is of class C^1.

2. $\lim_{s \to -\infty} g_c^{sh}(s) = \lim_{s \to \infty} g_c^{sh}(s) = 0$

3. There exists unique $c' \in [-c, 0]$ such that $g_c^{sh}(c') = 1 - \tanh^2(c')$

$$4.\begin{cases} \dfrac{d\, g_c^{sh}(s)}{d\, s} > 0 & ; \quad s < c' \\ \dfrac{d\, g_c^{sh}(s)}{d\, s} < 0 & ; \quad s > c' \\ \dfrac{d\, g_c^{sh}(s)}{d\, s} = 0 & ; \quad s = c' \end{cases}$$

Proof See Appendix A.

The length of the sector of $a(s)$ (which is the difference between the upper and lower sector bound) plays an important role in the stability of the BPRNN. As we will show in the next chapter, the shorter the length of the sector of $a(s)$, the easier the stability criterion can be satisfied. The maximum upper bound of the sector $a(s)$ can be derived by finding the root of the numerator of $\frac{d}{ds}\left(\frac{a(s)}{s}\right)$. This is because $\frac{a(s)}{s}$ has a unique maximum.

Theorem 4.2 *(Sector Upper Bound) Let* $a : \mathbf{R} \to \mathbf{R}$ *be defined by (4.6), where* $c \in \mathbf{R}$ *is a constant, then* $\frac{a(s)}{s}$ *has a unique maximum.*

Proof: This theorem can be proved by showing how the derivative of $\frac{a(s)}{s}$ will change sign for different values of c. Differentiating $\frac{a(s)}{s}$ with respect to s yields,

$$\frac{d}{ds}\left(\frac{a(s)}{s}\right) = \frac{s[1 - \tanh^2(s+c)] - [\tanh(s+c) - \tanh c]}{s^2} \tag{4.9}$$

Define new functions $f_1 : \mathbf{R} \to \mathbf{R}$, $f_2 : \mathbf{R} \to \mathbf{R}$ and $f_3 : \mathbf{R} \to \mathbf{R}$ as

$$\begin{aligned} f_1(s) &\equiv \frac{\tanh(s+c) - \tanh(c)}{s}, \\ f_2(s) &\equiv 1 - \tanh^2(s+c), \\ f_3(s) &= f_2(s) - f_1(s) \cdot \end{aligned} \tag{4.10}$$

Using the new functions defined in (4.10), the derivative in (4.9) can be rewritten as

$$\frac{d}{ds}\left(\frac{a(s)}{s}\right) = \frac{f_2(s) - f_1(s)}{s}. \tag{4.11}$$

The derivative in (4.9) at the point $s = 0$ is not defined. Using L'Hôpital's rule yields

$$\begin{aligned}\lim_{s \to 0} \frac{d}{ds}\left(\left(\frac{a(s)}{s}\right)\right) &= \frac{1 - \tanh^2(s+c) - 2s\tanh(s+c)[1 - \tanh^2(s+c)] - [1 - \tanh^2(s+c)]}{2s} \\ &= \frac{s\tanh^3(s+c) - s\tanh(s+c)}{s}\end{aligned} \tag{4.12}$$

Since the right hand side of (4.12) is still not defined at $s = 0$, applying L'Hôpital's rule again yields

$$\lim_{s \to 0} \frac{d}{ds}\left(\left(\frac{a(s)}{s}\right)\right) = \tanh^3(c) - \tanh(c) \tag{4.13}$$

The constant c can have three possible cases:

Case 1: $c = 0$. In this case the derivative $\frac{d}{ds}\left(\frac{a(s)}{s}\right)$ is equal to zero at the point $s = 0$. This is the only point where the derivative is equal to zero, because f_1 and f_2 never cross at any other point. This is because f_3 never changes sign:

$$f_3(s) = 1 - \tanh^2(s) - \frac{\tanh(s)}{s} < 0 \quad \forall s \in \mathbf{R} \tag{4.14}$$

[For the proof of (4.14) check Corollary A.1, Figure (A.4)]. Hence the origin is the unique critical point of $\frac{a(s)}{s}$. This unique critical point is the maximum point because

$$\frac{d}{ds}\left(\frac{a(s)}{s}\right) = \frac{1-\tanh^2(s)-\dfrac{\tanh(s)}{s}}{s} \begin{cases} > 0 & ; s < 0 \\ < 0 & ; s > 0 \end{cases} \tag{4.15}$$

In other words, $\frac{a(s)}{s}$ is monotonically increasing for negative s and monotonically decreasing for positive s; hence, the origin must be the maximum point. Figure (4.6) illustrates the above argument.

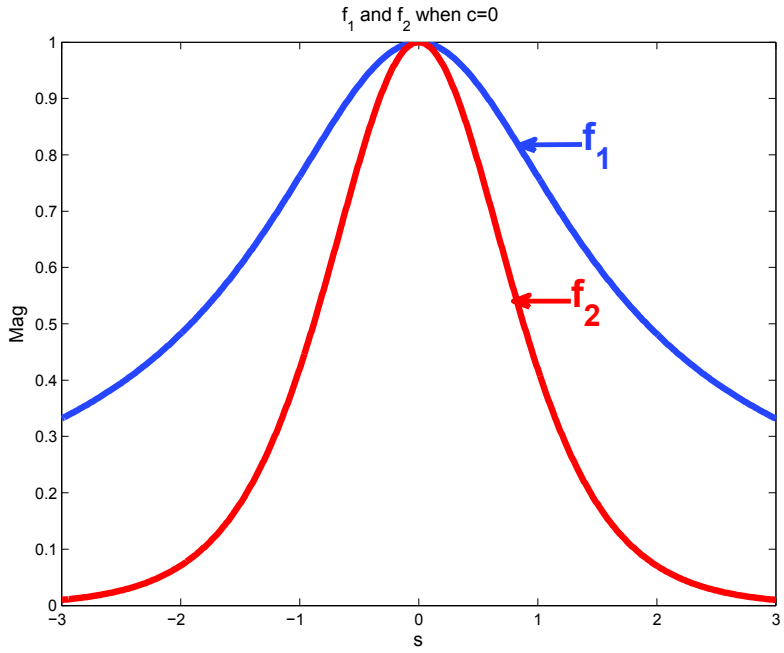

Figure (4.6) f_1 and f_2 when $c = 0$

Case 2: $c > 0$ (The case $c < 0$ would be similar) In this case, f_1 and f_2 do not cross at the origin, because $f_3(0) \neq 0$. In order to find where f_3 changes sign, we will evaluate f_3 at two separate points $s = -c$ and $s = -2c$, as follows:

$$
\begin{aligned}
f_3(-c) &= 1 - \frac{\tanh(c)}{c} \\
f_3(-2c) &= 1 - \tanh^2 c - \frac{\tanh(c)}{c}
\end{aligned}
\tag{4.16}
$$

$f_3(-c) > 0$ because $0 < \frac{\tanh(c)}{c} < 1$. Also, $f_3(-2c) < 0$ [Check , Figure (A.4) for the proof]. Since $f_3(s)$ is a continuous function, by the intermediate value theorem, there exists $s_1 \in [-2c, -c]$ such that $f_3(s_1) = 0$. So f_1 and f_2 cross at $s = s_1$ and the derivative $\frac{d}{ds}\left(\frac{a(s)}{s}\right)\Big|_{s = s_1} = 0$. This point is the maximum of $\frac{a(s)}{s}$, because the derivative of $\frac{a(s)}{s}$ changes sign from negative to positive in the interval $[-2c, -c]$.

$$
\begin{aligned}
\frac{d}{ds}\left(\frac{a(s)}{s}\right)\Big|_{-2c} &= \frac{1 - \tanh^2 c - \frac{\tanh c}{c}}{-2c} > 0 \\
\frac{d}{ds}\left(\frac{a(s)}{s}\right)\Big|_{-c} &= \frac{-c + \tanh c}{c^2} < 0
\end{aligned}
\tag{4.17}
$$

Figure (4.7) shows an example of the case when $c = 1$ and the maximum of the sector of $a(s)$ occurs at the point $s_1 \in [-2c, -c]$.

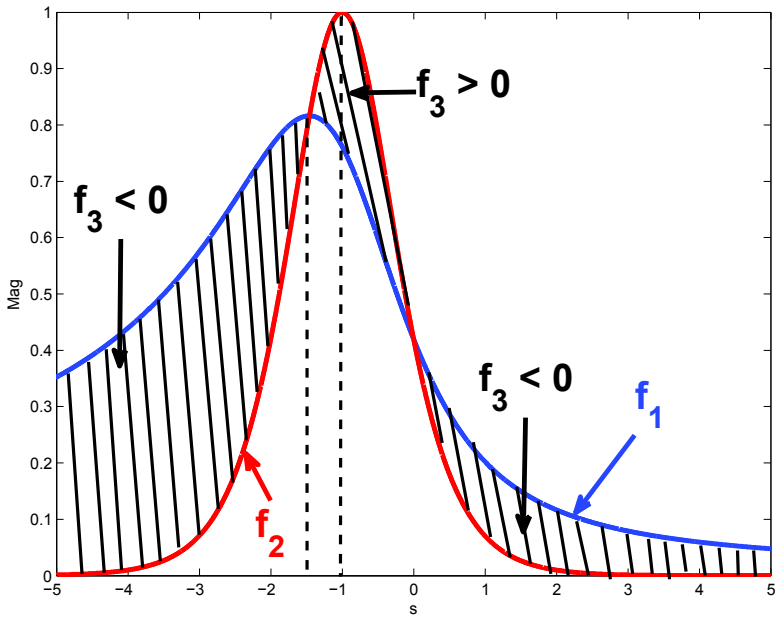

Figure (4.7) f_1 and f_2 when $c = 1$

To show the uniqueness of the maximum point s_1, it needs to be proven that s_1 is the only point where the derivative of $\frac{a(s)}{s}$ is zero. According to the proof given in , Lemma A.2 the sign of $f_3(s)$ changes as follows:

$$\begin{cases} f_3(s) < 0 \;\; ; \;\; -\infty < s < s_1 \\ f_3(s) > 0 \;\; ; \;\; s_1 < s < 0 \\ f_3(s) < 0 \;\; ; \;\; 0 < s < \infty \end{cases} \tag{4.18}$$

The manner in which the sign of (4.18) changes in different regions is illustrated in Figure (4.7). According to (4.14), $s = s_1$ and $s = 0$ are the only roots of $f_3(s)$. Since at $s = 0$ the derivative is not zero, hence the point $s_1 \in [-2c, -c]$ is the unique maximum point of

$\frac{a(s)}{s}$. □

For the derivation of the length of the sector of $a(s)$, the lower sector bound also needs to be calculated. Considering the range of s to be $(-\infty, \infty)$, the minimum lower bound of the sector is always zero, because

$$\lim_{s \to \infty} \frac{\tanh(s+c) - \tanh(c)}{s} = \lim_{s \to -\infty} \frac{\tanh(s+c) - \tanh(c)}{s} = 0 \tag{4.19}$$

However, the operating range of a BPRNN is a compact subset of $\mathbf{R}^n$, which needs to be calculated to obtain an exact derivation of the lower sector bound. The derivation of lower sector bound of $a(s)$ is given in Theorem 4.3.

The exact operation range of a BPRNN can be calculated by the affine transformation $\mathbf{x} = \mathbf{y} - \mathbf{z}$ and defining $\mathbf{c} = \mathbf{W}\mathbf{z} + \mathbf{b}$ in system (3.6) as follows:

$$\begin{aligned} |\sigma + c| &= \left|\mathbf{w}^T\mathbf{x} + \mathbf{w}^T\mathbf{z} + b\right| \\ &= \left|\mathbf{w}^T(\mathbf{x} + \mathbf{z}) + b\right| \\ &= \left|\mathbf{w}^T\mathbf{y} + b\right| \end{aligned} \tag{4.20}$$

The above equality can be written for every neuron of a BPRNN. According to (3.4) and the particular structure of $\mathbf{A}$ and $\mathbf{B}$, the absolute value of each coordinate of the vector $\mathbf{y}(k)$ is less than or equal to one. This is because every element of $\mathbf{y}(k)$ is the output of the hyperbolic tangent function. This will assure the existence of a sector upper bound for (4.20), and we can convert it to the following inequality:

$$|\sigma + c| \le \sum_{k=1}^{n} |w_k| + |b| \\ = r \tag{4.21}$$

In the following theorem, the exact lower bounds of the sector $a(s)$ will be calculated based on the sector upper bound given in (4.21).

Theorem 4.3 *(Sector Lower Bound) Let* $a : \mathbf{R} \to \mathbf{R}$ *be defined in (4.6) where* c *is a real number. If* $\sup_s (s + c) = r$ *then* $\frac{\tanh(r) - \tanh|c|}{r - |c|}$ *is the lower bound of the sector of* $a(s)$.

Proof: For the proof of this theorem we will consider two cases, as follows:

Case 1: $c = 0$. As proved in Theorem 4.2, $s = 0$ is the unique maximum point of $\frac{a(s)}{s}$, which is monotonically increasing for $s < 0$ and monotonically decreasing for $s > 0$. Furthermore, for $-r \le s \le r$ the following inequality can be written

$$\frac{\tanh s}{s} \ge \frac{\tanh r}{r} \tag{4.22}$$

which proves the theorem for the case when $c = 0$.

Case 2: $c > 0$ (The case $c < 0$ would be similar) and $c < r$. Consider the point $s = s_1$ as the point where the upper bound of the sector $a(s)$ occurs (defined in Theorem 4.2). In order to prove the theorem for this case, we define two separate regions. Denote M_1 as a region where $s \in [-r-c, s_1]$ and M_2 as a region where $s \in [s_1, r - c]$. We will

calculate the lower bound of the sector for the region M_1 and M_2 separately. Figure (4.8) illustrates the regions M_1 and M_2 as well as the critical point s_1.

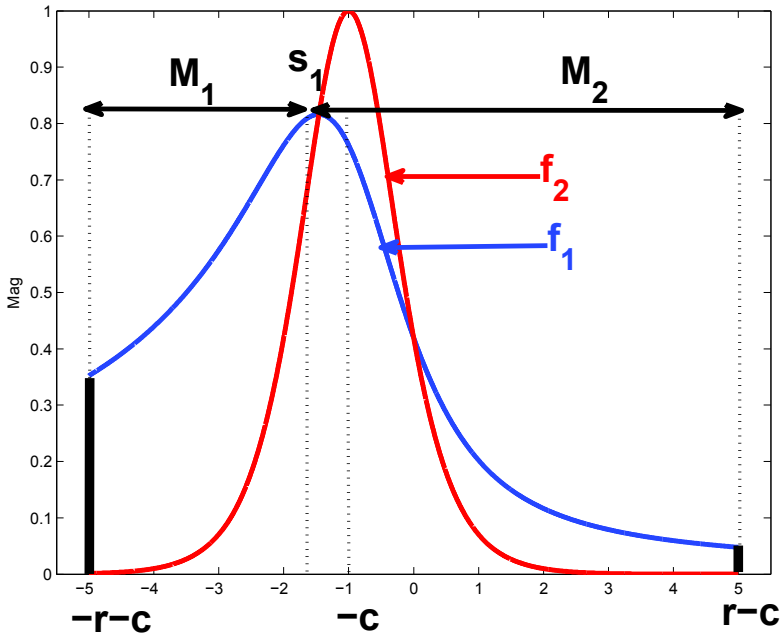

Figure (4.8) Lower sector bound of $a(s)$

In the region M_1, the function $\frac{a(s)}{s}$ is monotonically increasing. According to (4.18), $f_3(s) < 0$, and s_1 is the unique maximum point over the whole domain. Since $\frac{a(s)}{s}$ is monotonically increasing for $s \in [-r-c, s_1]$, the minimum of $\frac{a(s)}{s}$ occurs at the lower bound, which is $s = -r-c$. In other words,

$$\begin{aligned} \frac{\tanh(s+c)-\tanh(c)}{s} &\geq \frac{\tanh(-r-c+c)-\tanh(c)}{-r-c} \\ &\geq \frac{\tanh(r)+\tanh(c)}{r+c} \end{aligned} \tag{4.23}$$

Considering the results derived in Appendix A , Lemma A.1,

$$\frac{\tanh r + \tanh c}{r + c} \geq \frac{\tanh(r) - \tanh c}{r - c} \tag{4.24}$$

which proves the theorem for the interval $s \in [-r-c, s_1]$. In the region M_2, the function $f_3(s)$ becomes zero at $s = 0$, by (4.18). However, the derivative of $\frac{a(s)}{s}$ at this point is non-zero, by (4.13). This means that $\frac{a(s)}{s}$ is monotonically decreasing for the interval $s \in [s_1, r - c]$. Consequently, the minimum occurs at the lower bound, which is $s = r - c$. In other words,

$$\begin{aligned} \frac{\tanh(s + c) - \tanh c}{s} &\geq \frac{\tanh(r - c + c) - \tanh c}{r - c} \\ &\geq \frac{\tanh r - \tanh c}{r - c} \end{aligned} \tag{4.25}$$

This proves the theorem for the interval $s \in [s_1, r - c]$. Combining the result for M_1, (4.24), and the result for M_2, (4.24), the theorem is proven. □

In the following section, the quadratic forms based on the sector condition will be derived for the activation function given in (4.6).

Quadratic Form and Vector Case

In the previous section, we proved that (4.6) has unique lower bound and upper bound. In this section, we will use these lower and upper sector bounds to derive quadratic forms. For every neuron in the BPRNN we will derive a quadratic form, and we will use all these quadratic forms in Chapter 5 to derive the final stability criterion.

For every neuron in the BPRNN, the nonlinearity can be written as $a_i = f_i({}_i\mathbf{w}^T\mathbf{x})$. If f_i is assumed to be a tangent hyperbolic function, then by Theorem 4.2 and Theorem 4.3 a_i has finite sector bounds. Denote the lower and upper sector bound of a_i by l_i and u_i. Hence,

$$l_i \le \frac{a_i}{{}_i\mathbf{w}^T\mathbf{x}} \le u_i \quad i = 1, 2, \ldots, m \tag{4.26}$$

where m is the total number of neurons in the BPRNN, ${}_i\mathbf{w}^T\mathbf{x} \neq 0$ and $\mathbf{x} \in \mathbf{R}^n$. According to (4.4), the inequality (4.26) implies the following inequality,

$$(a_i - l_i\,{}_i\mathbf{w}^T\mathbf{x})(u_i\,{}_i\mathbf{w}^T\mathbf{x} - a_i) \ge 0 \tag{4.27}$$

Expanding the left hand side of the inequality (4.27) yields

$$\begin{gathered} a_i u_i\,{}_i\mathbf{w}^T\mathbf{x} - (a_i)^2 - \mathbf{x}^T\,{}_i\mathbf{w} l_i u_i\,{}_i\mathbf{w}^T\mathbf{x} + \mathbf{x}^T{}_i\mathbf{w} l_i a_i \ge 0 \\ -(a_i)^2 - \mathbf{x}^T({}_i\mathbf{w}^T l_i u_i\,{}_i\mathbf{w})\mathbf{x} + a_i(u_i\,{}_i\mathbf{w}^T)\mathbf{x} + \mathbf{x}^T(l_i\,{}_i\mathbf{w})a_i \ge 0 \\ \begin{bmatrix} a_i & \mathbf{x}^T \end{bmatrix} \begin{bmatrix} -1 & \dfrac{{}_i\mathbf{w}^T(u_i + l_i)}{2} \\ \dfrac{{}_i\mathbf{w}(u_i + l_i)}{2} & -l_i u_i\,{}_i\mathbf{w}_i\mathbf{w}^T \end{bmatrix} \begin{bmatrix} a_i \\ \mathbf{x} \end{bmatrix} \ge 0 \end{gathered} \tag{4.28}$$

The quadratic inequality derived in (4.28) is the quadratic form derived from the sector condition for the i^{th} neuron. Define $\mathbf{z} = \begin{bmatrix} \mathbf{a}^T & \mathbf{x}^T \end{bmatrix}^T$ where $\mathbf{a} = \begin{bmatrix} a_1, a_2, \ldots, a_m \end{bmatrix}^T \in \mathbf{R}^m$ and rewrite (4.28) in terms of $\mathbf{z}$:

$$q_i(\mathbf{z}) = \mathbf{z}^T \begin{bmatrix} {}_i\mathbf{T}_1 & {}_i\mathbf{T}_2 \\ {}_i\mathbf{T}_2^T & {}_i\mathbf{T}_3 \end{bmatrix} \mathbf{z} \geq 0 \tag{4.29}$$

In above equation, ${}_i\mathbf{T}_1 = diag(0, \ldots, 0, -1, 0, \ldots, 0)$ is an $m \times m$ block diagonal matrix (-1 is in the i^{th} diagonal entry), ${}_i\mathbf{T}_2 = \left[\mathbf{0}^T, \ldots, \mathbf{0}^T, \frac{{}_i\mathbf{w}^T(u_i + l_i)}{2}, \mathbf{0}^T, \ldots, \mathbf{0}^T \right]^T$ is an $m \times n$ block matrix ($\frac{{}_i\mathbf{w}^T(u_i + l_i)}{2}$ is a row vector located at the i^{th} row) and ${}_i\mathbf{T}_3 = -u_i l_i \, {}_i\mathbf{w}_i\mathbf{w}^T$ is an $n \times n$ block matrix.

The quadratic form in (4.29) will be used in Chapter 5 for the derivation of the final stability criterion.

Conclusion

The main contribution of this chapter was the derivation of sector conditions for a certain class of functions. Some important theorems were given that described the behavior of the sector width for zero biases and non-zero biases. We showed here that the bias affects the width of the sector. When the biases are zero, the sector has the maximum width. In the following chapter we will show that the width of the sector has a direct affect on stability. We will also show how the quadratic forms derived in (4.29) and the Lyapunov theorem can be used to derive the final criterion for stability of the Barabonov-Prokharov model given in (3.1).

5 ABSOLUTE STABILITY ANALYSIS

Stability analysis plays an important role in control system theory. There are different types of stability analysis that can be studied for different types of dynamical systems. The main focus of this chapter is devoted to the absolute stability analysis of a special kind of RNN.

In this chapter, the basic definition of stability for dynamical systems will be introduced. Moreover, the most famous definitions of stability will be reviewed briefly. In the second part, important stability theorems will be investigated. The final section is devoted to a derivation of sufficient conditions for the absolute stability of the BPRNN given in (3.1). These conditions will be derived by merging the sector condition derived in (4.29) and the Lyapunov theorem. The final criterion will be given in terms of a Linear Matrix Inequality (LMI). The feasibility of the solution to this LMI guarantees the absolute stabil-

ity of the equilibrium point of the BPRNN. The derived criterion will be verified through a numerical example.

Definitions of Stability

Let $X \subset \mathbf{R}^n$. A discrete dynamical system on X can be defined by a continuous function $\mathbf{f} : X \to X$. Such a system is represented by

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{5.1}$$

where $\mathbf{x}(k)$ is the state of the system at time step k. One main topic of interest in studying dynamical systems (5.1) is the behavior of the trajectories with respect to the equilibrium point, since this describes the long term behavior of the physical system that is being represented. The equilibrium point of a dynamical system is defined bellow.

Definition 5.1 *[Elya98] A point* $\mathbf{x}^*$ *is said to be an equilibrium point of the dynamical system (5.1) if* $\mathbf{x}^* = \mathbf{f}(\mathbf{x}^*)$.

In general, stability is defined for a specific equilibrium point. When we say a dynamical system (5.1) is stable, it means that it has only one equilibrium point and that equilibrium point is stable. In other words, system trajectories starting from arbitrary initial conditions will converge to the stable equilibrium point. The stability of the equilibrium point $\mathbf{x}^*$ for system (5.1) can be defined as follows:

Definition 5.2 *[Elya98] The equilibrium point* $\mathbf{x}^*$ *is said to be* ***stable*** *if* $\forall \varepsilon > 0$, $\exists \delta > 0$ *such that* $\|\mathbf{x}_0 - \mathbf{x}^*\| < \delta$ *implies* $\|\mathbf{x}_k - \mathbf{x}^*\| < \varepsilon$ *for all* $k \geq 0$. *This definition is also called stability* ***In the Sense of Lyapunov (ISL)****.*

Definition 5.2 becomes the definition of **uniform stability** if δ only depends on ε ($\delta = \delta(\varepsilon)$). The equilibrium point of dynamical system (5.1) is **unstable** if it is not stable.

The equilibrium point of system (5.1) can always be shifted to the origin, hence without loss of generality we will consider only the stability of the origin.

Another important technical term in the theory of stability analysis is *attraction.* The attracting equilibrium point of system (5.1) can be defined as follows:

Definition 5.3 *[Elya98] The equilibrium point* $\mathbf{x}^*$ *is* ***attracting*** *if* $\exists \eta > 0$ *such that* $\|\mathbf{x}_0 - \mathbf{x}^*\| < \eta$ *implies* $\lim_{k \to \infty} \mathbf{x}_k = \mathbf{x}^*$. *If* $\lim_{k \to \infty} \mathbf{x}_k = \mathbf{x}^*$ *for all* $\mathbf{x}_0$, *then* $\mathbf{x}^*$ *is called* ***globally attracting***.

The other type of stability for the equilibrium point of (5.1) can be defined by combining Definition 5.2 and Definition 5.3.

Definition 5.4 *[Elya98] The equilibrium point* $\mathbf{x}^*$ *is* ***asymptotically stable*** *if it is stable and attracting. The equilibrium point* $\mathbf{x}^*$ *is* ***Globally Asymptotically Stable (GAS)*** *if it is stable and globally attracting.*

If the origin of system (5.1) is uniformly stable and attracting then the origin is **uniformly asymptotically stable**. GAS is the strongest type of stability, and it is usually an objective in the design of control systems. If a dynamical system has a GAS equilibrium point, then system trajectories starting from arbitrary initial conditions converge to the same equilibrium point. In the Chapter 6 and Chapter 7, we will develop stability criteria that will be used in Chapter 9 to design a system with RNN controller and RNN emulator.

By Definition 5.4, asymptotic stability implies stability whereas the opposite implication is not true. An example of a system that illustrates this, is a ball rolling inside a cup

without friction. The graphical representation of this system is shown in Figure (5.1). In this example, the ball oscillates between points A and B and never converges to the equilibrium point C. By Definition 5.2, the point C is stable ISL, but it is not asymptotically stable, because it is not an attracting point.

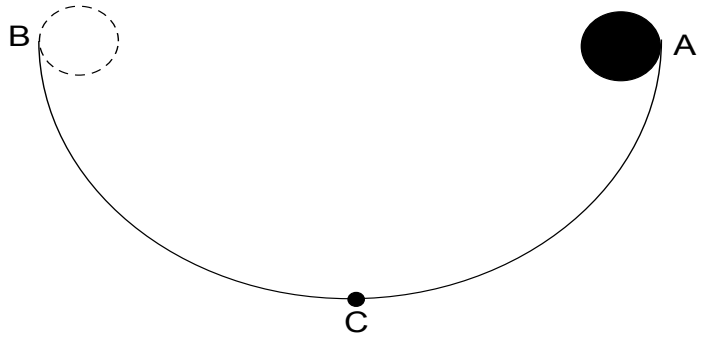

Figure (5.1) Stable Equilibrium point but not Asymptotically stable

The distinction between a globally attracting point and stable point is subtle. The following example shows a dynamical system with a globally attracting equilibrium point that is not stable.

Example 5.1 *[Elya98] Consider a dynamical system defined on the unit circle as*

$$\theta(k+1) = \sqrt{2\pi\theta(k)},\ \forall\theta(k) \in [0, 2\pi) \tag{5.2}$$

The solution of the above difference equation is

$$\theta(k) = (2\pi)^{1-2^{-k}}\theta(0)^{2^{-k}} \tag{5.3}$$

$\forall k \geq 0$. *The equilibrium point of system (5.2) can be derived by finding the limit of (5.3) as* $k \to \infty$. *It turns out that the point* $\theta^* = 0$ *is the equilibrium point. Since* $\theta(k+1) > \theta(k)$, *any trajectories that start on the unit circle (excluding the equilibrium point) converge to the equilibrium point in the counter clockwise direction. The point*

$\theta^* = 0$ *is not stable because* $\exists \varepsilon > 0$ *such that* $\forall \delta > 0$ *there is some* $\theta(0)$ *in the* δ*-neighborhood of* θ^* *such that* $\theta(k)$ *leaves the* ε*-neighborhood. This is the negation of Definition 5.2. However, by Definition 5.3 the equilibrium point is globally attracting because all trajectories that start on the unit circle converge to the equilibrium point. The behavior of system (5.2) for initial conditions* A, B *and* C *is illustrated in Figure (5.2). For more detail about the characteristics of systems with globally attracting points see [Jafar11]*

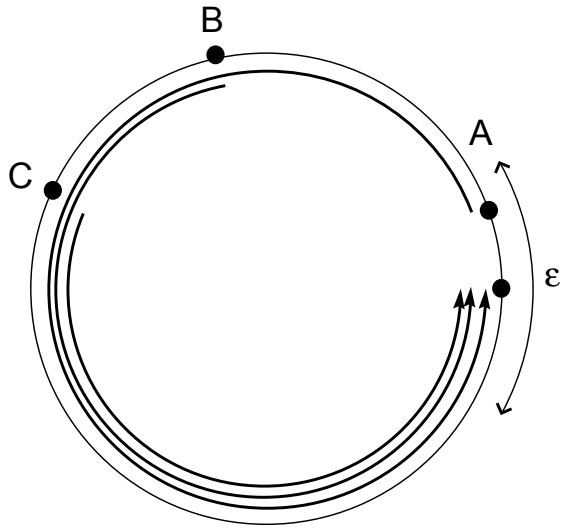

Figure (5.2) Globally Attracting but not Stable

Next, we will define absolute stability. This definition will be used later in this chapter to derive a sufficient condition for the absolute stability of BPRNNs.

Consider system (3.6), with block diagram given in Figure (3.2), which satisfies the sector condition given in (4.1). The problem of interest in absolute stability is to study the stability of the origin, not for one particular nonlinearity, but rather for a special class of nonlinearity. Considering this, we will define absolute stability as follows:

Definition 5.5 *[Khal01]: Consider the system* (3.6), *where* $\mathbf{f}$ *satisfies a sector condition (4.1). The system is* ***absolutely stable*** *if the origin is asymptotically stable for any nonlinearity in the given sector. It is absolutely stable with a finite domain if the origin is* ***uniformly asymptotically stable****.*

In order to check the absolute stability of the equilibrium point of a system, the system needs to be put in a special form. This form should consist of a linear part in the forward direction and a nonlinear part in the feedback direction. Moreover, the nonlinearity needs to satisfy a sector condition as well. The form is illustrated in Figure (5.3). $\mathbf{G}(s)$ represents the linear part, and $\mathbf{a}(.)$ represents the nonlinearity.

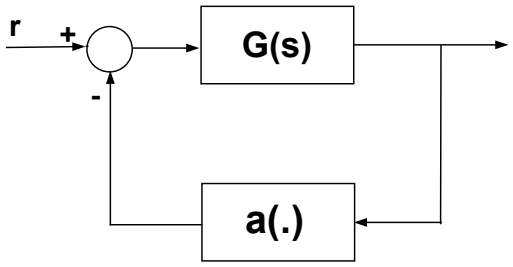

Figure (5.3) Suitable Form for Absolute Stability Analysis

The transformation of the BPRNN given in (3.1) to its equivalent model (3.6) in Chapter 3, and the demonstration that $\mathbf{f}$ satisfies sector conditions in Chapter 4, laid the groundwork for the derivation of the absolute stability criterion in this chapter. Now that the problem is represented in a suitable form, we will use the Lyapunov theorem and the S-procedure to derive a criterion for absolute stability of the origin of BPRNNs.

Stability Theorems

There are several different theorems available in the literature for stability analysis of closed-loop dynamical systems, i.e. the Lyapunov theorem, LaSalle's theorem, the Popov criterion (frequency approach), the Circle criterion (frequency approach), etc. In this section, we will use the main stability theorem, the Lyapunov theorem, to derive an absolute stability criterion for the BPRNN given in (3.6).

Lyapunov Stability Theorem

The Lyapunov stability theorem is the most famous theorem in stability analysis. The direct Lyapunov theorem states that if a Lyapunov function exists, then the equilibrium point of a system is either stable or asymptotically stable. The converse Lyapunov theorem states that if the equilibrium point of a dynamical system is stable ISL or GAS, then there exists a Lyapunov function. (Theorem 6.5 will be proved by the converse theorem.) The main difficulty with the Lyapunov theorems is the fact that they are not constructive, and finding a Lyapunov function could be challenging. (For the proof of direct and converse Lyapunov theorem see [Jafar11].) In Chapter 7, we will introduce a method which can numerically approximate a Lyapunov level surface (the region where the Lyapunov function is constant).

Since the main goal in this section is to derive sufficient conditions for the stability of system (3.6), we will use the direct Lyapunov theorem. Before stating the direct Lyapunov theorem, we will introduce the following definitions:

Definition 5.6 [Jafar11] *Let* $\mathbf{x}^* \in X \subset \mathbf{R}^n$ *and* V *be a function on* X *with values in* $\mathbf{R}$. *Then we will say that* V *satisfies the* ***annulus condition*** *with respect to* $\mathbf{x}^*$ *if*

1. $V(\mathbf{x}^*) = 0$,

2. *There is a number* $\alpha > 0$ *such that if* $0 < \delta < \alpha$ *then*

$$inf\{ V(\mathbf{x}) \mid \mathbf{x} \in X, \delta \le \|\mathbf{x} - \mathbf{x}^*\| \le \alpha \} > 0 . \tag{5.4}$$

Definition 5.7 [Jafar11] *We say that real-valued function* V *is positive definite at* $\mathbf{x}^*$ *if*

1. $V(\mathbf{x}^*) = 0$,

2. $V(\mathbf{x}) > 0$ *for all* $\mathbf{x} \in \mathbf{B}(\mathbf{x}^*, \alpha), \mathbf{x} \neq \mathbf{x}^*$ *for some* $\alpha > 0$.

where in the Definition 5.7, $\mathbf{B}(\mathbf{x}^*, \alpha)$ is a ball centered at $\mathbf{x}^*$ with radius α. Note that the annulus condition says that V has a positive lower bound at $\mathbf{x} \in X$ lying on the annulus. If V satisfies the annlus condition with respect to $\mathbf{x}^*$, then V is positive definite at $\mathbf{x}^*$. The opposite direction is not necessary true, unless V is continuous and X is closed.

Using Definition 5.6 and Definition 5.7, the direct Lyapunov theorem can be stated as follows:

Theorem 5.1 [Jafar11] *(Direct Lyapunov Theorem) Let* $X \subset \mathbf{R}^n$ *be a closed set for system* (5.1) *with an equilibrium point* $\mathbf{x}^* \in X$. *Suppose that there is a function* $V : X \to [0, \infty)$ *such that*

1. V *satisfies annlus condition with respect to* $\mathbf{x}^*$,

2. For all $\mathbf{x} \in X - \{\mathbf{x}^*\}$, $V(\mathbf{f}(\mathbf{x})) < V(\mathbf{x})$,

3. V is continuous,

4. For all $C \in [0, \infty)$, the set $\{\mathbf{z} \in X \mid V((\mathbf{z}) \le C)\}$ is bounded,

then $\mathbf{x}^*$ *is GAS.*

In Theorem 5.1 if for all $\mathbf{x} \in X - \{\mathbf{x}^*\}$, $V(\mathbf{f}(\mathbf{x})) \leq V(\mathbf{x})$ then $\mathbf{x}^*$ is stable. For the proof of Theorem 5.1 see [Jafar11].

There are many systems for which the equilibrium point is GAS, but for which it is not easy to derive a Lyapunov function. It is common to restrict considerations to Lyapunov functions from some multi-parametric class (i.e a class of quadratic forms). With such restrictions, the problem of obtaining effective necessary and sufficient conditions for the existence of a Lyapunov function become solvable. This is because it turns into a pure algebraical problem. Although a solution of such problems leads only to sufficient conditions for stability, these sufficient conditions are usually general enough, because they embrace all the conditions that can be obtained by using Lyapunov functions from the class selected. The wider the class of Lyapunov functions, the more complicated the corresponding conditions, and the more information about the system they require. Therefore it may be unreasonable to take too wide a class of Lyapunov functions [YaLe04].

As proved in the previous chapter, the original system (3.1) and the transformed system (3.6) are equivalent in terms of stability. Consider the transformed system, and choose a candidate Lyapunov function to be $V(\mathbf{x}(k)) = \mathbf{x}^T(k)\mathbf{H}\mathbf{x}(k)$, where $\mathbf{H} = \mathbf{H}^T \in \mathbf{R}^{n \times n}$ is a positive definite matrix. Then by Theorem 5.1, and without loss of generality, the origin of system (3.6) is GAS if there exists $\mathbf{H} = \mathbf{H}^T > 0$ such that

$$\mathbf{x}^T(k+1)\mathbf{H}\mathbf{x}(k+1) - \mathbf{x}^T(k)\mathbf{H}\mathbf{x}(k) < 0 \tag{5.5}$$

By substituting $\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{a}(k)$ into inequality (5.5) and removing k (time in-

dex) for simplicity, we have

$$(\mathbf{Ax} + \mathbf{Ba})^T \mathbf{H}(\mathbf{Ax} + \mathbf{Ba}) - \mathbf{x}^T \mathbf{Hx} < 0 \tag{5.6}$$

Expanding the above inequality yields

$$\begin{aligned} &\mathbf{x}^T \mathbf{A}^T \mathbf{HAx} + \mathbf{x}^T \mathbf{A}^T \mathbf{HBa} + \mathbf{a}^T \mathbf{B}^T \mathbf{HAx} + \mathbf{a}^T \mathbf{B}^T \mathbf{HBa} - \mathbf{x}^T \mathbf{Hx} < 0 \\ &\mathbf{a}^T (\mathbf{B}^T \mathbf{HB})\mathbf{a} + \mathbf{x}^T (\mathbf{A}^T \mathbf{HA} - \mathbf{H})\mathbf{x} + 2\mathbf{a}^T (\mathbf{B}^T \mathbf{HA})\mathbf{x} < 0 \\ &\begin{bmatrix} \mathbf{a}^T & \mathbf{x}^T \end{bmatrix} \begin{bmatrix} \mathbf{B}^T \mathbf{HB} & \mathbf{B}^T \mathbf{HA} \\ \mathbf{A}^T \mathbf{HB} & \mathbf{A}^T \mathbf{HA} - \mathbf{H} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{x} \end{bmatrix} < 0 \end{aligned} \tag{5.7}$$

Denote the above inequality as $q_0(\mathbf{z})$, and rewrite it in terms of the variable $\mathbf{z}$, as follows:

$$q_0(\mathbf{z}) = \mathbf{z}^T \mathbf{T}_0 \mathbf{z} < 0 \tag{5.8}$$

$\mathbf{T}_0 = \begin{bmatrix} \mathbf{B}^T \mathbf{HB} & \mathbf{B}^T \mathbf{HA} \\ \mathbf{A}^T \mathbf{HB} & \mathbf{A}^T \mathbf{HA} - \mathbf{H} \end{bmatrix}$ where $\mathbf{A} \in \mathbf{R}^{n \times n}$, $\mathbf{B} \in \mathbf{R}^{n \times m}$.

If there exists $\mathbf{H} = \mathbf{H}^T > 0$ such that $q_0(\mathbf{z}) < 0$, then $\mathbf{x}^T(k)\mathbf{Hx}(k)$ is a Lyapunov function, and the origin of system (3.6) is GAS. These are the most general results that can be obtained for the convergence of all the solutions of system (3.6). This is because the condition was derived without considering any constraint on the nonlinear function. Unfortunately, inequality (5.6) does not always have a feasible solution for $\mathbf{H}$, in which case nothing can be said about the stability of system (3.6). Infeasibility of inequality (5.6) does not confirm instability, because we are searching for Lyapunov functions only inside the quadratic class of functions. However, the equilibrium point might be demonstrated as GAS with a non-quadratic Lyapunov function.

The nonlinear function $\mathbf{a}(k)$ in (5.6) belongs to a class of functions that satisfies the sector condition derived in (4.29). Considering the fact that $\mathbf{a}(k)$ belongs to special class of function, makes the search for a Lyapunov function easier. Yakubovich *et al* [YaLe04] proved theoretically that consideration of the sector conditions changes inequality (5.6) in such a way that the inequality becomes easier to solve.

The sector condition in quadratic form (4.29) is derived for single neurons of a BPRNN. Considering all the neurons in (3.6), we will have numbers of single inequalities. In the following, a special technique will be introduced to merge several inequalities into a single inequality.

S-procedure [BoFe94]

A special method, called the S-procedure, is used in a large number of nonlinear control problems. The S-procedure method simply gives us a method for converting a set of inequalities into single inequality.

Let $q_1(\mathbf{z}) \geq 0$, $q_2(\mathbf{z}) \geq 0$, …, $q_m(\mathbf{z}) \geq 0$ be the sector conditions in quadratic form derived for every neuron of the BPRNN and $-q_0(\mathbf{z}) > 0$ be the Lyapunov inequality derived in (5.8). The sector condition inequalities and the Lyapunov inequality need to be true at the same time. By the S-procedure method, these inequalities can be converted to a single inequality if there exists $\tau_1 \geq 0$, $\tau_2 \geq 0$, …, $\tau_m \geq 0$ such that

$$S(\boldsymbol{\tau}, \mathbf{z}) = -q_0(\mathbf{z}) - \tau_1 q_1(\mathbf{z}) - \tau_2 q_2(\mathbf{z}) - \ldots - \tau_m q_m(\mathbf{z}) > 0 \tag{5.9}$$

where $\mathbf{z} \in \mathbf{R}^{n+m}$ and $\tau = \begin{bmatrix} \tau_1 & \tau_2 & \ldots & \tau_m \end{bmatrix}$. The S-procedure method only gives a sufficient condition for converting a set of inequalities into a single inequality. Hence any further cri-

terion which is derived based on this method will only be a sufficient condition for stability. In the next section, we will use the S-procedure method to derive a criterion for absolute stability of BPRNNs.

Absolute Stability Criterion for BPRNNs

In order to derive the absolute stability criterion for system (3.6), the Lyapunov inequality (5.8) and all sector conditions for all neurons of the BPRRN (4.29) need to be satisfied at the same time. In other words,

$$q_0(\mathbf{z}) < 0 \text{ for all } q_i(\mathbf{z}) \geq 0 \quad 1 \leq i \leq m \tag{5.10}$$

Since $q_0(\mathbf{z})$ and $q_i(\mathbf{z})$ are both quadratic functions of $\mathbf{z}$, by applying the S-procedure method, the $m+1$ inequalities given in (5.10) can be converted into a single inequality if there exists $\tau_1 \geq 0, \tau_2 \geq 0, \ldots, \tau_m \geq 0$ such that

$$-q_0(\mathbf{z}) - \sum_{i=1}^{m} \tau_i q_i(\mathbf{z}) > 0 \tag{5.11}$$

By substituting $q_0(\mathbf{z})$ and $q_i(\mathbf{z})$ from (5.8) and (4.29) into (5.11) and multiplying -1 times both sides of inequality (5.11), we will get the following inequality:

$$\mathbf{z}^T \begin{bmatrix} \mathbf{B}^T\mathbf{H}\mathbf{B} & \mathbf{B}^T\mathbf{H}\mathbf{A} \\ \mathbf{A}^T\mathbf{H}\mathbf{B} & \mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H} \end{bmatrix} \mathbf{z} + \tau_1 \mathbf{z}^T \begin{bmatrix} {}_1\mathbf{T}_1 & {}_1\mathbf{T}_2 \\ {}_1\mathbf{T}_2^T & {}_1\mathbf{T}_3 \end{bmatrix} \mathbf{z} + \ldots + \tau_m \mathbf{z}^T \begin{bmatrix} {}_m\mathbf{T}_1 & {}_s\mathbf{T}_2 \\ {}_m\mathbf{T}_2^T & {}_s\mathbf{T}_3 \end{bmatrix} \mathbf{z} < 0$$

$$\mathbf{z}^T \begin{bmatrix} \mathbf{B}^T\mathbf{H}\mathbf{B} & \mathbf{B}^T\mathbf{H}\mathbf{A} \\ \mathbf{A}^T\mathbf{H}\mathbf{B} & \mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H} \end{bmatrix} \mathbf{z} + \mathbf{z}^T \begin{bmatrix} \tau_1({}_1\mathbf{T}_1) + \ldots + \tau_s({}_s\mathbf{T}_1) & \tau_1({}_1\mathbf{T}_2) + \ldots + \tau_m({}_m\mathbf{T}_2) \\ \tau_1({}_1\mathbf{T}_2^T) + \ldots + \tau_s({}_s\mathbf{T}_2^T) & \tau_1({}_1\mathbf{T}_3) + \ldots + \tau_m({}_m\mathbf{T}_3) \end{bmatrix} \mathbf{z} < 0 \tag{5.12}$$

Let $\mathbf{M}$ be the matrix of lower bounds, $\mathbf{M} = diag(l_1, l_2, \ldots, l_m) \in \mathbf{R}^{m \times m}$, let $\mathbf{N}$ be the matrix of upper bounds, $\mathbf{N} = diag(u_1, u_2, \ldots, u_m) \in \mathbf{R}^{m \times m}$, and let $\Gamma = diag(\tau_1, \ldots, \tau_m) \in \mathbf{R}^{m \times m}$. Then (5.12) may be rewritten as

$$\mathbf{z}^T \begin{bmatrix} \mathbf{B}^T\mathbf{H}\mathbf{B} & \mathbf{B}^T\mathbf{H}\mathbf{A} \\ \mathbf{A}^T\mathbf{H}\mathbf{B} & \mathbf{A}^T\mathbf{H}\mathbf{A} - \mathbf{H} \end{bmatrix} \mathbf{z} + \mathbf{z}^T \begin{bmatrix} & -\Gamma & & \frac{{}_1\mathbf{w}^T(u_1 + l_1)\tau_1}{2} \\ & & & \cdots \\ & & & \frac{{}_s\mathbf{w}^T(u_s + l_m)\tau_m}{2} \\ \frac{{}_1\mathbf{w}(u_1 + l_1)\tau_1}{2} & \cdots & \frac{{}_s\mathbf{w}(u_s + l_m)\tau_m}{2} & -\sum_{1 \le i \le m} \tau_i l_i u_i \; {}_i\mathbf{w}_i\mathbf{w}^T \end{bmatrix} \mathbf{z} < 0 \tag{5.13}$$

The second matrix in the above inequality contains block matrices that could be rewritten in a simpler form considering

$$\begin{bmatrix} \frac{{}_1\mathbf{w}^T(u_1 + l_1)\tau_1}{2} \\ \frac{{}_2\mathbf{w}^T(u_2 + l_2)\tau_2}{2} \\ \cdots \\ \frac{{}_s\mathbf{w}^T(u_m + l_m)\tau_m}{2} \end{bmatrix} = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{\Gamma}(\mathbf{M} + \mathbf{N})\mathbf{W} \tag{5.14}$$

$$\begin{bmatrix} \frac{{}_1\mathbf{w}(u_1 + l_1)\tau_1}{2} & \frac{{}_2\mathbf{w}(u_2 + l_2)\tau_2}{2} & \cdots & \frac{{}_m\mathbf{w}(u_m + l_m)\tau_m}{2} \end{bmatrix} = \frac{\mathbf{1}}{\mathbf{2}}\mathbf{W}^T(\mathbf{M} + \mathbf{N})\mathbf{\Gamma} \tag{5.15}$$

$$\sum_{1 \le i \le m} \tau_i l_i u_i \; {}_i\mathbf{w}_i\mathbf{w}^T = \mathbf{W}^T\mathbf{M}\mathbf{\Gamma}\mathbf{N}\mathbf{W} \tag{5.16}$$

In consideration of (5.14), (5.15) and (5.16), the inequality (5.13) may be rewritten as follows

$$\mathbf{z}^T\begin{bmatrix}\mathbf{B}^T\mathbf{H}\mathbf{B} & \mathbf{B}^T\mathbf{H}\mathbf{A}\\ \mathbf{A}^T\mathbf{H}\mathbf{B} & \mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H}\end{bmatrix}\mathbf{z}+\mathbf{z}^T\begin{bmatrix}-\boldsymbol{\Gamma} & \frac{1}{2}\boldsymbol{\Gamma}(\mathbf{M}+\mathbf{N})\mathbf{W}\\ \frac{1}{2}\mathbf{W}^T(\mathbf{M}+\mathbf{N})\boldsymbol{\Gamma} & -\mathbf{W}^T\mathbf{M}\boldsymbol{\Gamma}\mathbf{N}\mathbf{W}\end{bmatrix}\mathbf{z}<0$$

$$\mathbf{z}^T\begin{bmatrix}\mathbf{B}^T\mathbf{H}\mathbf{B}-\boldsymbol{\Gamma} & \mathbf{B}^T\mathbf{H}\mathbf{A}+\frac{1}{2}\boldsymbol{\Gamma}(\mathbf{M}+\mathbf{N})\mathbf{W}\\ \mathbf{A}^T\mathbf{H}\mathbf{B}+\frac{1}{2}\mathbf{W}^T(\mathbf{M}+\mathbf{N})\boldsymbol{\Gamma} & \mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H}-\mathbf{W}^T\mathbf{M}\boldsymbol{\Gamma}\mathbf{N}\mathbf{W}\end{bmatrix}\mathbf{z}<0 \tag{5.17}$$

Substituting $\mathbf{z} = \begin{bmatrix}\mathbf{a} & \mathbf{x}\end{bmatrix}^T$ in above inequality yields

$$\begin{bmatrix}\mathbf{a} & \mathbf{x}\end{bmatrix}^T\begin{bmatrix}\mathbf{B}^T\mathbf{H}\mathbf{B}-\boldsymbol{\Gamma} & \mathbf{B}^T\mathbf{H}\mathbf{A}+\frac{1}{2}\boldsymbol{\Gamma}(\mathbf{M}+\mathbf{N})\mathbf{W}\\ \mathbf{A}^T\mathbf{H}\mathbf{B}+\frac{1}{2}\mathbf{W}^T(\mathbf{M}+\mathbf{N})\boldsymbol{\Gamma} & \mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H}-\mathbf{W}^T\mathbf{M}\boldsymbol{\Gamma}\mathbf{N}\mathbf{W}\end{bmatrix}\begin{bmatrix}\mathbf{a}\\ \mathbf{x}\end{bmatrix}<0 \tag{5.18}$$

Expanding the above inequality will produce the following:

$$\mathbf{a}^T(\mathbf{B}^T\mathbf{H}\mathbf{B}-\boldsymbol{\Gamma})\mathbf{a}+\mathbf{x}^T(\mathbf{A}^T\mathbf{H}\mathbf{A}-\mathbf{H}-\mathbf{W}^T\mathbf{M}\boldsymbol{\Gamma}\mathbf{N}\mathbf{W})\mathbf{x}+\mathbf{a}^T\left(\mathbf{B}^T\mathbf{H}\mathbf{A}+\frac{1}{2}\boldsymbol{\Gamma}(\mathbf{M}+\mathbf{N})\mathbf{W}\right)\mathbf{x}$$
$$+\mathbf{x}^T\left(\mathbf{A}^T\mathbf{H}\mathbf{B}+\frac{1}{2}\mathbf{W}^T(\mathbf{M}+\mathbf{N})\boldsymbol{\Gamma}\right)\mathbf{a}<0 \tag{5.19}$$

A further expansion produces

$$\begin{aligned}&\mathbf{x}^T(\mathbf{A}^T\mathbf{H}\mathbf{A})\mathbf{x}+\mathbf{x}^T(\mathbf{A}^T\mathbf{H}\mathbf{B})\ \mathbf{a}+\mathbf{a}^T(\mathbf{B}^T\mathbf{H}\mathbf{A})\mathbf{x}+\mathbf{a}^T(\mathbf{B}^T\mathbf{H}\mathbf{B})\mathbf{a}-\mathbf{x}^T(\mathbf{H})\mathbf{x}\\ &-\mathbf{a}^T\boldsymbol{\Gamma}\mathbf{a}-(\mathbf{x}^T\mathbf{W}^T)\mathbf{M}\boldsymbol{\Gamma}\mathbf{N}(\mathbf{W}\mathbf{x})+\mathbf{a}^T\mathbf{B}^T\mathbf{H}\mathbf{A}+\mathbf{a}^T\boldsymbol{\Gamma}\mathbf{N}(\mathbf{W}\mathbf{x})+(\mathbf{x}^T\mathbf{W}^T)\mathbf{M}\boldsymbol{\Gamma}\mathbf{a}<0\end{aligned} \tag{5.20}$$

Finally, the above inequality can be rearranged as

$$(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{a})^T\mathbf{H}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{a})-\mathbf{x}^T\mathbf{H}\mathbf{x}+(\mathbf{a}-\mathbf{M}\mathbf{W}\mathbf{x})^T\boldsymbol{\Gamma}(\mathbf{N}\mathbf{W}\mathbf{x}-\mathbf{a})<0 \tag{5.21}$$

Inequality (5.21) is linear with variable $\mathbf{H}$ and Γ. Since it is an inequality in matrix form, it is called a Linear Matrix Inequality (LMI). If this inequality has a feasible solution for $\mathbf{H} > 0$ and $\Gamma \geq 0$ (or better if $\Gamma \geq I$), then the equilibrium point of (3.6) or, equivalently, (3.1) will be absolutely stable. (The origin is globally assymtotrically stable for any nonlinearity in the given sector.) If inequality (5.21) does not have feasible solution for $\mathbf{H} > 0$ and $\Gamma \geq 0$, then nothing can be said about the stability of the system. However, a better stability criterion may be needed to verify the stability of the system.

The LMI derived in (5.21) has more terms than the LMI derived in (5.6). The additional terms are due to the consideration of the sector conditions. Yakubovich *et al* [YaLe04] proved that the inequality (5.21) is more likely to have a feasible solution than (5.6).

The effect of biases in the BPRNN can be seen in (5.21). Consider the case I when $\mathbf{b} = 0$ and the case II when $\mathbf{b} \neq 0$. We proved in Chapter 4 that for case I the sector has the longest length, with $\mathbf{M} = \mathbf{0}$ and $\mathbf{N} = \mathbf{I}$ ($\mathbf{I}$ represents the identity matrix). In case II, $\mathbf{M} > 0$ and $\mathbf{N} < \mathbf{I}$. It can observed that (5.21) is more difficult to satisfy in case I than in case II. Finally, we can conclude that the consideration of the biases makes the stability criterion less conservative.

In the following example, we will demonstrate the derived criterion in (5.21) through a numerical example.

Example 5.2 *Consider Example 3.1 in Chapter 3. The BPRNN given in (3.8) where*

$$\mathbf{LW}^{1,1} = \begin{bmatrix} -0.42 & -0.23 \\ 0.32 & 0.38 \end{bmatrix}, \mathbf{LW}^{1,2} = \begin{bmatrix} -0.62 & -1.01 & -1.17 \\ 0.03 & 0.79 & -0.70 \end{bmatrix}, \mathbf{LW}^{2,1} = \begin{bmatrix} -2.42 & 0.13 \\ 0.26 & -2.41 \\ 2.27 & 0.83 \end{bmatrix},$$

$$\mathbf{LW}^{2,2} = \begin{bmatrix} -0.4 & 0.3 & 0.01 \\ -0.48 & -0.53 & 0.57 \\ 0.16 & 0.52 & 0.48 \end{bmatrix} \mathbf{b}^1 = \begin{bmatrix} -1.66 \\ 1.66 \end{bmatrix} \mathbf{b}^2 = \begin{bmatrix} 2.42 \\ 0 \\ 2.42 \end{bmatrix}$$ *and the extended BPRNN is*

given in (3.9) where $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I}_{2\times 2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times 3} & \mathbf{0} \end{bmatrix} \mathbf{B} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{3\times 3} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$

$$\mathbf{W} = \begin{bmatrix} \mathbf{0} & \mathbf{LW}^{1,1} & \mathbf{LW}^{1,2} & \mathbf{0} \\ \mathbf{LW}^{2,1} & \mathbf{0} & \mathbf{0} & \mathbf{LW}^{2,2} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \end{bmatrix},$$

$\mathbf{M} = diag(0.005, 0.063, 0.001, 0.12, 0.02)$ *and* $\mathbf{N} = diag(0.51, 0.77, 0.28, 0.91, 0.75)$

The LMI given in (5.21), solved by the Matlab LMI toolbox, has a feasible solution for this example where

$$\mathbf{H} = 10^3 \times \begin{bmatrix} 8.91 & 0.83 & -0.02 & -0.04 & -0.01 & -0.17 & -0.08 & 0.56 & 1.53 & 2.42 \\ 0.8 & 7.1 & 0.04 & -0.02 & -0.04 & -0.17 & -0.37 & 0.63 & -0.1 & 0.37 \\ -0.02 & 0.04 & 2.4 & 0.4 & 1.63 & 0.43 & -0.8 & -0.02 & -0.07 & -0.03 \\ -0.04 & -0.02 & 0.4 & 3.06 & 0.08 & 1.92 & -0.72 & -0.01 & -0.05 & -0.11 \\ -0.01 & -0.04 & 1.63 & 0.08 & 3.7 & 0.74 & 0.32 & 0.01 & 0.03 & -0.02 \\ -0.1 & -0.17 & 0.43 & 1.92 & 0.74 & 3.63 & -0.01 & -0.05 & -0.01 & -0.03 \\ -0.08 & -0.37 & -0.8 & -0.72 & 0.32 & -0.01 & 4.72 & -0.01 & -0.01 & -0.07 \\ 0.56 & 0.63 & -0.02 & -0.01 & 0.01 & -0.05 & -0.01 & 1.66 & 0.36 & 0.24 \\ 1.53 & -0.1 & -0.07 & -0.05 & 0.03 & -0.01 & -0.01 & 0.36 & 1.37 & 0.27 \\ 2.42 & 0.37 & -0.03 & -0.11 & -0.02 & -0.03 & -0.07 & 0.24 & 0.27 & 2.28 \end{bmatrix} \quad (5.22)$$

$$\Gamma = 10^3 \times \begin{bmatrix} 2.29 & 0 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 2.59 & 0 & 0 \\ 0 & 0 & 0 & 0.11 & 0 \\ 0 & 0 & 0 & 0 & 0.40 \end{bmatrix} \qquad (5.23)$$

Since there exists $\mathbf{H}$ *and* Γ *both positive definite such that (5.21) is negative definite, the equilibrium point of the system must be GAS. The BPRNN response to a random initial condition is illustrated in Figure (5.4). The figure demonstrates the stability of the equilibrium point. Figure (5.5) shows the response of the equivalent system under an affine transformation to make the origin the equilibrium point.*

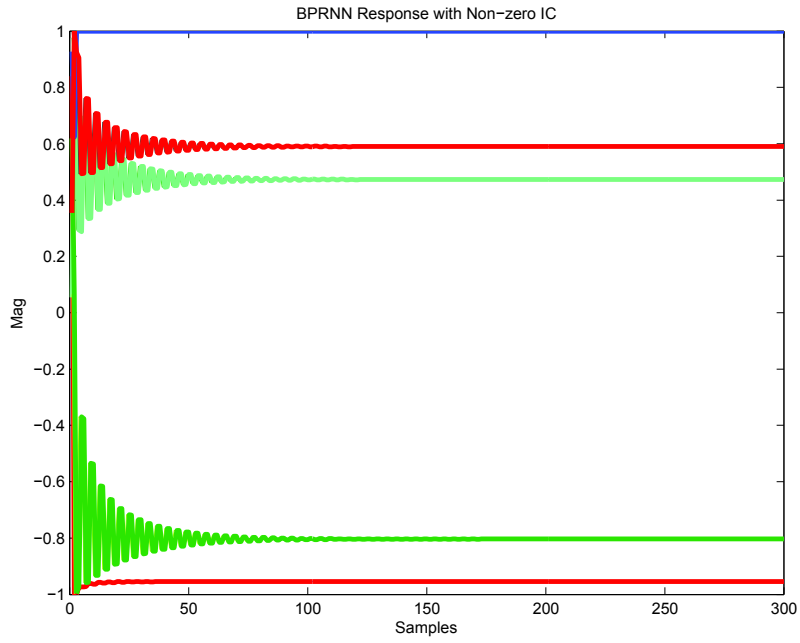

Figure (5.4) Absolute Stability of a BPRNN with non-zero Equilibrium point

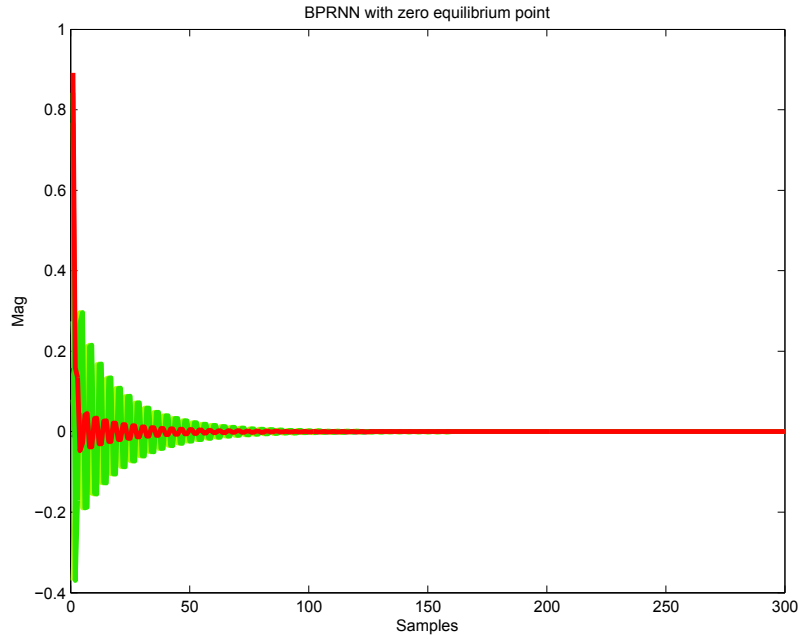

Figure (5.5) Absolute Stability of BPRNN with zero Equilibrium point

Conclusion

The chapter was devoted to the derivation of an absolute stability criterion for the BPRNN given in (3.1). The model of the BPRNN was introduced in Chapter 3, and the sector conditions were derived in Chapter 4. In this chapter, the sector conditions and the Lyapunov conditions were merged together to produce the final criterion for the absolute stability of the BPRNN.

The theory of absolute stability is a useful technique for the derivation of a criterion that guarantees global asymptotic stability of the equilibrium point for systems satisfying a sector condition, but this technique does not give a complete solution to the stability question. There are many systems with a globally asymptotically stable equilibrium point where (5.21) does not have a feasible solution. This is because the method considers only the quadratic class of Lyapunov functions.

In the next two chapters, we are going to introduce another stability technique, which will give a more complete answer to the stability question.

6 STABILITY ANALYSIS USING REDUCTION OF DISSIPATIVITY DOMAIN

The Reduction Of Dissipativity Domain (RODD) method is one of the most practical techniques for stability analysis of dynamical systems. The reason why this method is more practical than other stability analysis methods is because the stability criterion derived by this method is less conservative (restrictive) compared to many existing stability analysis methods. The conservativeness of a stability criterion can be defined as follows: Denote M as a *space of stability* for a specific RNN architecture, and let it be defined as the set of all parameter values for which a given RNN is stable. The best stability analysis method would be the one that can identify the largest possible subset of M. Except in spe-

cial cases, the exact determination of the full set M is not possible. We will say that a stability criterion is conservative to the extent that is does not identify the full set M. For example stability criteria developed by the Contraction Mapping Theorem (CMT) method (explained in the next section) are believed to be more conservative than stability criteria developed by the RODD method, because several stable systems have been found that the RODD method can demonstrate are stable, but CMT can not.

We divided the RODD method into two categories: RODD-LB and RODD-EB. RODD-LB is based on linear approximation of reachable sets (will be mainly explained in this chapter) and RODD-EB is based on quadratic approximation of reachable sets (will be explained in chapter 7). The RODD-LB method can be divided into two methods, RODD-LB1 and RODD-LB2. Both versions are based on linear boundaries, with the difference that RODD-LB2 is more efficient in terms of convergence compared to RODD-LB2. RODD-LB1 will be introduced in this chapter, and RODD-LB2 will be introduced in chapter 7. All the RODD methods use the concept of CMT for determination of Global Asymptotic Stability (GAS) of the equilibrium point. Due to the importance of CMT method in stability analysis, the first part of this chapter is devoted to the determination of stability using the CMT method.

Global Asymptotic Stability using CMT

The stability analysis for a given dynamical system can be verified via several methods e.g., the Lyapunov theorem, LasSalle's theorem, etc. The CMT method, which is one of these techniques, is easy to implement, but it is conservative. This is because the space of stable parameters derived from the CMT method is usually smaller than those pro-

duced by other methods. Unlike the CMT criterion, the RODD method, for example, is difficult to implement, but it is less conservative.

Contraction Mapping Theorem

Because the RODD methods are related to the contraction mapping theorem, the CMT will be explained in detail in this section. We will begin with some preliminary definitions.

Definition 6.1: *Let* $\mathbf{X} \subset \mathbf{R}^n$. *A semi-metric on* $\mathbf{X}$ *is a function* $d : \mathbf{X} \times \mathbf{X} \to \mathbf{R}$ *such that*

1.1. $d(x, y) \geq 0, d(x, x) = 0$

2.2. $d(x, y) = d(y, x)$

3.3. $d(x, y) \leq d(x, z) + d(z, y)$

for each $x, y, z \in \mathbf{X}$. *If in addition* $d(x, y) = 0$ *implies* $x = y$ *then* d *is called a metric.*

Definition 6.2: *Let* $(\mathbf{X}, d)$ *be a metric space. A mapping* $f : \mathbf{X} \to \mathbf{X}$ *is a contraction mapping if there exists a constant* c, *with* $0 \leq c < 1$ *such that*

$$d(f(x), f(y)) \leq cd(x, y) \quad \forall x, y \in \mathbf{X} \tag{6.1}$$

By the contraction mapping definition, if the map contracts, then the image of a set under the map will shrink by a factor $c < 1$. This is the basic concept behind the RODD stability analysis method. The method will be explained in detail later in this chapter.

Definition 6.3: *A point* $\mathbf{x}^*$ *is said to be an equilibrium point of dynamical system* $f : \mathbf{X} \to \mathbf{X}$ *if* $\mathbf{x}^* = f(\mathbf{x}^*)$ *[Elya98].*

Now by knowing the definition of metric space and equilibrium point we can proceed to introduce the contraction mapping theorem.

Theorem 6.1 Let $\mathbf{X}$ be a complete metric space and $f : \mathbf{X} \to \mathbf{X}$ be a map such that

$$d(f(x), f(y)) \leq c\,d(x, y) \tag{6.2}$$

for all $x, y \in \mathbf{X}$ *and* $0 \leq c < 1$. *Then* f *has a unique equilibrium point in* $\mathbf{X}$. *For any* $x_0 \in \mathbf{X}$ *the sequence of iterates* $x_0, f(x_0), f(f(x_0)), \ldots$ *converges to the equilibrium point of* f.

Proof: For the proof of the theorem refer to [Bana22].

As explained in chapter 2, the state space representation is one way of representing dynamical systems. Let's consider an RNN, which is an example of a dynamical system, to be represented by a state space equation as follows

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{6.3}$$

where $\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^n$ and $\mathbf{x} \in \mathbf{R}^n$. The objective here is to derive a criterion which guarantees that the origin of system (6.3) is GAS. If the origin is not the equilibrium point of system (6.3), then an affine transformation can shift the non-zero equilibrium to the origin without loss of generality. In order to proceed with the derivation of the stability criterion there is a need to introduce the Mean Value Theorem (MVT).

Theorem 6.2 Mean Value Theorem (MVT)[Buck78]: Let $f \in C'$ in an open convex set $\mathbf{X}$ in n space. Then for any point $\mathbf{p}_1$ and $\mathbf{p}_2$ in $\mathbf{X}$ there is a point $\mathbf{p}^*$ lying on the segment joining them such that

$$f(\mathbf{p}_2) - f(\mathbf{p}_1) = \mathbf{D}_f\big|_{p^*} \cdot (\mathbf{p}_2 - \mathbf{p}_1) \tag{6.4}$$

where $\mathbf{D}$ *is a Jacobian Matrix.*

In order to derive a general criterion for global stability of (6.3) we will merge the CMT and the MVT. If the vector function $\mathbf{f} \in C'$, then by the MVT there exists $\mathbf{z}$ lying on the segment joining $\mathbf{x}$ and $\mathbf{y}$ such that

$$\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) = \mathbf{D}_f\big|_{\mathbf{z}} \cdot (\mathbf{x} - \mathbf{y}) \tag{6.5}$$

Taking the norm of both sides of the above equation, and considering the Cauchy-Schwarz inequality, yields

$$\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\| \le \left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\| \cdot \|\mathbf{x} - \mathbf{y}\| \tag{6.6}$$

If $\left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\| < 1$ then we can chose $c = max\left\{\left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\|\right\}$ and by the CMT the origin of system (6.3) will be GAS, and all the system trajectories starting from arbitrary initial conditions converge to the origin. The criterion $\left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\| < 1$ is a sufficient condition, but not a necessary condition, for stability.

In the following section the same method will be used to derive a criterion for global asymptotic stability of the Lur'e model explained in chapter 3. The Lur'e model has been chosen for stability analysis, because, as we will explain in chapter 8, the overall model of an RNN controller and an RNN emulator in the Model Reference Control system can be put in the Lur'e model form.

Global Stability Analysis of Neural Network Model

In this section, the contraction mapping stability criterion developed in the previous section will be applied to two models introduced in Chapter 3: the Lur'e model and the BPRNN model.

The Lur'e model (3.4) can be represented by

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k)) \\ &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\{\mathbf{f}(\mathbf{W}\mathbf{x}(k) + \mathbf{c}) - \mathbf{f}(\mathbf{c})\} \end{aligned} \tag{6.7}$$

From MVT there exists $\mathbf{z}$ lying on the segment joining $\mathbf{x}$ and $\mathbf{y}$ such that

$$\begin{aligned} &\mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{x}(k) + \mathbf{c}) - \mathbf{B}\mathbf{f}(\mathbf{c}) - \mathbf{A}\mathbf{y}(k) - \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{y}(k) + \mathbf{c}) - \mathbf{B}\mathbf{f}(\mathbf{c}) \\ &\qquad = \mathbf{D}_{\mathbf{f}}\big|_{\mathbf{z}} \cdot (\mathbf{x} - \mathbf{y}) \end{aligned} \tag{6.8}$$

Taking the norm of both sides of (6.8) and canceling $\mathbf{B}\mathbf{f}(\mathbf{c})$ from the left hand side yields

$$\|\mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{x}(k) + \mathbf{c}) - \mathbf{A}\mathbf{y}(k) - \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{y}(k) + \mathbf{c})\| = \left\| \mathbf{D}_{\mathbf{f}}\big|_{\mathbf{z}} \cdot (\mathbf{x} - \mathbf{y}) \right\|$$

Applying the Cauchy-Schwarz inequality to the above equation yields

$$\|\mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{x}(k) + \mathbf{c}) - \mathbf{A}\mathbf{y}(k) - \mathbf{B}\mathbf{f}(\mathbf{W}\mathbf{y}(k) + \mathbf{c})\| \le \left\| \mathbf{D}_{\mathbf{f}}\big|_{\mathbf{z}} \right\| \|(\mathbf{x} - \mathbf{y})\| \tag{6.9}$$

By designing the network parameters in (6.7) such that $\left\| \mathbf{D}_{\mathbf{f}}\big|_{\mathbf{z}} \right\| < 1$, we can guarantee that the origin of the system is GAS, because we can always chose c in the contraction mapping theorem to be $c = max\left\{ \left\| \mathbf{D}_{\mathbf{f}}\big|_{\mathbf{z}} \right\| \right\}$, hence the equilibrium point of $\mathbf{f}$ will be GAS.

The other neural network model which will be investigated for stability analysis is the BPRNN model introduced in (3.1). The network behavior at the steady state can be written as

$$\begin{aligned} \mathbf{z} &= \mathbf{f}(\mathbf{z}) \\ &= \tanh(\mathbf{W}\mathbf{z} + \mathbf{b}) \end{aligned} \tag{6.10}$$

where

$$\mathbf{z} = \left[\mathbf{z}^1 \ \mathbf{z}^2 \ \dots \ \mathbf{z}^M\right]^T \quad , \mathbf{b} = \left[\mathbf{b}^1 \ \mathbf{b}^2 \ \dots \ \mathbf{b}^M\right]^T$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{LW}^{1,1} & 0 & \dots & 0 & \mathbf{LW}^{1,M} \\ \mathbf{LW}^{2,1} & \mathbf{LW}^{2,2} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & \dots & \mathbf{LW}^{M,M-1} & \mathbf{LW}^{M,M} \end{bmatrix} \tag{6.11}$$

In order to show the equilibrium point of the BPRNN is GAS using CMT, we need to show that $\left\|\mathbf{D_f}\big|_{\mathbf{z}}\right\| < 1$. Since the dynamics of the BPRNN model is known, the Jacobian matrix can be derived analytically as follows

$$\mathbf{D_f}\big|_{\mathbf{z}} = \Lambda\mathbf{W} \tag{6.12}$$

where

$$\Lambda = \begin{bmatrix} 1-\tanh(\mathbf{\Pi}_1)^2 & 0 & \dots & 0 \\ 0 & 1-\tanh(\mathbf{\Pi}_2)^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1-\tanh(\mathbf{\Pi}_M)^2 \end{bmatrix} \tag{6.13}$$

$$\begin{aligned} \mathbf{\Pi}_1 &= \mathbf{LW}^{1,1}\mathbf{z}^1 + \mathbf{LW}^{1,M}\mathbf{z}^M + \mathbf{b}^1 \\ \mathbf{\Pi}_2 &= \mathbf{LW}^{2,2}\mathbf{z}^2 + \mathbf{LW}^{2,1}\mathbf{z}^1 + \mathbf{b}^2 \\ &\dots \\ \mathbf{\Pi}_M &= \mathbf{LW}^{M,M}\mathbf{z}^M + \mathbf{LW}^{M,M-1}\mathbf{z}^{M-1} + \mathbf{b}^M \end{aligned} \tag{6.14}$$

In the following theorem, we will show that the upper bound of $\left\|\mathbf{D_f}\big|_{\mathbf{z}}\right\|$ depends on the BPRNN parameters.

Theorem 6.3 If $\mathbf{D}_f\big|_{\mathbf{z}}$ is defined in (6.12), then $max\left\{\left\|\mathbf{D}_f\big|_{\mathbf{z}^*}\right\|\right\} \le \|\mathbf{W}\|$ where $\mathbf{z}^* = \left[\mathbf{z}^{*1}\ \mathbf{z}^{*2}\ \ldots\ \mathbf{z}^{*M}\right]^T$ is defined such that $\mathbf{\Pi}_1 = \mathbf{\Pi}_2 = \ldots = \mathbf{\Pi}_M = 0$.

Proof Take the l_2 norm of both sides of (6.12), then by [Meye00]

$$\begin{aligned}\left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\|_2 &= \|\Lambda\mathbf{W}\|_2 \\ &\le \|\Lambda\|_2\|\mathbf{W}\|_2\end{aligned} \tag{6.15}$$

so if we can show that $\|\Lambda\|_2 \le 1$, then $\left\|\mathbf{D}_f\big|_{\mathbf{z}}\right\|_2 \le \|\mathbf{W}\|_2$. Rewrite Λ, defined in (6.13), as

$$\Lambda = \begin{bmatrix} \alpha_1 & 0 & \ldots & 0 \\ 0 & \alpha_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \alpha_n \end{bmatrix} \tag{6.16}$$

where $\alpha_1 = 1 - \tanh(\mathbf{\Pi}_1)^2$, $\alpha_2 = 1 - \tanh(\mathbf{\Pi}_2)^2$, …, $\alpha_n = 1 - \tanh(\mathbf{\Pi}_n)^2$. Then consider an arbitrary $\mathbf{x} \in \mathbf{R}^n$ and construct $\Lambda\mathbf{x}$. Considering the definition of the l_2 norm, $\left(\frac{\|\Lambda\mathbf{x}\|_2}{\|\mathbf{x}\|_2}\right)^2$ can be written as

$$\left(\frac{\|\Lambda\mathbf{x}\|_2}{\|\mathbf{x}\|_2}\right)^2 = \frac{(\alpha_1 x_1)^2 + (\alpha_2 x_2)^2 + \ldots + (\alpha_n x_n)^2}{(x_1)^2 + (x_2)^2 + \ldots + (x_n)^2} \tag{6.17}$$

Define $\mathbf{x}^* = \arg\ \max_{\mathbf{x}}\left(\frac{\|\Lambda\mathbf{x}\|_2}{\|\mathbf{x}\|_2}\right)^2$. The equation given in (6.17) is a weighted average of $(\alpha_i)^2$, hence

$$(\alpha_{i^*})^2 = max\{(\alpha_i)^2\} \ge \left(\frac{\|\Lambda\mathbf{x}\|_2}{\|\mathbf{x}\|_2}\right)^2 \ge min\{(\alpha_i)^2\} \tag{6.18}$$

where $\alpha_{i^*} \geq \alpha_i$ for $i = 1, 2, \ldots, n$. The maximum of $\left(\frac{\|\Lambda \mathbf{x}\|_2}{\|\mathbf{x}\|_2}\right)^2$ occurs when $\mathbf{x}^*_{i^*} = 1$ and $\mathbf{x}^*_i = 0$ for $i \neq i^*$ and by the definition of the induced norm

$$\begin{aligned} \|\Lambda\| &= \max_{\mathbf{x}} \left\{ \left(\frac{\|\Lambda \mathbf{x}\|_2}{\|\mathbf{x}\|_2} \right) : \mathbf{x} \in \mathbf{R}^n \right\} \\ &= \alpha_{i^*} \\ &\leq 1 \quad \text{From the definition of } \alpha_i \end{aligned} \tag{6.19}$$

The inequality derived in (6.19) proves the theorem. □

By Theorem 6.3 the origin of the BPRNN given in (3.1) is GAS whenever the norm of $\mathbf{W}$ given in (6.11) is less than 1. When $\|\mathbf{W}\| < 1$, then by Theorem 6.3 $\left\| \mathbf{D}_{\mathbf{f}}|_{\mathbf{z}} \right\| < 1$, and the map $\mathbf{f}$ given in (6.10) will be a contraction mapping with $c = \max\left\{ \left\| \mathbf{D}_{f}|_{\mathbf{z}} \right\| \right\}$. In the following example, we will demonstrate this result.

Example 6.1 Consider a BPRNN defined in (3.1) with two layers ($M = 2$), with 2 neurons in the first layer and 3 neurons in the second layer. The network dynamics can be written as follows

$$\begin{aligned} \mathbf{a}^1(k+1) &= \tanh(\mathbf{LW}^{1,1}\mathbf{a}^1(k) + \mathbf{LW}^{1,2}\mathbf{a}^2(k) + \mathbf{b}^1) \\ \mathbf{a}^2(k+1) &= \tanh(\mathbf{LW}^{2,2}\mathbf{a}^2(k) + \mathbf{LW}^{2,1}\mathbf{a}^1(k+1) + \mathbf{b}^2) \end{aligned} \tag{6.20}$$

We would like to design a BPRNN with GAS equilibrium point. This network at the steady state can be written in the form of (6.10), where

$$\mathbf{W} = \begin{bmatrix} \mathbf{LW}^{1,1} & \mathbf{LW}^{1,2} \\ \mathbf{LW}^{2,1} & \mathbf{LW}^{2,2} \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \mathbf{z}^1 \\ \mathbf{z}^2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \end{bmatrix} \tag{6.21}$$

According to the CMT criterion, the equilibrium point of (6.20) is GAS if we choose the network parameters such that $\|\mathbf{W}\| < 1$. This condition satisfied with

$\mathbf{LW}^{1,1} = \begin{bmatrix} -0.31 & 0.13 \\ 0.2 & 0.15 \end{bmatrix}$, $\mathbf{LW}^{1,2} = \begin{bmatrix} 0.11 & 0.25 & 0.09 \\ 0.39 & -0.25 & -0.44 \end{bmatrix}$, $\mathbf{LW}^{2,1} = \begin{bmatrix} 0.17 & -0.57 \\ -0.23 & -0.49 \\ 0.31 & -0.43 \end{bmatrix}$, and

$\mathbf{LW}^{2,2} = \begin{bmatrix} 0.11 & -0.09 & -0.53 \\ -0.11 & 0.43 & 0.14 \\ 0.44 & 0.07 & 0.45 \end{bmatrix}$. Considering these values, $\|\mathbf{W}\| = 0.95$. Figure (6.1) illustrates the system (6.20) response for a random initial condition. The figure shows that the equilibrium point

$z = \begin{bmatrix} 0.85 & -0.61 & 0.85 & -0.61 & 0.99 & -0.25 & -0.95 & 0.99 & -0.25 & -0.95 \end{bmatrix}$ is GAS.

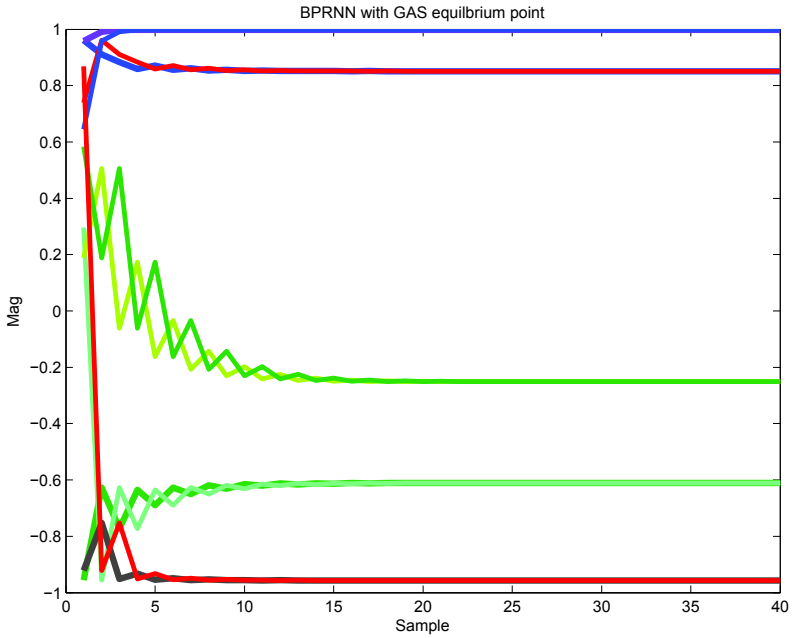

Figure (6.1) BPRNN with GAS Equilibrium point

The main problem with the criterion derived using CMT is the fact that it does not produce the most complete results. This is because the space of stable parameters derived by this method is too small. The following example demonstrates a stable system for which the contraction mapping theorem fails to detect stability.

Example 6.2 Consider a simple 2-D system

$$\mathbf{x}(k+1) = \mathbf{W}\tanh(\mathbf{x}(k)) \text{ where } \mathbf{W} = \begin{bmatrix} 1.8 & 0.95 \\ -0.95 & 0 \end{bmatrix} \tag{6.22}$$

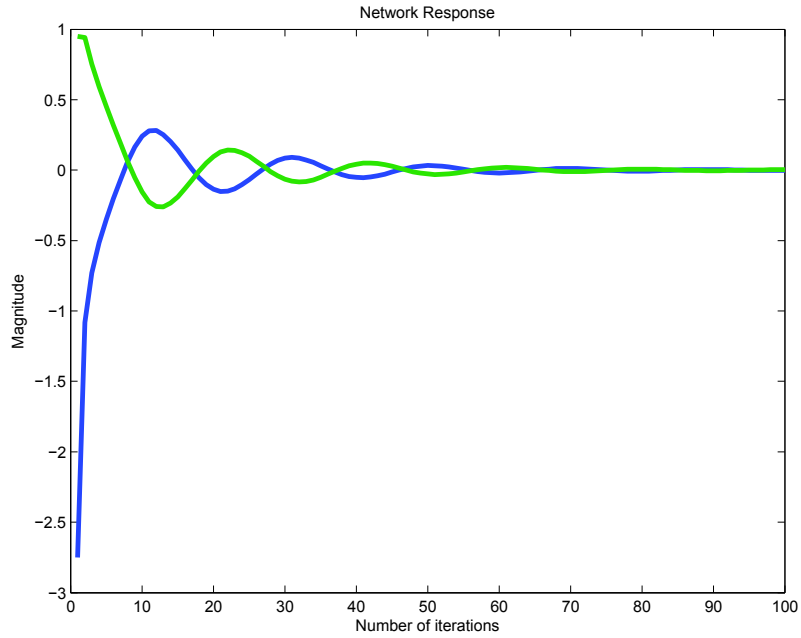

Figure (6.2) Network Respond after 100 iterations

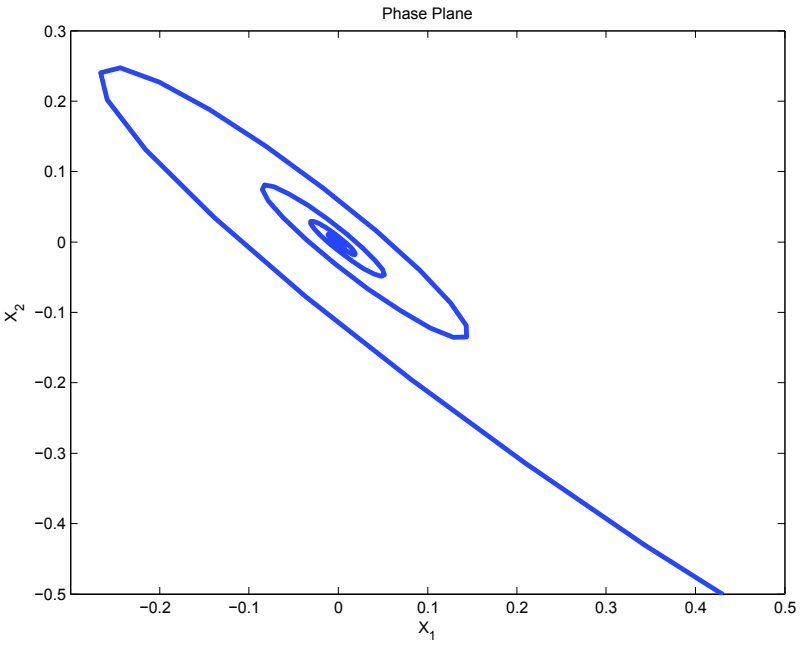

Figure (6.3) Phase plane

The origin of the network in this example is GAS, but the CMT method fails to determine

stability because

$$\|\mathbf{W}\|_2 = 2.2086 > 1 \tag{6.23}$$

The above example shows that the space of stable parameters derived by the contraction mapping theorem is not complete. In the following section the RODD-LB1 method [BaPr03] will be introduced as an alternative technique that discovers a larger space of stable parameters.

Global Stability Analysis Using RODD-LB1

The RODD-LB1 method, which will be introduced in detail in this section, not only derives a larger space of stable parameters, but it can also be applied to a wider class of systems. The only constraint for applying this method is that the system needs to be differentiable and bounded.

Before explaining the RODD-LB1 method, we will define reachable sets and approximate reachable sets.

Reachable Set

Consider a dynamical system given by

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{6.24}$$

where $\mathbf{f} : \mathbf{R}^n \rightarrow \mathbf{R}^n$ $\mathbf{x} \in \mathbf{R}^n$ ($\mathbf{x}$ is the state of the system) and assume that $\mathbf{f} = col\{f_i\}$ consists of known smooth bounded functions for $i = 1, 2, \ldots, n$. For each initial condition $\mathbf{x}(0) \in \mathbf{R}^n$ there exists a trajectory of system (6.24). The system trajectories at time step k under all possible initial conditions $\mathbf{x}(0) \in \mathbf{R}^n$ make up a set called the *reachable set* [Cher93] and denoted by $\mathbf{D}_k^*$. The reachable sets are fundamental characteristics of dy-

namical systems. To determine the reachable set, system (6.24) can be solved recursively as follows:

$$\begin{aligned} \mathbf{x}(1) &= \mathbf{f}(\mathbf{x}(0)) \\ \mathbf{x}(2) &= \mathbf{f}(\mathbf{x}(1)) = \mathbf{f}(\mathbf{f}(\mathbf{x}(0))) = \mathbf{f}^2(\mathbf{x}(0)) \\ &\cdots \\ \mathbf{x}(k) &= \mathbf{f}^k(\mathbf{x}(0)) \end{aligned} \tag{6.25}$$

The reachable set $\mathbf{D}^*_{k+1}$ is the image of the set of initial conditions $\mathbf{R}^n$ under the mapping $\mathbf{f}^k$. The graphical representation of the reachable set is given in Figure (6.4)

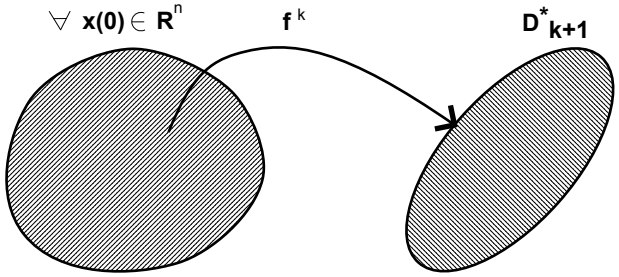

Figure (6.4) Reachable Set

The exact knowledge of the reachable sets plays an important role in control theorey. Many basic problems of this theory can be solved in terms of reachable sets. For example, we can determine the stability or instability of system (6.24) based on the definition of the reachable set. Ideally when the exact knowledge of the reachable set is available, we will have the following equality

$$\mathbf{D}^*_{k+1} = \mathbf{f}(\mathbf{D}^*_k) \tag{6.26}$$

In order to prove that the origin of system (6.24) is GAS, it is enough to show that $\mathbf{D}^*_k \to 0$ as $k \to \infty$. This means that regardless of the initial conditions, all possible trajec-

tories converge to the origin. This is the definition of GAS for a dynamical system [Khal01]. Figure (6.5) shows an example of a reachable set that shrinks forward in time.

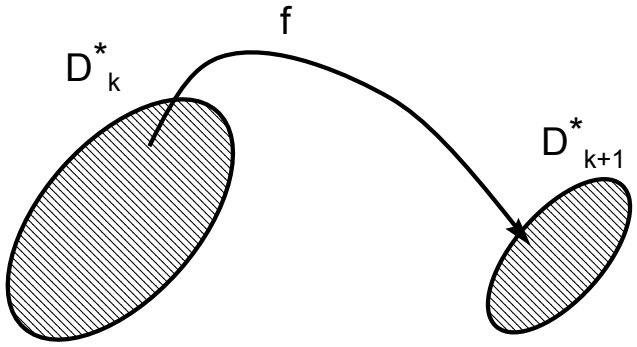

Figure (6.5) Sequence of Reachable Sets

Similarly, we can prove the lack of global stability of the origin for dynamical systems by showing that

$$\mathbf{D}^*_k \subset \mathbf{D}^*_{k+1} \tag{6.27}$$

In this work we will focus on systems with reachable sets that are convex. This will be the case when system (6.24) is a state space representation of a general type of RNN with activation function satisfying sector conditions (the sector conditions explained in chapter 4). However, the shape of reachable sets for general dynamical systems could be more complex.

The main difficulty with using reachable sets to determine stability is the fact that they cannot generally be derived exactly. For this reason, there is a need for an approximation. Several methods for approximating reachable sets are available, and they will be explained in detail in the following section.

Approximate Reachable Set

The approximation of convex reachable sets can be accomplished through a variety of different methods. The RODD-LB1 method approximates convex reachable sets by using a set of linear boundaries. In the next chapter, we will introduce a new method which approximates reachable sets by using quadratic functions.

For a good approximation of a reachable set, there is a need to produce a set $\mathbf{D}_k$ that contains all the system trajectories starting from arbitrary initial conditions. In the other words,

$$\mathbf{f}(\mathbf{D}^*_k) = \mathbf{D}^*_{k+1} \subset \mathbf{D}_{k+1} \tag{6.28}$$

The approximate reachable set $\mathbf{D}_k$ must contain the true reachable set $\mathbf{D}^*_k$ so that if $\{\mathbf{D}_k\} \to \mathbf{0}$ as $k \to \infty$ then it guarantees that $\{\mathbf{D}^*_k\} \to \mathbf{0}$, which proves that the origin of (6.24) is GAS.

An example of using linear boundaries to approximate a convex reachable set is shown in Figure (6.6). The linear approximation has the advantage that increasing the number of linear boundaries will increase the accuracy. If the number of linear boundaries goes to infinity then $\mathbf{D}_{k+1} \to \mathbf{f}(\mathbf{D}^*_k)$

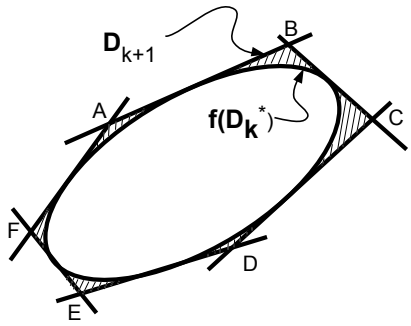

Figure (6.6) Approximate Reachable Set

Figure (6.7) illustrates how the accuracy of the approximation of the convex reachable set can be increased by increasing the number of linear boundaries.

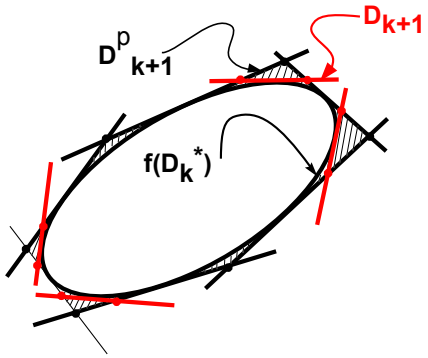

Figure (6.7) Better Approximation of Reachable Set

In the following section we will explain the RODD-LB1 method, which uses linear boundaries for approximating the reachable set and detecting the stability of the origin for system (6.24)

RODD-LB1[BaPr03]

The method of reduction of dissipativity domain is a computational technique to prove global stability of the origin. The efficiency of the method depends on how accurately the reachable set can be approximated. The approximate reachable set at time step $k + 1$

($\mathbf{D}_{k+1}$) will be compared against the approximate reachable set at the previous time step k ($\mathbf{D}_k$) to check for a reduction in size. The continuous shrinkage of the reachable set to $\{0\}$ is an indication of the asymptotic stability of the origin, whereas the enlargement of the reachable set may be an indication of instability. It is possible that the system could be GAS even though the approximate reachable set does not shrink. The enlargement could be due to an approximation error between the actual reachable set and the approximate reachable set. In other words, the RODD-LB1 method cannot prove instability, due to potential approximation errors in the reachable set.

The concept of the RODD-LB1 method is based on the CMT. We intend to construct a sequence of sets $\{\mathbf{D}_k\}$ such that $\mathbf{D}_{k+1} \subset \mathbf{D}_k$. Also, it is important that at every step k the set $\mathbf{D}_k$ contains the set $\mathbf{D}^*_k$ (all the system trajectories at time step k starting from arbitrary initial conditions). One way to keep all system trajectories inside the set $\mathbf{D}_k$ is to define the set $\mathbf{D}_0$ as

$$\mathbf{D}_0 = \{\mathbf{x} : |x_i| \le max\{|f_i(\mathbf{x})|\} \quad \mathbf{x} \in \mathbf{R}^n\} \text{ for } i = 1, 2, \ldots, n \tag{6.29}$$

By the Extreme Value Theorem (EVT) the maximum of f_i can be achieved because the function is assumed to be continuous and the set $\mathbf{R}^n$ is bounded. The set $\mathbf{D}_0$ contains all possible system trajectories after one time step starting from arbitrary initial conditions chosen from the set $\mathbf{R}^n$. In fact, the set $\mathbf{D}_0$ is a rough approximation for the set $\mathbf{f}(\mathbf{R}^n)$. In order to define the set $\mathbf{D}_{k+1}$ for $k > 0$ we need to consider a set of continuous functions

$\{h_{k,1}, h_{k,2}, \ldots, h_{k,m_k}\}$, where $h_{k,j} : \mathbf{R}^n \to \mathbf{R}$. Denote

$$\gamma_{k,j} = max\{h_{k,j}(\mathbf{f}(\mathbf{x})) : \mathbf{x} \in \mathbf{D}_k\} \tag{6.30}$$

Knowing $h_{k,j}$ and $\gamma_{k,j}$ we can define the set $\mathbf{D}_{k+1}$ as

$$\mathbf{D}_{k+1} = \{\mathbf{x} \in \mathbf{D}_k : h_{k,j}(\mathbf{x}) \le \gamma_{k,j} \quad \forall j = 1, 2, \ldots, m_k\} \tag{6.31}$$

The functions $h_{k,j}$ and f_i are continuous and $\mathbf{D}_k$ is a closed and bounded set (compact). By the EVT, a continuous real valued function on a compact set obtains its maximum so, $\gamma_{k,j}$ is computable. The definition for $\mathbf{D}_{k+1}$ given in (6.31) is constructive if the set of functions $h_{k,j}$ are known. The key factor in the effectiveness and applicability of this method depends strongly on the choice of $h_{k,j}$. There are several possibilities for the choice of $h_{k,j}$. Linear $h_{k,j}$ functions will be presented in this chapter. In the next chapter a set of quadratic $h_{k,j}$ will be discussed as an alternative to the current method.

Theorem 6.4 Consider the definition for $\mathbf{D}_{k+1}$ in (6.31). Show that $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_{k+1} \subseteq \mathbf{D}_k$.

Proof: The proof of this theorem can be divided into three parts. These parts can be listed as follows:

4.1. $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_k$,

5.2. $\mathbf{D}_{k+1} \subseteq \mathbf{D}_k$,

6.3. $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_{k+1}$.

We will prove part 1 by induction. Let $\mathbf{y} \in \mathbf{f}(\mathbf{D}_0)$ where $\mathbf{y} \in \mathbf{f}(\mathbf{x})$ for some $\mathbf{x} \in \mathbf{D}_0$. Thus

$$|y_i| = |f_i(\mathbf{x})| \le \max_{\mathbf{z} \in \mathbf{D}_0} |f_i(\mathbf{z})| \le \max_{\mathbf{z} \in \mathbf{R}^n} |f_i(\mathbf{z})| \tag{6.32}$$

Thus $\mathbf{y} \in \mathbf{D}_0$. This implies that $\mathbf{f}(\mathbf{D}_0) \subseteq \mathbf{D}_0$. For the case when $k > 0$, we need to show if $\mathbf{f}(\mathbf{D}_{k-1}) \subseteq \mathbf{D}_{k-1}$ then $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_k$. In order words we need to show that if $\mathbf{x} \in \mathbf{D}_k$ then $\mathbf{f}(\mathbf{x}) \in \mathbf{D}_k$. According to (6.31) the set $\mathbf{D}_{k-1}$ and $\mathbf{D}_k$ can be written as

$$\begin{aligned} \mathbf{D}_{k-1} &= \{\mathbf{x} \in \mathbf{D}_{k-2} : h_{k-2,j}(\mathbf{x}) \le \gamma_{k-2,j} \quad \forall j = 1, 2, \ldots, m_k\} \\ \mathbf{D}_k &= \{\mathbf{x} \in \mathbf{D}_{k-1} : h_{k-1,j}(\mathbf{x}) \le \gamma_{k-1,j} \quad \forall j = 1, 2, \ldots, m_k\} \end{aligned} \tag{6.33}$$

Assume $\mathbf{x} \in \mathbf{D}_k$ then by (6.33) $\mathbf{x} \in \mathbf{D}_{k-1}$ and by induction hypothesis $\mathbf{f}(\mathbf{x}) \in \mathbf{D}_{k-1}$. Note that $\gamma_{k-1,j} = \max\{h_{k-1,j}(\mathbf{f}(\mathbf{t})) : \mathbf{t} \in \mathbf{D}_{k-1}\}$ and considering the assumption $\mathbf{x} \in \mathbf{D}_{k-1}$ we will have the following inequality

$$h_{k-1,j}(\mathbf{f}(\mathbf{x})) \le \max_{\mathbf{t} \in \mathbf{D}_{k-1}} h_{k-1,j}(\mathbf{f}(\mathbf{t})) \tag{6.34}$$

The inequality (6.34) is valid because $\mathbf{x} \in \mathbf{D}_{k-1}$, and the left hand side is an element of the set $\{h_{k-1,j}(\mathbf{f}(\mathbf{x})) : \mathbf{t} \in \mathbf{D}_{k-1}\}$. Inequality (6.34) and $\mathbf{f}(\mathbf{x}) \in \mathbf{D}_{k-1}$ implies that $\mathbf{f}(\mathbf{x}) \in \mathbf{D}_k$. In other words $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_k$.

The proof to part 2 can be derived by inspection through the definition of $\mathbf{D}_{k+1}$ given in (6.31).

In order to prove part 3 , we need to show that $\mathbf{f}(\mathbf{D}_k) \subseteq \mathbf{D}_{k+1}$. Suppose $\mathbf{u} \in \mathbf{f}(\mathbf{D}_k)$, we need to show $\mathbf{u} \in \mathbf{D}_{k+1}$. In other words, we need to show if $\mathbf{w} \in \mathbf{D}_k$ then $\mathbf{f}(\mathbf{w}) \in \mathbf{D}_{k+1}$. Suppose $\mathbf{w} \in \mathbf{D}_k$ then by the 1 , $\mathbf{f}(\mathbf{w}) \in \mathbf{D}_k$ and the following inequality holds,

$$h_{k,j}(\mathbf{f}(\mathbf{w})) \leq \max_{\mathbf{t} \in \mathbf{D}_k} h_{k,j}(\mathbf{f}(\mathbf{t})) \tag{6.35}$$

because $\mathbf{w} \in \mathbf{D}_k$ and the left side of inequality (6.35) searches for $\mathbf{t} \in \mathbf{D}_k$. Considering the definition of $\mathbf{D}_{k+1}$ in (6.31), $\mathbf{f}(\mathbf{w}) \in \mathbf{D}_k$ and inequality (6.35) implies $\mathbf{f}(\mathbf{w}) \in \mathbf{D}_{k+1}$, as it is required. This proves part 3 . □

The choice of the set of continuous functions $h_{k,j}$ is important for approximating reachable sets. The following theorem proves the existence of functions $h_{k,j}$ for systems with a GAS equilibrium point.

Theorem 6.5 [BaPr03] Assume that the origin is the equilibrium point of system (6.24) and it is GAS. Then there exists a function h such that $\{\mathbf{D}_k\} \to \{0\}$ as $k \to \infty$.

Proof: The equilibrium point of system (6.24) is GAS, so by the converse theorem [Khal01] there exists a continuous function $V: \mathbf{R}^n \to \mathbf{R}$ such that

7.1. $V(\mathbf{x}) > 0 \quad \forall \mathbf{x} \neq 0$,

8.2. $V(\mathbf{0}) = 0$,

9.3. $\Delta V(\mathbf{x}) = V(\mathbf{f}(\mathbf{x})) - V(\mathbf{x}) < 0$.

Let $h \equiv V$. Considering (6.30), γ_k can be written as

$$\gamma_k = max\{V(\mathbf{f}(\mathbf{x})) \ : \ \mathbf{x} \in \mathbf{D}_k\} \tag{6.36}$$

and so $\gamma_k = V(\mathbf{f}(\mathbf{p}_k))$ for some $\mathbf{p}_k \in \mathbf{D}_k$, since the sets $\mathbf{D}_k$ are all compact, and V and $\mathbf{f}$ are continuous. By definition of $\mathbf{D}_k$, $V(\mathbf{f}(\mathbf{p}_k)) \leq \gamma_{k-1}$ and it follows that

$$\gamma_k = V(\mathbf{f}(\mathbf{p}_k)) < V(\mathbf{p}_k) \leq \gamma_{k-1} \tag{6.37}$$

That is, γ_k is a decreasing sequence of real numbers. Since $\gamma_k \geq 0$ for all k, it follows that $\gamma_k \to \tau$ for some $\tau \geq 0$.

We wish to show that $\tau = 0$. Suppose $\tau > 0$. By the continuity of V at the origin, there is a $\delta > 0$ such that if $\|\mathbf{x}\| < \delta$ then $V(\mathbf{x}) < \tau$. The set $C = \mathbf{D}_0 - \mathbf{B}(0, \delta)$ is closed and bounded and, therefore, compact. The function ΔV is continuous and strictly negative on C, and hence there is a $\rho > 0$ such that $\Delta V(\mathbf{x}) \leq -\rho$ for $\mathbf{x} \in C$. For all $k \geq 0$ we have $V(\mathbf{p}_k) \geq \gamma_k \geq \tau$ and so $\mathbf{p}_k \notin \mathbf{B}(0, \delta)$. Thus $\mathbf{p}_k \in C$ for all k, and we may improve the above inequality to

$$\gamma_k = V(\mathbf{f}(\mathbf{p}_k)) \leq V(\mathbf{p}_k) - \rho \leq \gamma_{k-1} - \rho \tag{6.38}$$

for each $k \geq 1$. However, if γ_k decreases by at least a fixed positive amount each time, then it will eventually become negative. This contradiction shows that $\tau = 0$ so that $\gamma_k \to 0$.

Now let $\varepsilon > 0$. We wish to show that there is some M such that $\mathbf{D}_k \subset \mathbf{B}(0, \varepsilon)$ for $k \geq M$. The function V achieves a minimum value on the set $\mathbf{D}_0 - \mathbf{B}(0, \varepsilon)$, and this minimum value is positive. Call it $\alpha > 0$. By the previous step, we may find some M such that $\gamma_k < \alpha$ for all $k \geq M - 1$. Suppose that $\mathbf{y} \in \mathbf{D}_k$ with $k \geq M$. Then $\mathbf{y} \in \mathbf{D}_0$ and

$V(\mathbf{y}) \le \gamma_{k-1} < \alpha$. It follows that $\mathbf{y} \notin \mathbf{D}_0 - \mathbf{B}(0, \varepsilon)$, and so $\mathbf{y} \in \mathbf{B}(0, \varepsilon)$. That is, $\mathbf{D}_k \subset \mathbf{B}(0, \varepsilon)$ for all $k \ge M$. This concludes the proof. □

The above theorem shows the existence of a continuous function h for systems with GAS equilibrium point. However, the method is not constructive. The accuracy of RODD-LB1 depends on the right choice for h. If we chose the right h, then the boundaries of the sets $\{\mathbf{x} : h_{k,j}(\mathbf{x}) \le \gamma_{k,j}, \forall j = 1, 2, \ldots, m_k\}$ approximate the reachable set at every time step k. The RODD-LB1 method with a sufficient number of linear boundaries can approximate exact reachable sets with sufficiently small errors. The relationship between increasing the number of linear boundaries and decreasing the errors is shown in Figure (6.7).

Theorem 6.5 shows that if the origin of system (6.24) is GAS, then the set $\mathbf{D}_k$ is contained in a ball of radius ε (where ε is a small positive number) centered at the origin ($\mathbf{D}_k \subset \mathbf{B}_{0,\varepsilon}$) as $k \to \infty$. However, it is possible that the origin of system (6.24) could be GAS even though the set $\mathbf{D}_k$ stabilizes as $k \to \infty$ and does not shrink. This is because $\mathbf{D}_k$ is an approximation to the set $\mathbf{f}(\mathbf{D}^*_{k-1})$, and there are errors associated with any approximation. The following theorem shows how to add additional linear boundaries (to better approximate $\mathbf{f}(\mathbf{D}^*_{k-1})$) when $\mathbf{D}_k \approx \mathbf{D}_{k+1}$. Increasing the number of linear boundaries makes the approximation more accurate and in most cases enables the set $\mathbf{D}_k$ to shrink to the origin.

Lemma 6. 1: *Consider dynamical system (6.24) with* $\mathbf{0}$ *as a globally asymptotically stable equilibrium point and convex Lyapunov function* V. *If we are given a compact,*

convex, invariant set other than $\{\mathbf{0}\}$*, we may construct a proper convex, compact invariant set by adding a linear inequality.*

Proof: Let $\mathbf{D}$ be a convex, compact invariant set that does not just contain $\mathbf{0}$. If $\mathbf{f}(\mathbf{D}) = \{\mathbf{0}\}$ then $\mathbf{x}_0 \in \mathbf{D} - \{\mathbf{0}\}$ and consider the set

$$\left\{\mathbf{y} \in \mathbf{D} : \mathbf{x}_0^T \mathbf{y} \leq \frac{1}{2}\mathbf{x}_0^T \mathbf{x}_0\right\}. \tag{6.39}$$

This set is closed in $\mathbf{D}$ and hence compact. It is invariant, because it contains $\mathbf{0}$. It is proper, because it does not contain $\mathbf{x}_0$. It is convex, because $\mathbf{D}$ is convex and adding a linear inequality intersects the set with a half space and hence preserves convexity. This deals with the case where $\mathbf{f}(\mathbf{D}) = \{\mathbf{0}\}$.

Now suppose that $\mathbf{f}(\mathbf{D}) \neq \{\mathbf{0}\}$. Let

$$\mathbf{x} = \arg \max_{\mathbf{z} \in \mathbf{D}} V(\mathbf{f}(\mathbf{z})) \tag{6.40}$$

and note that $\mathbf{x} \neq \mathbf{0}$, since $\mathbf{f}(\mathbf{D}) \neq \{\mathbf{0}\}$. Define

$$\mathbf{D}' = \{\mathbf{y} \in \mathbf{D} : V(\mathbf{y}) \leq V(\mathbf{f}(\mathbf{x}))\} \tag{6.41}$$

Note that $\mathbf{D}'$ is the intersection of the convex set $\mathbf{D}$ and the convex set $\{\mathbf{y} \in \mathbf{R}^n : V(\mathbf{y}) \leq V(\mathbf{f}(\mathbf{x}))\}$, thus $\mathbf{D}'$ is convex. Moreover $\mathbf{D}'$ is closed in $\mathbf{D}$ and hence is compact. If $\mathbf{w} \in \mathbf{D}$ then $\mathbf{f}(\mathbf{w}) \in \mathbf{D}$, since $\mathbf{D}$ is invariant, and $V(\mathbf{f}(\mathbf{w})) \leq V(\mathbf{f}(\mathbf{x}))$ by (6.40). Consequently, $\mathbf{f}(\mathbf{w}) \in \mathbf{D}'$ by (6.41). This shows that $\mathbf{f}(\mathbf{D}) \subseteq \mathbf{D}'$.

We have $V(\mathbf{x}) > V(\mathbf{f}(\mathbf{x}))$ by the definition of Lyapunov function, and so $\mathbf{x} \notin \mathbf{D}'$. By the Separating Hyperplane Theorem [GoYa02], there is some $\mathbf{h} \in \mathbf{R}^n - \{\mathbf{0}\}$ and some $c \in \mathbf{R}$ such that $\mathbf{h}^T\mathbf{x} > c$ and $\mathbf{h}^T\mathbf{y} < c$ for all $\mathbf{y} \in \mathbf{D}'$. Define

$$\gamma = max\{\mathbf{h}^T\mathbf{y} : \mathbf{y} \in \mathbf{f}(\mathbf{D})\} \tag{6.42}$$

and

$$\mathbf{D}'' = \{\mathbf{y} \in \mathbf{D} : \mathbf{h}^T\mathbf{y} \leq \gamma\} \tag{6.43}$$

Since $\mathbf{D}''$ is the intersection of $\mathbf{D}$ with a half space, $\mathbf{D}''$ is convex. Since $\mathbf{D}''$ is closed in $\mathbf{D}$, $\mathbf{D}''$ is compact. We have seen that $\mathbf{f}(\mathbf{D}) \subseteq \mathbf{D}'$ and hence $\mathbf{h}^T\mathbf{y} < c$ for all $\mathbf{y} \in \mathbf{f}(\mathbf{D})$. Thus $\gamma < c$, and it follows that $\mathbf{x} \notin \mathbf{D}''$. Finally, if $\mathbf{w} \in \mathbf{D}''$ then $\mathbf{w} \in \mathbf{D}$ and so $\mathbf{f}(\mathbf{w}) \in \mathbf{f}(\mathbf{D})$. Thus $\mathbf{z}^T\mathbf{f}(\mathbf{w}) \leq \gamma$, and so $\mathbf{f}(\mathbf{w}) \subseteq \mathbf{D}''$ from (6.42). This shows that $\mathbf{f}(\mathbf{D}'') \subseteq \mathbf{D}''$. This construction shows that if $\mathbf{D}$ is a compact, convex, invariant set other than $\{\mathbf{0}\}$ then we may construct $\mathbf{D}''$, a proper convex, compact, invariant subset of $\mathbf{D}$ by adding a linear inequality. This proves the lemma. The graphical representation of the proof is given in Figure (6.8) □

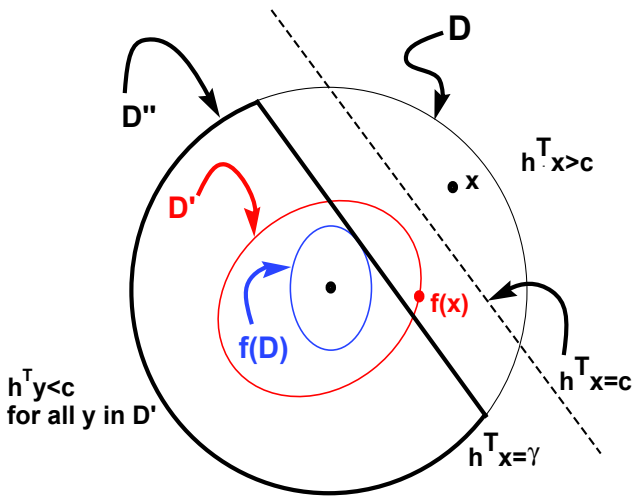

Figure (6.8) Hyperplane separating a point from a convex set

We will use Lemma 6. 1: to prove the following theorem.

Theorem 6.6 [BaPr03] Assume that there exists a convex Lyapunov function V for system (6.24). Then for any step k either there is some k such that $\mathbf{D}_k = \{\mathbf{0}\}$ or there exists a linear function h_k such that the set

$\mathbf{D}_{k+1} = \{\mathbf{x} \in \mathbf{D}_k : h_k(\mathbf{x}) \le max\{h_k(\mathbf{f}(\mathbf{x})) : \mathbf{x} \in \mathbf{D}_k\}\}$ is a proper subset of the set $\mathbf{D}_k$.

Proof: Let prove the theorem by induction. Initially, we need to show that $\mathbf{D}_1 \subset \mathbf{D}_0$. The set $\mathbf{D}_0$ defined in (6.29) is bounded and closed, thus it is compact. By the way $\mathbf{D}_0$ constructed, it is also convex. Moreover, by Theorem 6.4 the set $\mathbf{D}_0$ is invariant. Hence, by Lemma 6. 1:, there exists a proper convex, compact, invariant set $\mathbf{D}_1$. The set $\mathbf{D}$ in Lemma 6. 1: can be consider as the set $\mathbf{D}_0$ and the set $\mathbf{D}''$ can be considered as $\mathbf{D}_1$.

Similarly, we can show that if $\mathbf{D}_k \subset \mathbf{D}_{k-1}$, then $\mathbf{D}_{k+1} \subset \mathbf{D}_k$. This will conclude the proof of the theorem. □

In the following section we will put all the definitions and theorems together to introduce a systematic way of implementing the RODD-LB1 method.

Algorithm

In this section we will explain the implementation of the RODD-LB1 method. The implementation is based on the definitions and theorems which have been introduced and proved in the previous section. The RODD-LB1 algorithm consists of three main steps. These steps are given as follows:

Step 0

In this step the initial approximate reachable set $\mathbf{D}_0$ will be defined using the limits of the functions $\mathbf{f} = col\{f_i\}$. The initial set may be defined as a hypercube:

$$\mathbf{D}_0 = \{\mathbf{x} : |x_i| \le max\{|f_i(\mathbf{x})|\} \quad \mathbf{x} \in \mathbf{R}^n\} \text{ for } i = 1, 2, \ldots, n \tag{6.44}$$

The set $\mathbf{D}_0$ is a crude approximation of the initial reachable set $\mathbf{D}^*_0$.

Step 1

At each subsequent iteration, the set $\mathbf{D}_k$ needs to be updated. Consider a set of linear functions $\{h_{k,1}, h_{k,2}, \ldots, h_{k,m_k}\}$ where $h_{k,j} : \mathbf{R}^n \to \mathbf{R}$, and then define a real value $\gamma_{k,j}$ as

$$\gamma_{k,j} = max\{h_{k,j}(\mathbf{f}(\mathbf{x})) : \mathbf{x} \in \mathbf{D}_k\} \tag{6.45}$$

where $j = 1, 2, \ldots, m$ and $m = 2n$. By the EVT, the maximization in (6.45) always has a solution and $\gamma_{k,j}$ exists. This is because $\mathbf{D}_k$ is a compact set, and $h_{k,j}(\mathbf{f}(\mathbf{x}))$ is a continuous

function. Moreover, we showed inside the proof of Theorem 6.5 that $\gamma_{k,j}$ is a cauchy sequence. Knowing that $\gamma_{k,j}$ is always computable, the set $\mathbf{D}_{k+1}$ can be defined as follows

$$\mathbf{D}_{k+1} = \{\mathbf{x} \in \mathbf{D}_k : h_{k,j}(\mathbf{x}) \leq \gamma_{k,j} \quad \forall j = 1, 2, \ldots, m_k\} \tag{6.46}$$

The set $\mathbf{D}_{k+1}$ is defined in such a way that all of the linear boundaries of the set $\mathbf{D}_{k+1}$ are tangent to the surface of $\mathbf{f}(\mathbf{D}_k)$ (see the right side of Figure (6.9)). The maximization in (6.45) will guarantee this. At the point where all of the linear boundaries are tangent to the surface of $\mathbf{f}(\mathbf{D}_k)$, we will decide whether or not to add additional linear boundaries. Depending on the need for adding additional boundaries, the algorithm will either repeat step 1 or go to step 2.

If the set $\mathbf{D}_{k+1}$ shrinks sufficiently in size relative to the set $\mathbf{D}_k$, then there is no need for adding additional linear boundaries. In this case, the set $\mathbf{D}_{k+1}$ is a good approximation of $\mathbf{f}(\mathbf{D}^*_k)$, and there is no need to improve the approximation. However, if the potential approximate reachable set $\mathbf{D}_{k+1}$ does not shrink relative to $\mathbf{D}_k$, then there is a need to increase the accuracy of the set $\mathbf{D}_{k+1}$ by adding additional linear boundaries. In this case, the algorithm moves to Step 2.

Step 2

If the set $\mathbf{D}_{k+1}$ does not shrink sufficiently in size relative to the set $\mathbf{D}_k$, then the origin is either unstable or the set $\mathbf{D}_{k+1}$ is not a good approximation of the original reachable set $\mathbf{f}(\mathbf{D}^*_k)$. The former conclusion cannot be definitely made, because there always

exist errors associated with the approximation of the reachable set. For the latter conclusion, the reachable set $\mathbf{D}_k$ needs to be modified in order to be more accurate. The approximation error can be decreased by adding extra linear boundaries. The new boundaries will be added to the existing boundaries and they should be placed in locations where they will remove the most area that is not part of the true reachable set.

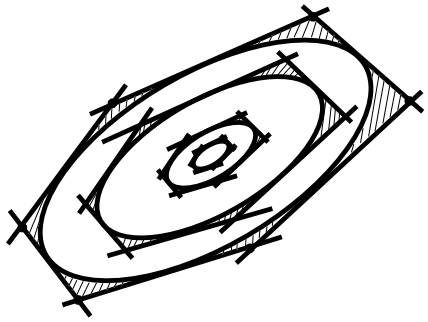

Figure (6.9) Typical Example of RODD-LB1 Step 1

In order to find the optimal location for a new boundary, the algorithm finds a unit vector $\mathbf{q}$ (which will be orthogonal to the resulting linear boundary) that maximizes the following difference:

$$\mathbf{q}^T\mathbf{x}^* - \max_{\mathbf{x} \in \mathbf{D}_k} \{\mathbf{q}^T\mathbf{f}(\mathbf{x})\} \tag{6.47}$$

where $\mathbf{x}^*$ is the point where the function $\mathbf{q}^T\mathbf{f}(\mathbf{x})$ is maximized, and its image $\mathbf{f}(\mathbf{x}^*)$ is located somewhere on the boundary of $\mathbf{f}(\mathbf{D}_k)$. The process is depicted for a simple example on the left side of Figure (6.10). (The right side of Figure (6.10) represents how $\mathbf{D}_{k+1}$ is computed if only Step1 is performed.) The dashed lines represent the contour lines of the

function $\mathbf{q}^T\mathbf{x}$. The contour line that is tangent to $\mathbf{f}(\mathbf{D}_k)$ will become the new linear boundary. The interior maximization in (6.47) will locate the $\mathbf{x}^*$ that on the next iteration of (6.24) will have the largest value of $\mathbf{q}^T\mathbf{f}(\mathbf{x})$. Thus, $\mathbf{f}(\mathbf{x}^*)$ will be on the boundary of $\mathbf{f}(\mathbf{D}_k)$ at a point that is tangent to a contour line of $\mathbf{q}^T\mathbf{x}$. The exterior optimization in (6.47) will then determine the direction $\mathbf{q}$ that produces the largest difference between $\mathbf{q}^T\mathbf{f}(\mathbf{x}^*)$ and $\mathbf{q}^T\mathbf{x}^*$, and therefore would produce the largest movement in this boundary line from $\mathbf{D}_k$ to $\mathbf{D}_{k+1}$. The new linear boundary will be located tangent to $\mathbf{f}(\mathbf{D}_k)$ at the point $\mathbf{f}(\mathbf{x}^*)$. This boundary should remove the most unwanted area from the approximation to the reachable set.

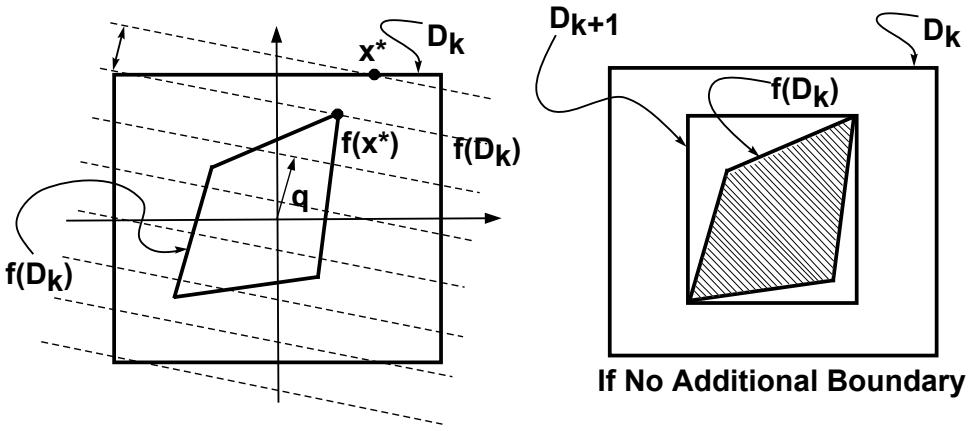

Figure (6.10) Additional Linear Boundaries

In the RODD-LB1 algorithm, the interior optimization from (6.47) is not performed over all **x**, but only over those points that represent the arguments of the optimizations that occurred in (6.45), which will be tangent points of the boundaries with $\mathbf{f}(\mathbf{D}_{k-1})$. This sim-

plifies the optimization significantly, but at the potential expense of the optimality of the new boundary. The optimization in is written as follows:

$$\mathbf{q}_j = \arg \max_{\|\mathbf{q}\| = 1} \left\{ \max_{\mathbf{x}^*} \left\{ \mathbf{q}^T \mathbf{x}^* - \max_{\mathbf{x}_i} \{ \mathbf{q}^T \mathbf{f}(\mathbf{x}_i) \ : \ i = 1, 2, \ldots, m \} \right\} \right\} \tag{6.48}$$

where $\mathbf{x}_j$, $j = 1, 2, \ldots, m$ represent the arguments of the optimizations that occurred in (6.45), which will be tangent points of the boundaries with $\mathbf{f}(\mathbf{D}_{k-1})$.

There are three reasons why limiting the interior optimization in (6.48) can limit the optimality of the new boundary. First, if $\mathbf{f}(\mathbf{D}_{k-1})$ had linear boundaries (like those shown for $\mathbf{f}(\mathbf{D}_k)$ in Figure (6.10)) then tangent points for the *m* current boundaries of $\mathbf{D}_k$ will always occur at the vertices. However, the set of tangent points for the *m* current boundaries does not include all of the vertices. Second, even if we were to include all of the vertices, $\mathbf{x}_j$, of $\mathbf{D}_k$ in the inner optimization of (6.48), there is no guarantee that the points $\mathbf{f}(\mathbf{x}_j)$ will be vertices of $\mathbf{f}(\mathbf{D}_k)$. Finally, even though $\mathbf{D}_k$ has linear boundaries, there is no guarantee that $\mathbf{f}(\mathbf{D}_k)$ will have linear boundaries. In the next chapter, we will suggest a modification to (6.48) that we have found to speed up the convergence of the algorithm.

After finding the $\mathbf{q}_j$ for $j = 1, 2, \ldots, m$ in (6.48), the RODD-LB1 algorithm selects the $\mathbf{q}_j$ that have the largest values of

$$\theta_j = \theta_{1,j} - \theta_{2,j} \tag{6.49}$$

where

$$\theta_{1,j} = \max_{\mathbf{x}} \{ \mathbf{q}_j^T \mathbf{x} \ : \mathbf{x} \in \mathbf{D}_k \} \tag{6.50}$$

$$\theta_{2,j} = \max_{\mathbf{x}} \{ \mathbf{q}_j^T \mathbf{f}(\mathbf{x}) \ : \mathbf{x} \in \mathbf{D}_k \} \tag{6.51}$$

After the new boundaries have been added, and $\mathbf{D}_{k+1}$ has been obtained (using (6.45) with the new boundaries included), the algorithm checks again to verify that there has been sufficient reduction in size from $\mathbf{D}_k$ to $\mathbf{D}_{k+1}$. If the more accurate approximate reachable set does not shrink enough, then no conclusion can be made about the stability or instability of the origin. In this case the method is inconclusive, and the algorithm stops.

If a sufficient reduction in size is obtained, then the algorithms tests to see if the size of $\mathbf{D}_{k+1}$ is essentially zero, in which case stability has been shown, and the algorithm stops. Otherwise, the algorithm continues with Step 1.

All the steps mentioned above are summarized in the flow chart given in Figure (6.11). The details of every steps will be discussed in the following section.

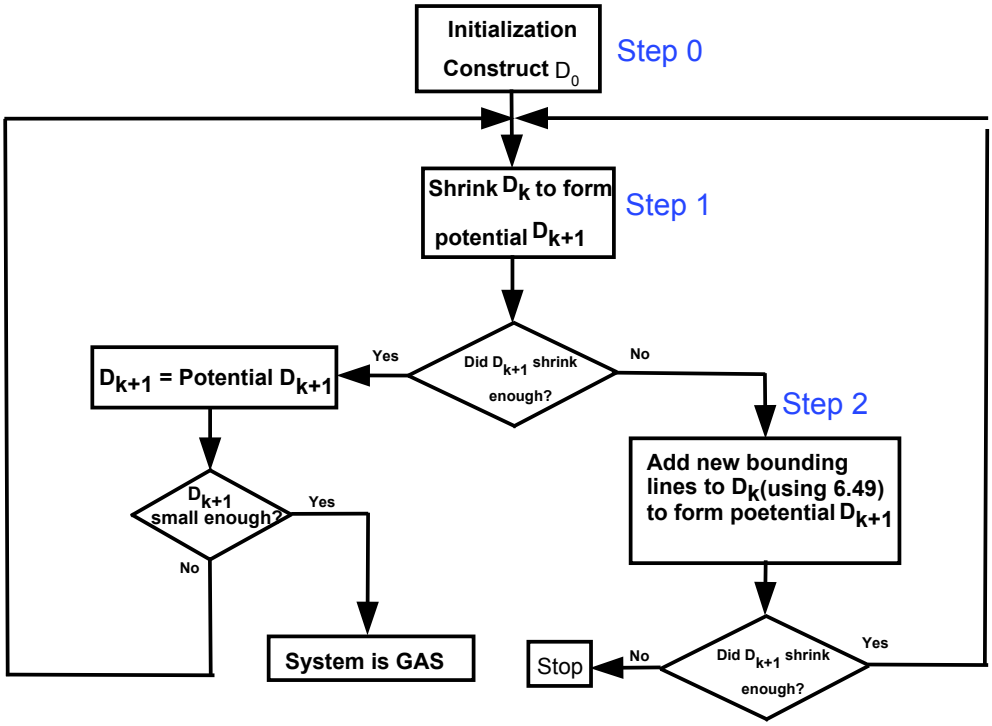

Figure (6.11) RODD-LB1 Algorithm

Let us consider again the stable network in Example 6.2, where the CMT method fails to detect the stability. Unlike the contraction mapping stability analysis technique, the RODD-LB1 method detects the GAS of the origin. Since the RODD-LB1 method is a computational technique for stability analysis, a Matlab program was written based on the algorithm. The graphical results of the simulation are given in Figure (6.12) and Figure (6.13). It is interesting to compare the phase plane representation given in Figure (6.3) and the evolution of the set $\mathbf{D}_k$ in Figure (6.12). At every time step k the set $\mathbf{D}_k$ contains all the possible system trajectories starting from different initial conditions. That is why Figure (6.3) and Figure (6.12) both have the same orientation.

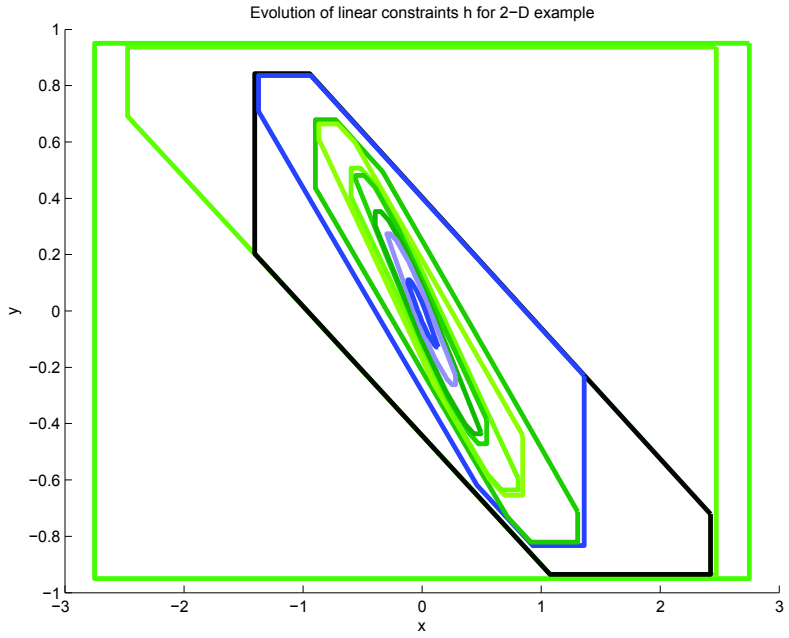

Figure (6.12) Evolution of linear constraints h for 2-D example

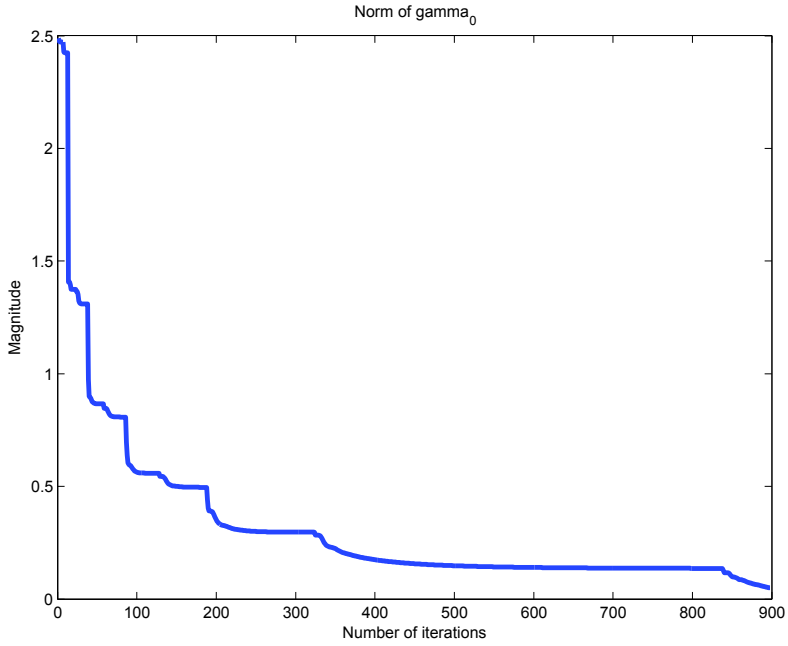

Figure (6.13) Convergence

Conclusion

Two methods of stability analysis have been discussed in this chapter. The first method is based on CMT, and the second method is based on reduction of dissipativity domain. We showed that the first method was not as useful, because the stability criterion was very conservative. However, the RODD-LB1 method solved this problem by expanding the space of stable parameters. Even though the second method is an improvement, it is still not practical for large systems. The main drawback of the second method is slow convergence.

In the next chapter we will propose two methods to overcome the problem of slow convergence. It will be shown that an extension to the current method increases the methods efficiency. Also, the quadratic approximation of reachable sets will be discussed in detail as a faster stability analysis method.

7 EXTENSIONS OF THE METHOD OF REDUCTION OF DISSIPATIVITY DOMAIN

In the previous chapter, two methods of stability analysis were developed for the following dynamical system

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)) \tag{7.1}$$

where $\mathbf{f} : \mathbf{R}^n \to \mathbf{R}^n$ and $\mathbf{x} \in \mathbf{R}^n$. The CMT method and the RODD-LB1 method can be used to analyze the stability of the origin of system (7.1). The CMT method is usually more conservative than the RODD-LB1 method. In other words, the RODD-LB1 method usually identifies a larger space of stable parameters for a given RNN compared to the CMT method. Although the RODD-LB1 method derives a less conservative stability criterion, the

method suffers from a slow rate of convergence. The slow convergence for large networks makes the method almost impractical. In this chapter, we will address this problem and introduce three alternative methods that fix the drawbacks of the RODD-LB1 method.

In the first section of the chapter, we will propose an improved version of the algorithm explained in the previous chapter. RODD-LB2, which is an improved version of RODD-LB1, is more efficient than the existing algorithm. In the second section of the chapter, we will introduce the RODD-EB method as an efficient algorithm for dynamical systems with elliptical reachable sets. Finally, we will introduce RODD-Hybrid as the most efficient algorithm, which is a combination of RODD-LB2 and RODD-EB.

The chapter will concluded with an explanation of how the shape of the reachable sets will affect the efficiency of the RODD methods.

Global Asymptotic Stability Using RODD-LB2 Method

In this section, we will introduce an extension to RODD-LB1, called RODD-LB2, that produces a more accurate approximation to the reachable set. The efficiency of the RODD-LB2 method compared to the RODD-LB1 method will be verified through numerical examples.

RODD-LB2

A potential area for improvement in the RODD-LB1 method is the maximization in Step 2 (explained in chapter 6). The inner maximization given in (6.48) is not searching for the optimum $\mathbf{x}$ over the entire domain $\mathbf{D}_k$, but only among points that are the arguments of the maximizations in (6.45). Limiting the search space enables the maximization to be done quickly, but it may fail to produce an optimal boundary, with maximum reduction in

the size of the approximate reachable set. Our objective is to increase somewhat the search space. This will require a somewhat larger computational burden at each iteration, but the improved boundaries for the approximate reachable set may allow the algorithm to converge in fewer iterations. In the next section we will test this on several sample problems.

In order to improve the inner optimization of (6.48), instead of searching for the optimal $\mathbf{x}$ only among the set of all points provided by (6.45), we will add all of the vertices of $\mathbf{D}_k$ to the existing search space. We found out that providing more points (vertices of the $\mathbf{D}_k$) make RODD-LB1 more efficient and even faster in higher dimensions.

Based on the discussion above, the optimization in (6.48) may be reformulated as follows

$$\mathbf{q}_j = \arg \max_{\|\mathbf{q}\| = 1} \left\{ \mathbf{q}^T \mathbf{x}_j - \max_{\mathbf{x}_i \in \mathbf{\Omega}} \{ \mathbf{q}^T \mathbf{f}(\mathbf{x}_i) \} \right\} \tag{7.2}$$

where $\mathbf{\Omega}$ is a set containing all the vertices of the set $\mathbf{D}_k$ and where $\mathbf{x}_j \in \mathbf{\Omega}$.

The concept of RODD-LB2 is illustrated with the 2-D example shown in Figure (7.1).

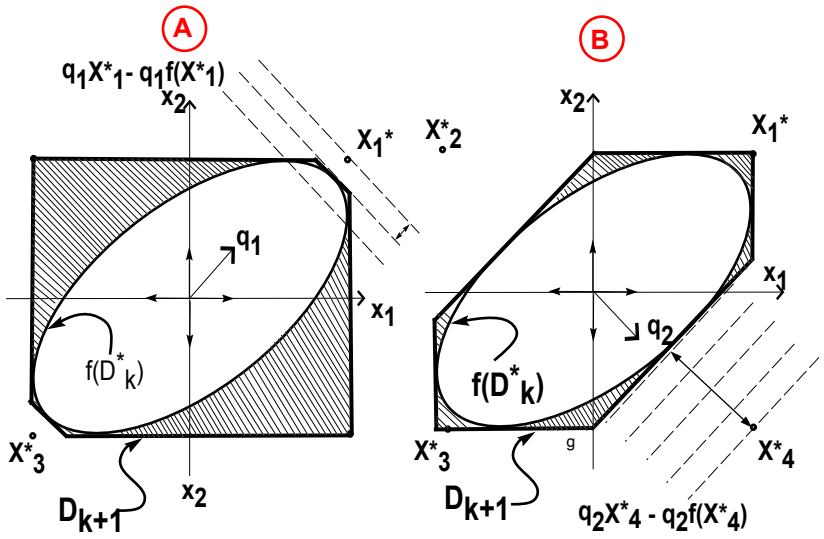

Figure (7.1) Operation of Step 2

The left graph (A) used the RODD-LB1 method to approximate the reachable set, whereas the right graph (B) used the RODD-LB2 method. It can be observed from the right graph that providing more search points for the inner maximization given in (7.2) results in removing more unwanted area from the approximate reachable set. The vertices at the top right and bottom left of $\mathbf{D}_k$ were the only vertices that were arguments of the maximizations in (6.45). However, if they were the only $\mathbf{x}_j$ used in the inner optimization of (6.48), then very little reduction in the size of the approximate reachable set is obtained. By including all of the vertices of $\mathbf{D}_k$, a more accurate reachable set is achieved.

RODD-LB2 Algorithm

The steps of RODD-LB2 are identical to the steps of RODD_LB1, except for the difference in step 2. The flow chart of the RODD-LB2 method is shown in Figure (7.2).

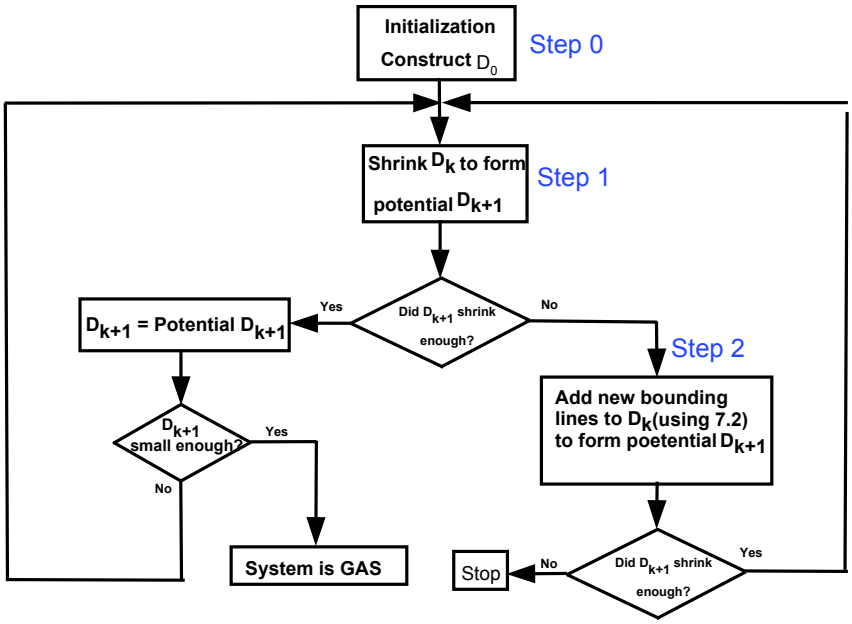

Figure (7.2) RODD-LB2 Algorithm

In the following section, the efficiency of RODD-LB2 will be compared with RODD-LB1 through numerical example.

Example 7.1 Consider Example 6.1, which is repeated here

$$\mathbf{x}(k+1) = \mathbf{W}\tanh(\mathbf{x}(k)) \text{ where } \mathbf{W} = \begin{bmatrix} 1.8 & 0.95 \\ -0.95 & 0 \end{bmatrix} \tag{7.3}$$

We know that the origin of system (7.3) is GAS, but as we showed in the previous chapter, the CMT method failed to determine stability of the origin. This is because the norm $\|\mathbf{W}\| = 2.208 > 1$ which violates the condition for the CMT method. The RODD-LB1 method can prove the stability of the origin for the above example, but the algorithm suffers from a slow rate of convergence. However, the RODD-LB2 method introduced in this chapter overcomes the slow rate of convergence. In order to verify the efficiency of the RODD-LB2 method, the above example was tested under same conditions as in the previ-

ous chapter with the RODD-LB2 method. The simulation results are given in Table (7.1). We have observed that the RODD-LB2 method is 55% to 75% faster than the RODD-LB1 method. The modifications in the optimization in step 2 make the algorithm more efficient by more accurately approximating the reachable set $\mathbf{D}_k^*$. The simulation results are given graphically in Figure (7.3) and Figure (7.4).

	RODD-LB1	RODD-LB2
Number of iterations	898	598
Time(sec)	203	103
Number of edges of $\mathbf{D}_k$	24	16

Table (7.1) Comparison of RODD methods

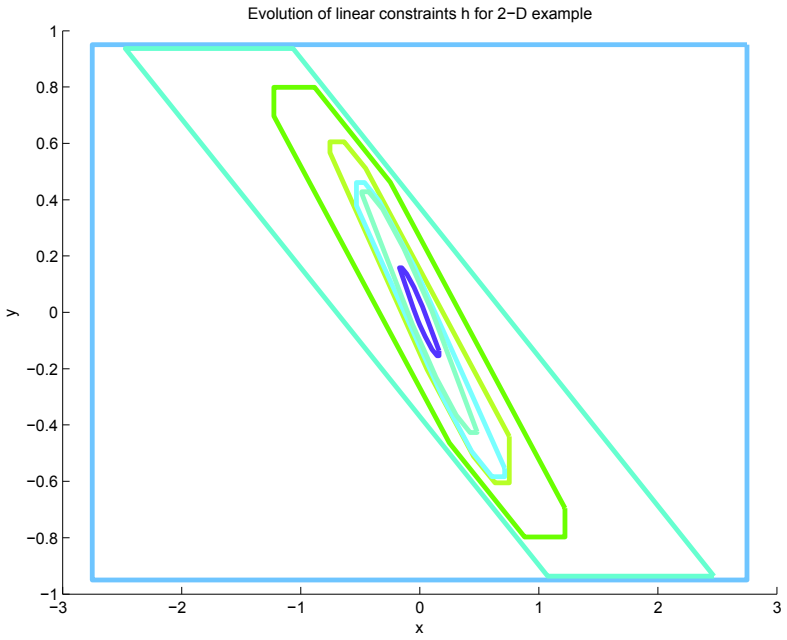

Figure (7.3) Evolution of linear constraints h for 2-D example (RODD-LB2)

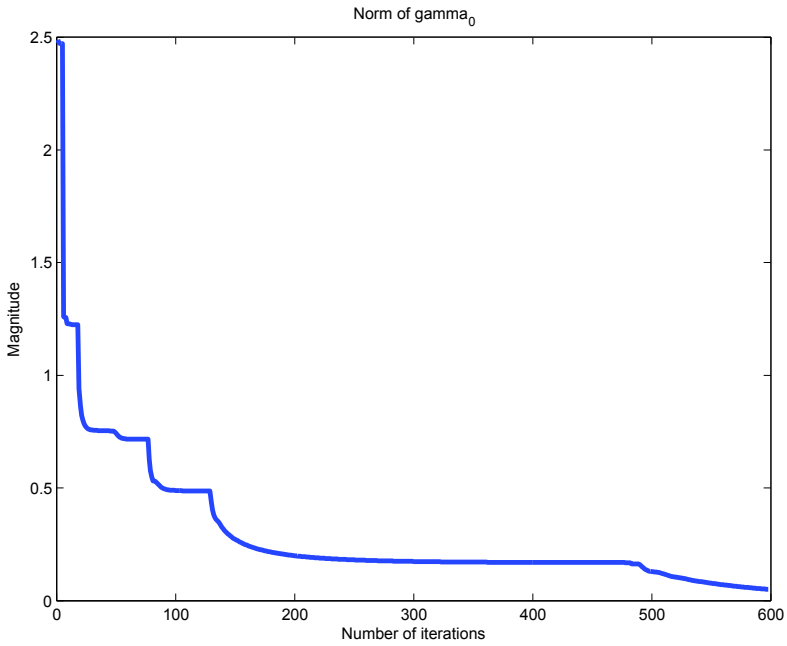

Figure (7.4) Convergence with RODD-LB2 method

The accuracy and efficiency of RODD-LB1 and RODD-LB2 depend not only on the optimization, but also on the choices of the set of functions $\{h_{k,j}\}$. The RODD-LB1 and RODD-LB2 methods are using a set of linear functions to approximate the reachable sets. However, $\{h_{k,j}\}$ can be chosen to be an arbitrary set of continuous functions. Consequently, the RODD-LB1 and RODD-LB2 methods could also works with other choices of $\{h_{k,j}\}$, i.e. quadratic functions, cubic functions, etc. In the following section we will choose a set of nonlinear functions $\{h_{k,j}\}$ for a faster approximation of reachable sets. It will be shown that the RODD method with a set of quadratic functions performs faster than to the RODD method with linear functions. This is only true for systems with convex reachable sets.

Global Asymptotic Stability Using RODD-EB Method

The RODD-EB method is a stability analysis technique based on the method of reduction of dissipativity domain with quadratic functions for $\{h_{k,j}\}$. The main difference between RODD-LB (RODD-LB1 and RODD-LB2) and RODD-EB is the choice for $\{h_{k,j}\}$. The RODD-LB methods use linear functions to approximate reachable sets whereas RODD-EB uses quadratic functions to approximate reachable sets.

In this section, we will show experimentally that the RODD-EB method is one of the most efficient of the algorithms we have studied in detecting stability for stable systems with elliptical reachable sets. There are many systems whose reachable sets are convex. For example, certain types of RNNs fall into this category. The stability analysis of the model reference adaptive control system, with RNN emulator and RNN controller, can be efficiently be investigated by the RODD-EB method. This will be discussed in Chapter 9.

In the following section, we will explain the RODD-EB method in detail.

RODD-EB

Since the RODD-EB method is based on the elliptical approximation of reachable sets, we will start the section with the definition of the ellipse. A closed ellipsoidal set in n-dimensional Euclidean space can be defined by the following inequality[Mosh05]

$$\Xi = \{\mathbf{x} \in \mathbf{R}^n \mid (\mathbf{x}-\mathbf{c})^T \mathbf{E} (\mathbf{x}-\mathbf{c}) \le 1\} \tag{7.4}$$

where $\mathbf{c} \in \mathbf{R}^n$ is the center of the ellipse Ξ, and $\mathbf{E} > 0$. The volume of Ξ is given by

$$Vol(\mathbf{E}) = \frac{v_0}{\sqrt{det(\mathbf{E})}} = v_0 det(\mathbf{E}^{-1})^{\frac{1}{2}} \tag{7.5}$$

where v_0 is the volume of the unit hypersphere in dimension n. The volume of an ellipse is important, since it is a measure of the size of an ellipse. However, the calculation of volume in high dimensions is very expensive. For this reason, we use another method to compare the size of $\mathbf{D}_k$ and $\mathbf{D}_{k+1}$. This method will be explained in the algorithm. Next, we will explain the concept of the RODD-EB method.

RODD-EB is a stability analysis technique based on the reduction of dissipativity domain. Similar to the RODD-LB1 and RODD-LB2 methods, the RODD-EB method is based on the concept that the approximate reachable set $\mathbf{D}_{k+1}$ must always contain the reachable set $\mathbf{f}(\mathbf{D}^*_k)$. In the RODD-EB method, the set of functions $\{h_{k,j}\}$ are quadratic, and the approximate reachable sets $\mathbf{D}_k$ are ellipses.

The RODD-EB algorithm starts with $\mathbf{R}^n$ as a set containing arbitrary initial conditions and derives the set $\mathbf{D}_0$ as an approximation to the set $\mathbf{f}(\mathbf{R}^n)$. In this method, the set $\mathbf{D}_0$ is an ellipse $(\mathbf{E}_0, 0)$ that contains $\mathbf{f}(\mathbf{R}^n)$. Without loss of generality, we can centralize the initial ellipse at the origin, because it is always assumed that the system under investigation has an equilibrium point at the origin. If the origin is not the equilibrium point, then the system can be modified so that the origin is the equilibrium point.

The algorithm will always calculate the set $\mathbf{D}_{k+1}$ from the set $\mathbf{D}_k$ by solving the optimization problem explained in the next section. The approximate reachable set at time step $k+1$ ($\mathbf{D}_{k+1}$) will be compared against the approximate reachable set at the previous time step k ($\mathbf{D}_k$) to check for a reduction in size. Since $\mathbf{D}_{k+1}$ is an approximation for

$\mathbf{f}(\mathbf{D}^*_k)$, the continuous shrinkage of the approximate reachable set to $\{\mathbf{0}\}$ is an indication of the global asymptotic stability of the origin. The enlargement of the reachable set $\mathbf{D}_k$, may be an indication of instability. However, it is possible that the equilibrium point of a system is GAS, even though the approximate reachable set does not shrink. The enlargement could be due an approximation error between the actual reachable set and the approximate reachable set.

RODD-EB Algorithm

In this section, we will explain the implementation of the RODD-EB method. The RODD-EB algorithm can be divided into three steps. These steps are given as follows:

Step 0

To initialize the RODD-EB method, a finite number of points in $\mathbf{R}^n$ need to be generated. These points will be updated one time step using (7.1). Then, an initial bounding ellipse $(\mathbf{E}_0, \mathbf{0})$ is computed using the method in [Mosh05] which finds the minimum volume ellipse enclosing a finite number of points. Since the method of finding the minimum volume bounding ellipse is a numerical method so there always exists numerical error. Because of potential numerical errors, the initial ellipse $(\mathbf{E}_0, \mathbf{0})$ is enlarged by a factor greater than 1. This will guarantee that $(\mathbf{E}_0, \mathbf{0})$ contains the entire set $\mathbf{f}(\mathbf{R}^n)$. In this research, we used the factor 1.2 (20% larger than the optimal ellipse) for all the simulations. Figure (7.5) shows the minimum bounding ellipse and the enlarged bounding ellipse (20% larger).

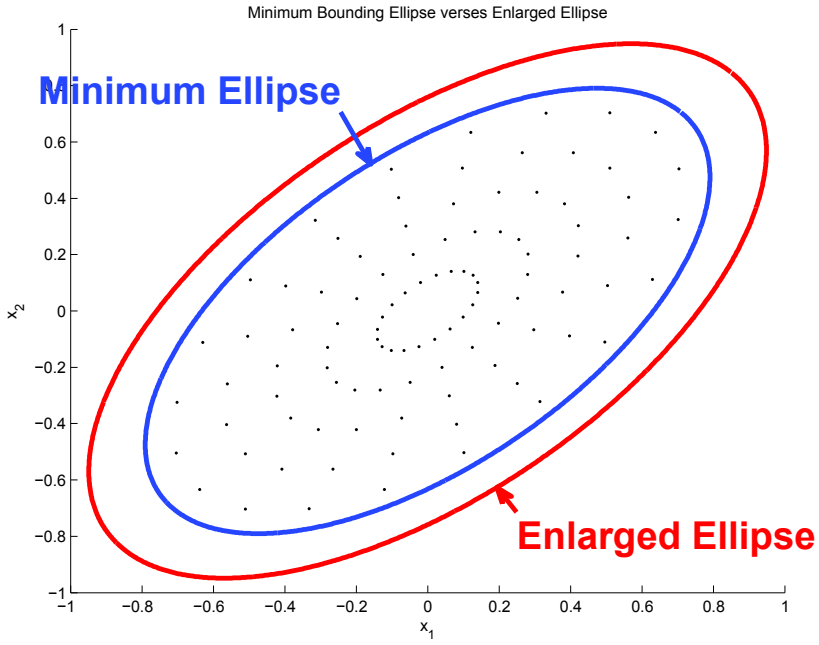

Figure (7.5) Minimum Bounded & Enlarged Bounded Ellipse

Step 1

In this step, the set $\mathbf{D}_{k+1}$ will be constructed from the set $\mathbf{D}_k$. The update involves the following maximization process:

$$\begin{aligned} \gamma_{k+1} &= \max_{\mathbf{x}} \{ \mathbf{f}(\mathbf{x}(k))^T \mathbf{E}_k \mathbf{f}(\mathbf{x}(k)) \;\; | \mathbf{x}(k) \in \mathbf{D}_k \} \\ \mathbf{E}_{k+1} &= \frac{\mathbf{E}_k}{\gamma_{k+1}} \end{aligned} \tag{7.6}$$

where $\mathbf{D}_k = \{ \mathbf{x} \in \mathbf{R}^n \mid \mathbf{x}(k)^T \mathbf{E}_k \mathbf{x}((k) \le 1) \}$. The maximization in (7.6) searches for an optimum point $\mathbf{x}^*$ to maximize $\mathbf{x}(k+1)^T \mathbf{E}_k \mathbf{x}(k+1)$ over the set $\mathbf{D}_k$. This maximization has a solution and γ_{k+1} is computable, because the set $\mathbf{D}_k$ is compact (bounded and closed), and the function $\mathbf{f}$ is continuous.

In order to be consistent with the definition of $\mathbf{D}_k$, we normalize $\mathbf{E}_k$ by γ_{k+1}. Then, if $\gamma_{k+1} < 1$, the set $\mathbf{D}_{k+1}$ shrinks compared to $\mathbf{D}_k$. The set $\mathbf{D}_k$ derived in this step is a potential $\mathbf{D}_k$, because if $\mathbf{D}_{k+1}$ does not shrinking compared to $\mathbf{D}_k$, then $\mathbf{D}_k$ needs to be changed.

The shrinkage of $\mathbf{D}_{k+1}$ compared to $\mathbf{D}_k$ means that all the system trajectories at time step $k+1$ stay inside the set $\mathbf{D}_{k+1} \subset \mathbf{D}_k$. This means that the set $\mathbf{D}_{k+1}$ is a good approximation of the set $\mathbf{f}(\mathbf{D}^*_k)$, and there is no need for a change of orientation of the ellipse $\mathbf{E}_k$. If $\mathbf{D}_{k+1}$ does not shrink compared to $\mathbf{D}_k$, then the algorithm moves to step 2. This is the case when the orientation of $\mathbf{E}_k$ needs to be changed.

Figure (7.6) illustrates a typical 2-D example of a system with GAS equilibrium point, for which $\mathbf{f}(\mathbf{D}^*_k) \subset \mathbf{D}_{k+1}$. If the equilibrium point of a system is GAS, and if $\mathbf{E}_k$ is oriented correctly, then $\mathbf{x}^T(k)\mathbf{E}_k\mathbf{x}(k) \le \gamma_k$ is a good approximation of the reachable set. For stable linear systems, there always exists an elliptical reachable set $\mathbf{x}^T(k)\mathbf{E}_k\mathbf{x}(k) \le \gamma_k$. For nonlinear systems with stable equilibrium point, a correct choice for $\mathbf{E}_k$ can make $\mathbf{x}^T(k)\mathbf{E}_k\mathbf{x}(k) \le \gamma_k$ a good approximation to the reachable set.

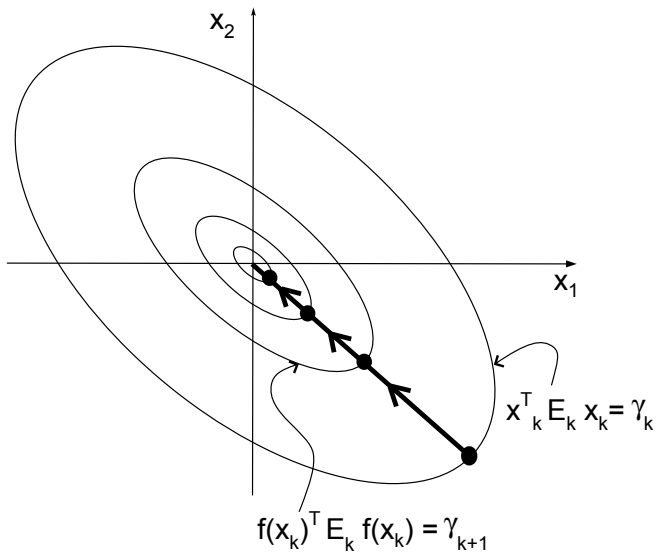

Figure (7.6) Typical example of system trajectory in step 1

Step 2

If the set $\mathbf{D}_{k+1}$ does not shrink sufficiently in size relative to the set $\mathbf{D}_k$, then the equilibrium point is either unstable or the set $\mathbf{D}_{k+1}$ is not a good approximation of $\mathbf{f}(\mathbf{D}^*_k)$. The former reason is not something we can determine, because there always exists approximation errors in the RODD-EB method. The latter reason could be possible, because the ellipse $\mathbf{E}_k$ may not be optimally orientated. In this case, the orientation of the ellipse $\mathbf{E}_k$ needs to be changed, in order to make $\mathbf{D}_{k+1}$ a better approximation of $\mathbf{f}(\mathbf{D}^*_k)$. Define

$$\Delta V = \mathbf{x}^T\mathbf{E}\mathbf{x} - \mathbf{f}(\mathbf{x})^T\mathbf{E}\mathbf{f}(\mathbf{x}) \tag{7.7}$$

Ideally, we would like to find an ellipse $\mathbf{E}$ that would maximize ΔV for all $\mathbf{x}$. Practically, this is not possible, because for every $\mathbf{x}$ we will get a different $\mathbf{E}$. An optimal solution to

this problem is to maximize the worst case over $\mathbf{x}$ (the worst case over $\mathbf{x}$ is the case when the $\mathbf{x}^T\mathbf{E}\mathbf{x} - \mathbf{f}(\mathbf{x})^T\mathbf{E}\mathbf{f}(\mathbf{x})$ is minimum for fixed $\mathbf{E}$). This can be formulated as follows:

$$\max_{\mathbf{E}} \quad \left\{ \min_{\mathbf{x} \in \mathbf{D}_k} \mathbf{x}^T\mathbf{E}\mathbf{x} - \mathbf{f}(\mathbf{x})^T\mathbf{E}\mathbf{f}(\mathbf{x}) \right\} \tag{7.8}$$

If the maximum of the worst case over $\mathbf{x}$ is positive, then it is guaranteed that the set $\mathbf{D}_{k+1}$, which is constructed based on the new $\mathbf{E}$, contains all the system trajectories at time step $k+1$, and the new $\mathbf{E}$ will be in an optimal location. In this case, $\mathbf{D}_{k+1}$ is a good approximation for $\mathbf{f}(\mathbf{D}^*_k)$ and the algorithm will be continued with new $\mathbf{E}$ in step 1.

Figure (7.7) explains the max-min optimization in (7.8) graphically. The dashed ellipses are contour lines of the potential $\mathbf{E}$. For the potential $\mathbf{E}$, $\Delta V > 0$ for the point $\mathbf{x}_1$ and $\Delta V < 0$ for the point $\mathbf{x}_2$. The contour lines for the optimal $\mathbf{E}$ are shown by the solid lines, where for all $\mathbf{x} \in \mathbf{D}_k$ the $\Delta V > 0$.

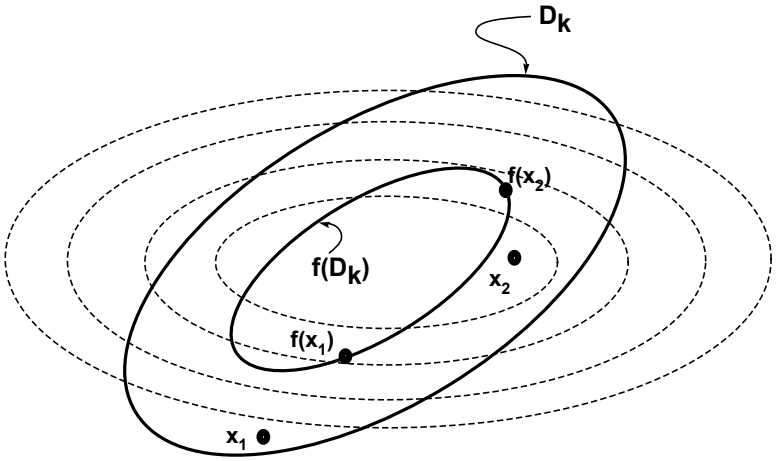

Figure (7.7) Optimal Orientation of **E**

Figure (7.8) shows the direction field for a stable 2-D example. In this figure, the solid lines represent an approximate reachable set that is not a good approximation of the reachable set, and the dashed ellipses represent a good approximation of the reachable set. The solid ellipses are not a good approximation of the reachable set, because ΔV is not positive for all $\mathbf{x} \in \mathbf{D}_k$. By inspection of Figure (7.8), it can be observed that some trajectories are going out of the solid ellipses. However the dashed ellipses represent a good approximation of the reachable set because ΔV is positive for all $\mathbf{x} \in \mathbf{D}_k$. (All trajectories are moving inside the dashed ellipses.)

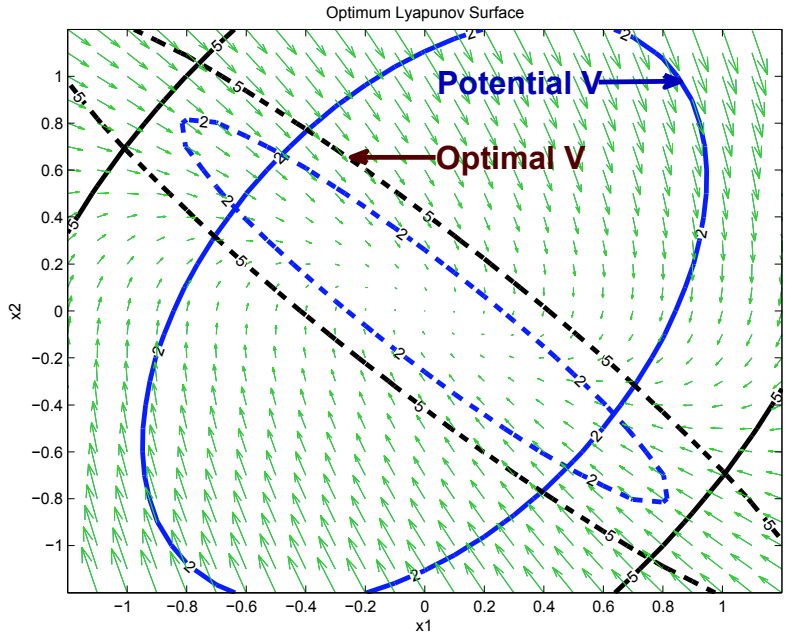

Figure (7.8) Optimal V versus Potential V

The method for finding the optimal ellipse $\mathbf{E}_k$ can be formulated as follows:

$$\begin{aligned}
\mathbf{E}_k &= \max_{\mathbf{E}} \left\{ \min_{\mathbf{x}} \{ \mathbf{x}^T \mathbf{E} \mathbf{x} - \mathbf{f}(\mathbf{x})^T \mathbf{E} \mathbf{f}(\mathbf{x}) : \mathbf{x}^T \mathbf{E}_{k-1} \mathbf{x} \le 1 \} \begin{pmatrix} min(eig(\mathbf{E})) > 0 \\ max(eig(\mathbf{E})) < 1 \end{pmatrix} \right\} \\
\gamma_k &= \max_{\mathbf{x}} \{ \mathbf{f}(\mathbf{x})^T \mathbf{E}_k \mathbf{f}(\mathbf{x}) : \mathbf{x}^T \mathbf{E}_{k-1} \mathbf{x} \le 1 \} \\
\mathbf{E}_k &= \frac{\mathbf{E}_k}{\gamma_k}
\end{aligned} \tag{7.9}$$

The inner minimization in (7.9) minimizes ΔV over $\mathbf{x} \in \mathbf{D}_k$ for fixed $\mathbf{E}$ whereas the outer maximization finds the optimal $\mathbf{E}$ to maximize the minimum value of ΔV when $\mathbf{x}$ is fixed. The inner minimization is constrained by $\mathbf{x}^T \mathbf{E}_{k-1} \mathbf{x} \le 1$, because $\mathbf{x}$ needs to be picked from the set $\mathbf{D}_k$, and the outer maximization is constrained by $min(eig(\mathbf{E})) > 0$ and $max(eig(\mathbf{E})) < 1$. The minimum eigenvalue of $\mathbf{E}$ is forced to be positive, because $\mathbf{E}$

needs to be positive definite, and the maximum eigenvalue of $\mathbf{E}$ is forced to be less than 1 to normalize $\mathbf{E}$ and to make sure that ΔV will not go to infinity. As in step 1, we always normalize $\mathbf{E}_k$ by γ_k. If $\gamma_k < 1$, then $\mathbf{D}_k$ has shrunk relative to $\mathbf{D}_{k-1}$. If $\mathbf{D}_k$ does not shrink enough, then no conclusion can be made about the stability or instability of the origin. In this case the method is inconclusive, and the algorithm stops.

If a sufficient reduction is obtained, then the algorithm tests to see if the size of $\mathbf{D}_k$ is essentially zero, in which case stability has been shown, and the algorithm stops. Otherwise, the algorithm continues with step 1.

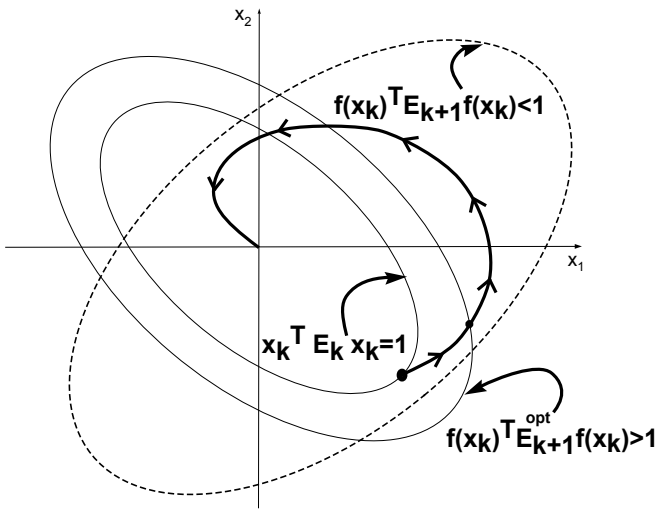

Figure (7.9) Typical example of step 2

Figure (7.9) shows a typical example of step 2. This figure shows that the potential set $\mathbf{D}^{opt}{}_{k+1}$ ($\mathbf{E}^{opt}{}_{k+1}$) does not shrink relative to the set $\mathbf{D}_k$ ($\mathbf{E}_k$). This is because $\mathbf{E}_k$ is not correctly oriented. However, by solving the optimization given in (7.9), the optimal

$\mathbf{E}_{k+1}$ can be calculated. The $\mathbf{E}_{k+1}$ is correctly oriented, because it contains all the system trajectories at time step $k+1$. The $\mathbf{E}_{k+1}$ is shown with dashed lines in Figure (7.9).

The steps of the RODD-EB method are summarized in the flow chart in Figure (7.10).

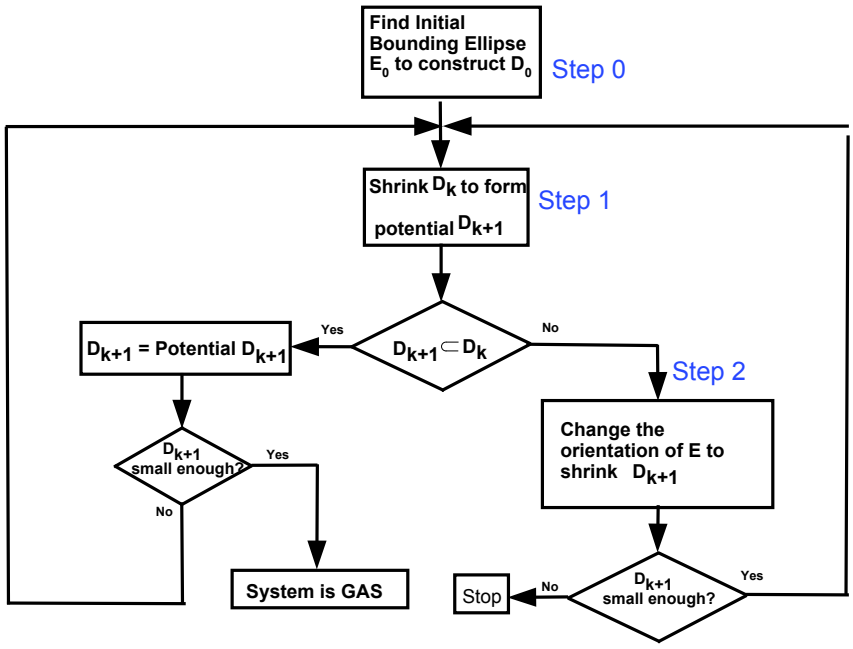

Figure (7.10) RODD-EB Algorithm

Global Asymptotic Stability Using RODD-Hybrid Method

The RODD-Hybrid method is a stability analysis technique based on the method of reduction of dissipativity domain with both linear and quadratic functions $\{h_{k,j}\}$. The RODD-Hybrid method uses a combination of linear and quadratic functions to approximate reachable sets.

The following section will explain the RODD-Hybrid method in detail.

RODD-Hybrid

The RODD-LB1 and RODD-LB2 methods approximate reachable sets with a set of linear boundaries. These methods are guaranteed to detect global asymptotic stability of any stable dynamical systems, if a sufficient number of linear boundaries issued, because any reachable set can be well approximated with enough linear boundaries. The main drawback of these methods is the slow rate of convergence, because it may require many linear boundaries to adequately approximate the reachable set, and adding linear boundaries increases the number of calculations. On the other hand, the RODD-EB method is an algorithm with a fast rate of convergence, if the reachable set is approximately quadratic. This is generally true near the equilibrium point, but may not be true at the early iterations of the algorithm. The flexibility of linear boundaries might be useful at certain stages of the algorithm, while the efficiency of elliptical boundaries could provide better performance at other stages.

The need for an accurate and fast algorithm for proving global asymptotic stability of dynamical systems leads us to develop a new algorithm based on a combination of the RODD-LB2 and RODD-EB. RODD-Hybrid is an efficient algorithm which combines RODD-LB2 and RODD-EB by switching them at various iterations of the algorithm.

In the following section, we will explain the RODD-Hybrid algorithm in detail.

RODD-Hybrid Algorithm

The main goal in the RODD-Hybrid method is to get an accurate approximation of reachable sets with the fastest rate of convergence. In order to accomplish this goal, the RODD-Hybrid algorithm is divided into three main modes, RODD-LB2, RODD-EB and

transition mode. The RODD-Hybrid method could start either with RODD-LB2 or RODD-EB mode. Assume that the algorithm starts in the RODD-EB mode (in either case, the initialization would remain the same, see the step 0 of RODD-LB2 or RODD-EB). After the initial phase, the algorithm continues in RODD-EB mode and switches to the RODD-LB2 mode when certain conditions are satisfied. The preparation of the algorithm to switch from one mode to the other mode occurs in the transition mode. The algorithm alternates between the two modes until the algorithm stops. The detail of each mode is given in the flowering sections:

RODD-EB Mode

Consider that the RODD-Hybrid method starts with the elliptical approximation of the initial reachable set ($\mathbf{f}(\mathbf{R}^n)$). In this case, the RODD-Hybrid algorithm uses (7.6) to derive $\mathbf{D}_{k+1}$. The shrinkage of $\mathbf{D}_{k+1}$ compared to $\mathbf{D}_k$ means that all the system trajectories at time step $k+1$ stay inside the set $\mathbf{D}_{k+1} \subset \mathbf{D}_k$. This means that the set $\mathbf{D}_{k+1}$ is a good approximation of the set $\mathbf{f}(\mathbf{D}^*_k)$. If $\mathbf{D}_{k+1}$ does not shrink compared to $\mathbf{D}_k$, then the algorithm uses (7.9) to find a better approximation of the reachable set. Unlike the RODD-EB algorithm, the RODD-Hybrid algorithm only allows one re-orientation of $\mathbf{D}_k$. If the re-oriented $\mathbf{D}_k$ does not make $\mathbf{D}_{k+1}$ shrink compared to $\mathbf{D}_k$, then the algorithm switches to RODD-LB2. The transition from RODD-EB to the RODD-LB2 will be explained in the transition mode section.

Figure (7.11) graphically illustrates the RODD-EB mode of the RODD-Hybrid algorithm. If the set $\mathbf{D}_{k+3}$ does not shrink compared to $\mathbf{D}_{k+4}$, then the algorithm switches to the RODD-LB2 mode.

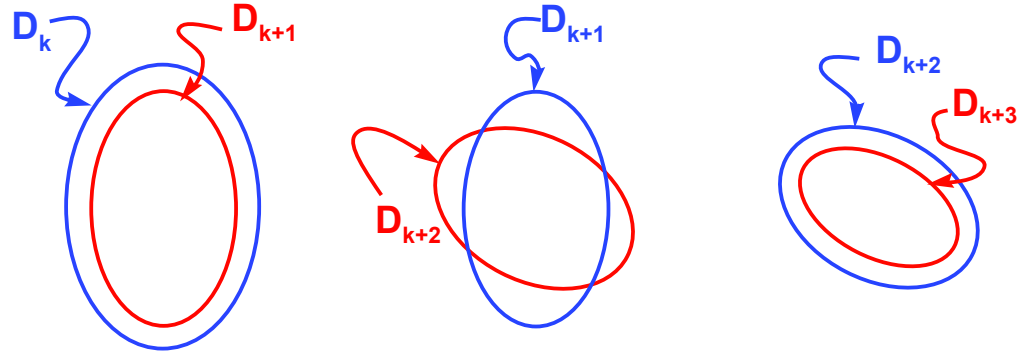

Figure (7.11) RODD-EB Mode of RODD-Hybrid Algorithm

RODD-LB2 Mode

The RODD-LB2 mode of RODD-Hybrid will take over when one re-orientation of the set $\mathbf{D}_k$ does not make $\mathbf{D}_{k+1}$ to shrink compared to $\mathbf{D}_k$. In this mode, the set $\mathbf{D}_k$ from the RODD-EB mode will be approximated by a set of linear boundaries. These set of linear boundaries will be calculated to enclose the ellipsoid $\mathbf{E}_k$ ($\mathbf{D}_k$).

The bounds of the optimum bounding polytope will be derived through the following optimization:

$$\gamma_k = max\{\mathbf{v}^T\mathbf{x} : \mathbf{x}^T\mathbf{E}_k\mathbf{x} < 1\} \tag{7.10}$$

where $\mathbf{v} = \{v_1, v_2, \ldots, v_n\}$ are the eigenvectors of $\mathbf{E}_k$. The sides of the bounding polytope are orthogonal to the eigen vectors $\mathbf{v}$, and the optimization (7.10) derives the bound such that the sides will be tangent to $\mathbf{E}_k$. From this point, the reachable set will be approximated

by the bounding polytope $\mathbf{D}_k$.

Figure (7.12) shows the bounding polygon ($n = 2$) derived through the optimization given in (7.10).

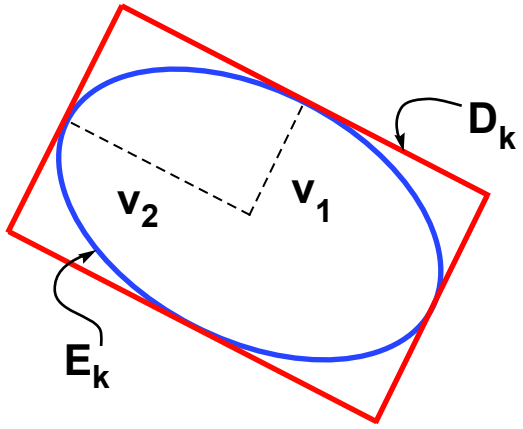

Figure (7.12) Bounding Polygon

In this mode, $\mathbf{D}_{k+1}$ is derived through (6.45) and (6.46). If the set $\mathbf{D}_{k+1}$ shrinks in size relative to the set $\mathbf{D}_k$, then there is no need for adding additional linear boundaries. However, if the potential approximated reachable set $\mathbf{D}_{k+1}$ does not shrink relative to $\mathbf{D}_k$, then there is a need to add additional linear boundaries. The additional linear boundaries will be calculated through the optimization given in (6.48). If the additional linear boundaries do not make $\mathbf{D}_{k+1}$ shrink relative to $\mathbf{D}_k$ then algorithm is inconclusive and will stop. Otherwise the algorithm will continue until the number of additional linear boundaries exceeds the maximum allowable number. According to several experiments, we found that 2 times the initial dimension is a working upper bound for the maximum number of linear boundaries. If $\mathbf{D}_k$ does not get small enough, and the algorithm exceeds the maximum

number of linear boundaries, then the algorithm will switch to the RODD-EB mode. The transition from RODD-LB2 to RODD-EB will be explained in the transition mode section.

Figure (7.13) graphically illustrates the RODD-LB2 mode of the RODD-Hybrid Algorithm. If the set $\mathbf{D}_{k+2}$ does not shrink compared to $\mathbf{D}_{k+1}$ then the algorithm will switch to the RODD-EB mode using the bounding ellipse $\mathbf{E}_{k+1}$.

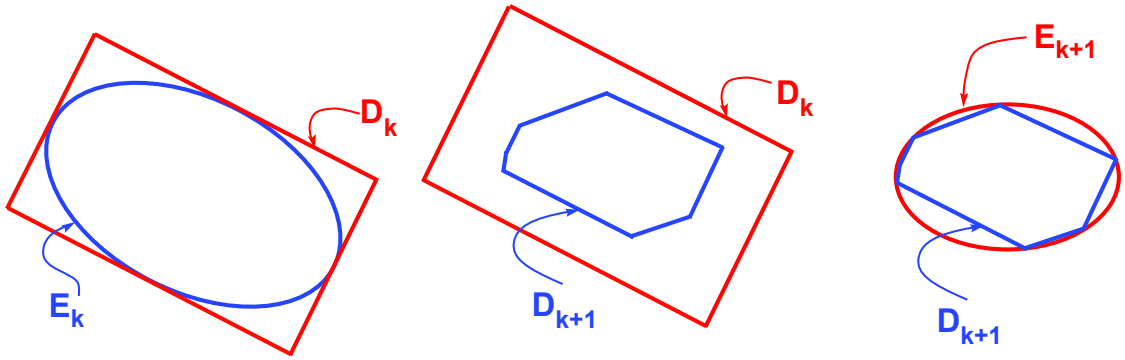

Figure (7.13) RODD-LB2 Mode of RODD-Hybrid Algorithm

Transition Mode

The transition mode is the mode when the algorithm switches from one mode to the other mode. The possible transition modes are as follows:

RODD-EB to RODD-LB2: This is the case when the algorithm switches from the quadratic approximation (RODD-EB) to the linear approximation (RODD-LB2). In this case, the last $\mathbf{D}_k$ in the RODD-EB method will be used to derive the first $\mathbf{D}_k$ in the RODD-LB2 mode. After that, the set $\mathbf{D}_{k+1}$ will be derived from $\mathbf{D}_k$ using the RODD-LB2 mode.

RODD-LB2 to RODD-EB: This is the case when the algorithm switches from the linear approximation to the quadratic approximation of reachable sets. In this case a bound-

ing ellipse will be calculated to enclose the final set $\mathbf{D}_k$ in the RODD-LB2 mode. The calculation of the bounding ellipse in higher dimensions using a convex hull is very expensive. We proposed a less expensive method, which is described below. The main idea of this approach is to find a set of points on the boundary of $\mathbf{D}_k$ and then to find out how these points are distributed. These points will be on the vertices of the linear boundaries and at certain locations in the middle of the boundaries.

For each decision linear boundary, there is a unit vector that is orthogonal to the boundary. Denote that vector by $\mathbf{p}_i$. In other words, the i^{th} boundary is defined as those points $\mathbf{x}$ such that

$$\mathbf{p}_i^T \mathbf{x} \leq \gamma_i \tag{7.11}$$

The point $\mathbf{x} = \gamma_i \mathbf{p}_i$ will be on the i^{th} linear boundary, because it satisfies (7.11). Some of these points may not fall on the edge of $\mathbf{D}_k$, because the set $\mathbf{D}_k$ in RODD-LB2 may be the interior of several linear boundaries. The set $\mathbf{D}_k$ is defined as follows:

$$\mathbf{D}_k = \{\mathbf{x} \in \mathbf{R}^n \,|\, \mathbf{p}_i^T \mathbf{x} \leq \gamma_i : i = 1, 2, \ldots, Q\} \tag{7.12}$$

In order to get a point on the edges of $\mathbf{D}_k$, we take the inner product of each $\mathbf{p}_i$ with all of the $\mathbf{p}_j$, including $\mathbf{p}_i$, and divide by γ_j. Then we find the maximum of $\frac{\mathbf{p}_j^T \mathbf{p}_i}{\gamma_j}$. Say that the maximum occurs for $\mathbf{p}_m$. Then $\frac{\gamma_m \mathbf{p}_i}{\mathbf{p}_m^T \mathbf{p}_i}$ is a point on the m^{th} edge of $\mathbf{D}_k$.

The next step is to find the vertices of $\mathbf{D}_k$ (This can be done using the method described in [con2ver]). Then we form the $\mathbf{Z}$ as follows:

$$\mathbf{Z} = [\mathbf{V}, \mathbf{M}] \tag{7.13}$$

where $\mathbf{V}$ contains all the vertices and $\mathbf{M}$ contains all previously described points on the edges of $\mathbf{D}_k$. Then construct the approximate covariance matrix $\mathbf{A} = \mathbf{Z}\mathbf{Z}^T$. The expression $\mathbf{x}^T\mathbf{A}^{-1}\mathbf{x} = \gamma$ gives a reasonably good orientation of the bounding ellipse. The correct value for γ can be found by the optimization in (7.6), where the constraint is the final $\mathbf{D}_k$ in the RODD-LB2 mode. After γ is found, the set $\mathbf{D}_{k+1}$ will be derived from $\mathbf{D}_k$ using RODD-EB. The transition from RODD-LB2 to RODD-EB saves many calculations, because the optimization in RODD-LB2 is time consuming. The optimization becomes more involved as the number of linear boundaries increases. However, the RODD-EB method is more efficient in this case.

Figure (7.14) illustrates the transition mode from RODD-LB2 to RODD-EB mode for a specific 2D example.

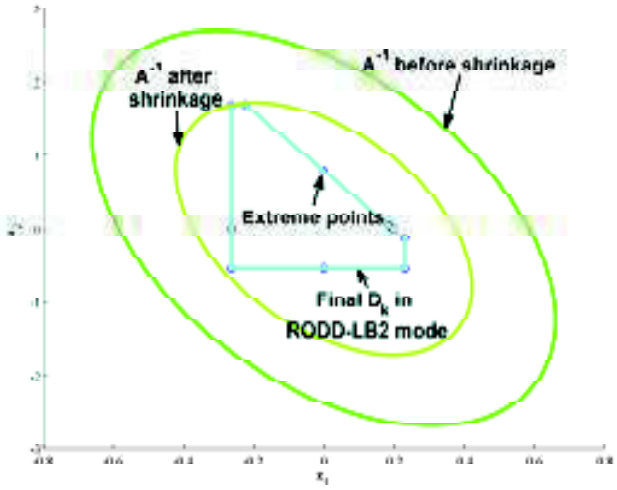

Figure (7.14) RODD-LB2 to RODD-EB Transition Mode

In the following section the RODD methods will be compared

Comparison of RODD Methods

The RODD methods can identify a larger subset of the space of stable parameters than many existing methods. The method is designed to determine stability for systems with convex reachable sets. The efficiency of this method depends on how accurate and how fast the reachable sets can be approximated. In this research, three methods of approximating the reachable sets were introduced. RODD-LB1[BaPr03] and RODD-LB2 use linear approximations of the reachable sets, RODD-EB uses quadratic approximations of the reachable sets and RODD-Hybrid uses combination of linear and quadratic approximation of reachable sets. The speed of convergence will depend on the shape of the reachable sets. For example, if a reachable set has a square shape, then the reachable set is better approximated by set of lines. In this case, the RODD-LB method converges faster than the RODD-EB method. On the other hand, if a reachable set has a shape similar to an ellipse, then it is better approximated by quadratic functions. In this case, RODD-EB converges faster than

RODD-LB. We have found that the RODD-Hybrid method is the most efficient method of detecting stability, because it uses both linear and quadratic approximations of reachable sets. However, RODD-Hybrid cannot detect global asymptotic stability in all cases. There might be some systems with very complex reachable sets for which the RODD-Hybrid method may fail to detect stability.

Figure (7.15) and Figure (7.16) compare the first few iterations of the RODD-LB and RODD-EB methods for two different reachable sets (the reachable sets are approximated by a finite number of points). Figure (7.15) shows a case where the shape of the reachable set is square. In this case, the RODD-LB method can develop a good approximation of the reachable set in few iterations. However, the RODD-EB method is not able to develop the approximate reachable set with the same precision.

Figure (7.16) shows a case where the reachable set is elliptical. In this case, the RODD-EB method can develop a good approximation of the reachable set in few iterations, whereas the RODD-LB method is not able to develop the approximate reachable set with the same precision (within the same number of iterations).

In practice, the shape of the reachable sets are unknown and could be complex, hence we may need to try both methods (RODD-LB and RODD-EB) to find out which method converges faster. For this reason, the RODD-Hybrid method has been introduced. In our experiments on RNN systems, we have found that RODD-Hybrid converges faster than all the previous methods.

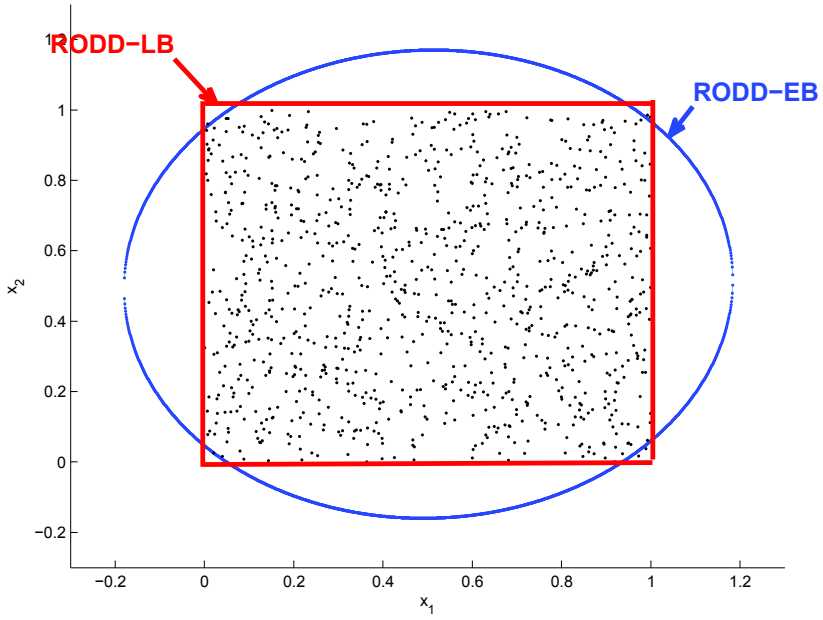

Figure (7.15) Square Reachable set

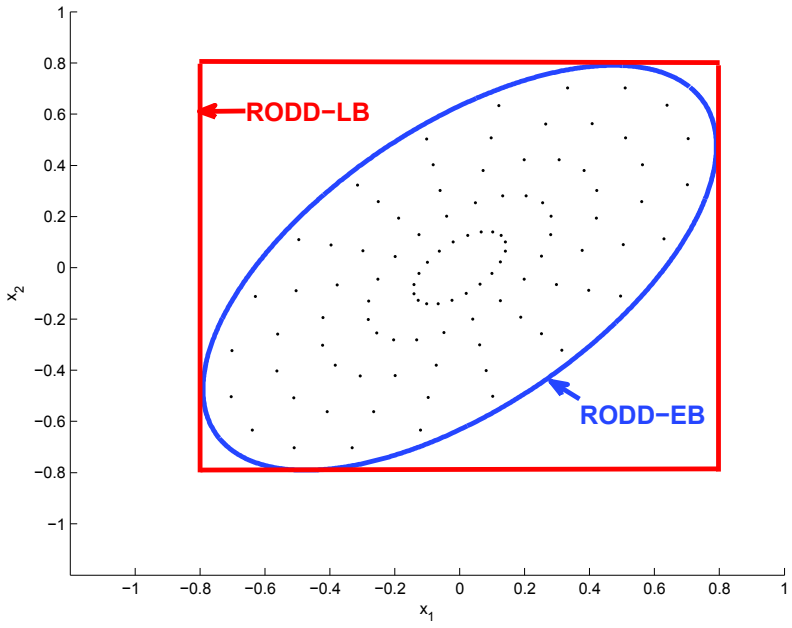

Figure (7.16) Elliptical Reachable set

All the RODD methods conclude global asymptotic stability when the set $\mathbf{D}_k$ becomes small enough. To be able to compare the speed of convergence, the same measurement has been used for all RODD methods. The size of the set $\mathbf{D}_k$ is measured by the following optimization:

$$\beta_k = max\{\|\mathbf{x} - \mathbf{x}^*\|^2 : \mathbf{x} \in \mathbf{D}_k\} \tag{7.14}$$

where $\mathbf{D}_k$ defined by linear or quadratic boundaries. The parameter β_k will be checked at every iteration. For $\beta_k < 10^{-3}$ the algorithm will stop conclude global asymptotic stability.

In the following section, the RODD-LB1, RODD-EB2, RODD-EB and RODD-Hybrid methods will be compared for a simple 2-D example.

Example 7.2 Consider Example 6.1:

$$\mathbf{x}(k+1) = \mathbf{W}\tanh(\mathbf{x}(k)) \text{ where } \mathbf{W} = \begin{bmatrix} 1.8 & 0.95 \\ -0.95 & 0 \end{bmatrix} \tag{7.15}$$

Unlike the CMT method, all of the RODD methods (RODD-LB1, RODD-LB2, RODD-EB and RODD-Hybrid) are able to determine the stability of the origin for the above example, but as shown in Table (7.2) (the simulation results of system (7.15)) RODD-Hybrid has the fastest rate of convergence.

Method	LB1	LB2	EB	Hybrid
Time(sec)	203	103	89	54

Table (7.2) Comparison of RODD methods

Conclusion

In this chapter, three methods of stability analysis for RNNs were developed: RODD-LB2, RODD-EB and RODD-Hybrid. The RODD-LB2 method, which is an extension to the existing RODD-LB1 method [BaPr03] (explained in Chapter 6), uses linear approximations of the reachable sets, and RODD-EB uses quadratic approximations of the reachable sets. RODD-LB2 is more efficient than RODD-LB1, because it uses a more accurate optimization. The accuracy in the optimization allows the RODD-LB2 method to detect stability in fewer iterations.

The RODD-EB method uses an elliptical approximation of the reachable sets. We found in our experiments that for dynamical systems with elliptical reachable sets, the RODD-EB method converges faster than RODD-LB1 and RODD-LB2.

The need for an accurate and fast converging method for detecting stability, leads us to propose a new algorithm. The RODD-Hybrid method, which is a combination of RODD-LB2 and RODD-EB, has been proposed as the most efficient algorithm.

In Chapter 8, we will use the RODD methods to detect global asymptotic stability for different test problems. The efficiency of the each algorithm will be investigated in that chapter.

8 TEST PROBLEMS

The previous chapters investigated different methods for detecting stability of RNNs. We proposed three new algorithms: RODD-LB2, RODD-EB and RODD-Hybrid. The next step is to investigate the performance of the proposed algorithms for different dynamical systems. This is the subject of the following two chapters.

The proposed algorithms either detect stability or are inconclusive. When a result is inconclusive, this does not directly imply instability of the equilibrium point. An equilibrium point could be GAS even though the proposed methods fail to detect stability. This is because all the proposed methods use the approximations of the reachable sets, and there always exist errors in any approximation.

In this chapter, several test problems have been collected. They are separated into a stable group and an unstable group. The proposed methods will be applied to both groups of systems.

Stable Dynamical Systems

In order to investigate the performance of the proposed algorithms, several test problems have been collected. Some of these test problems are in the Lure model from, and some are not. In the following section, the efficiency of the proposed algorithms will be tested on stable systems in the Lure model form.

Lure Model

The Lure model (3.5) is the state space representation which is most suitable for stability analysis with RODD methods. In addition, all systems in the Lure model from can be analyzed stability with the LMI analysis shown in (5.21). In the following examples, stability analysis will be performed with LMI, RODD-LB2, RODD-EB and RODD-Hybrid methods.

Example 8.1 Consider the following system

$$\mathbf{x}(k+1) = \mathbf{B}\tanh(\mathbf{W}\mathbf{x}(k)+\mathbf{b}) \tag{8.1}$$

where $\mathbf{W} = \begin{bmatrix} 0.7133 & 0.5571 \\ 0.7637 & 0.5651 \end{bmatrix}$ $\mathbf{B} = \begin{bmatrix} -0.0176 & 1.4660 \\ -1.9825 & -0.325 \end{bmatrix}$ $\mathbf{b} = \begin{bmatrix} -0.1768 \\ 1.5514 \end{bmatrix}$

This is an example of a system with GAS equilibrium point $\mathbf{z} = \begin{bmatrix} 1.4176 & -0.9396 \end{bmatrix}^T$.This can be verified in Figure (8.1), which shows the response of the system for an arbitrary initial condition. LMI and RODD-EB failed to detect stability of the equilibrium point, whereas RODD-LB2 and RODD-Hybrid detected stability of the equilibrium point. The LMI

criterion was not satisfied, because it is a conservative method of checking stability. The RODD-EB method also failed, due to the shape of the reachable set. To understand the shape of the reachable set in this system, consider Figure (8.2) and Figure (8.3). Figure (8.2) shows a set of points randomly scattered in the initial reachable set $\mathbf{D}_0$. Figure (8.3) shows the same points that have been updated on time step. They represent the shape of $\mathbf{f}(\mathbf{D}_0)$, which could be $\mathbf{D}_1$ in the ideal case. It can be seen that the reachable set does not have an elliptical shape, hence it is very difficult for RODD-EB to detect stability. In this case RODD-LB2 is more efficient. Moreover, the RODD-Hybrid was able to detect stability and improved the speed of convergence 1.5 times compared to RODD-LB2.

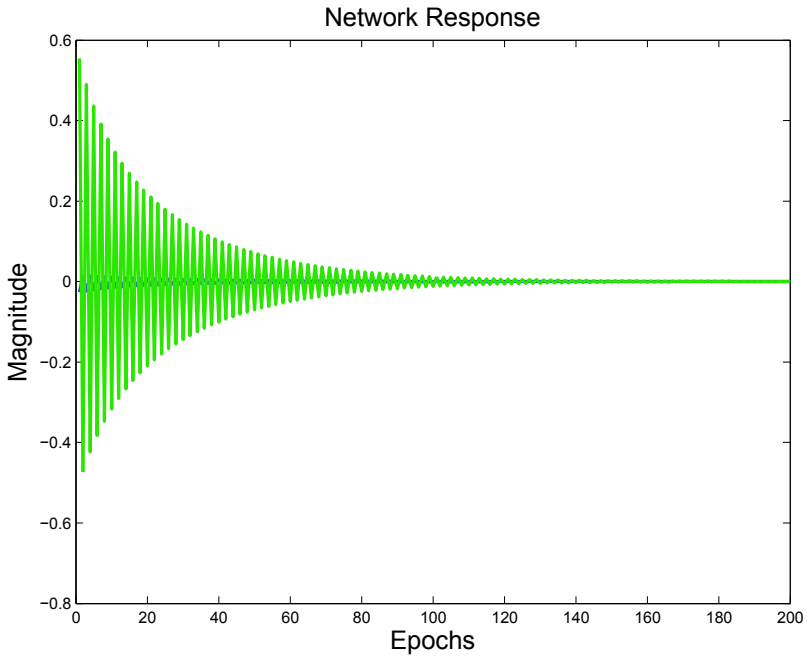

Figure (8.1) System Response

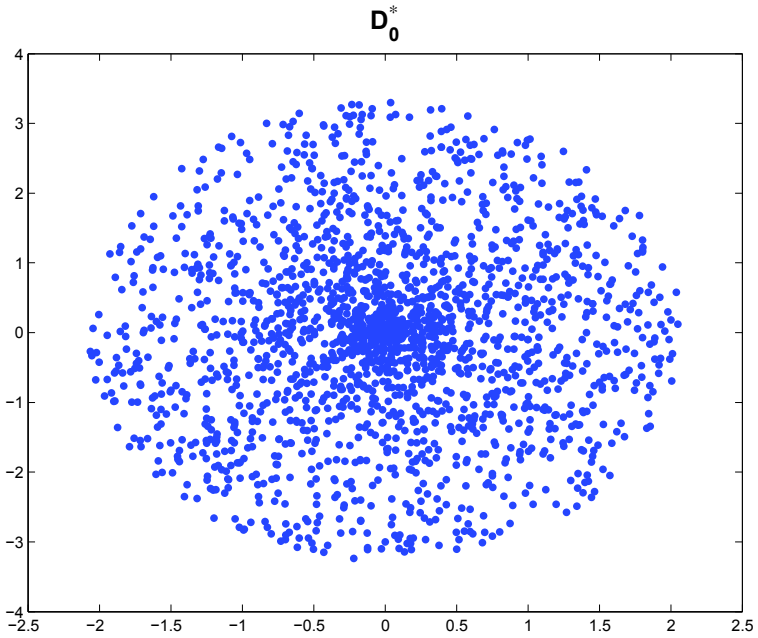

Figure (8.2) Original Data

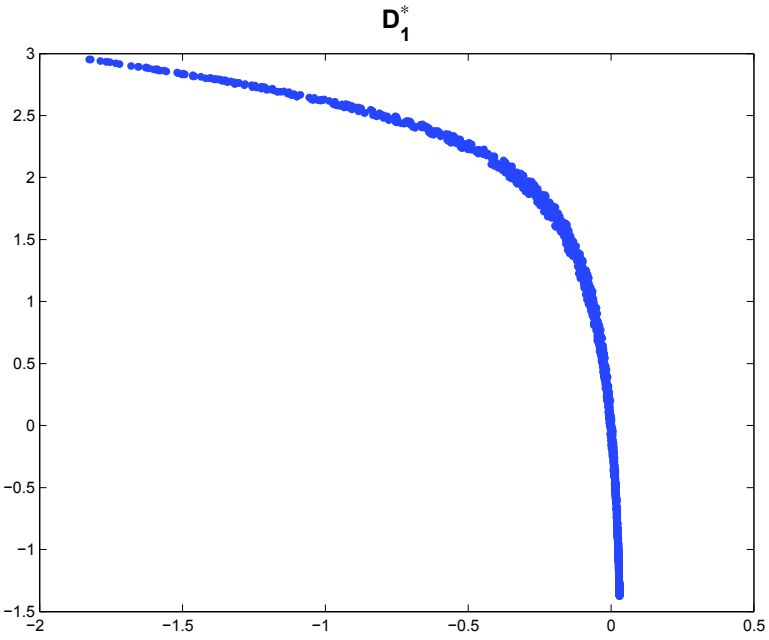

Figure (8.3) Original Data Updated one Time Step

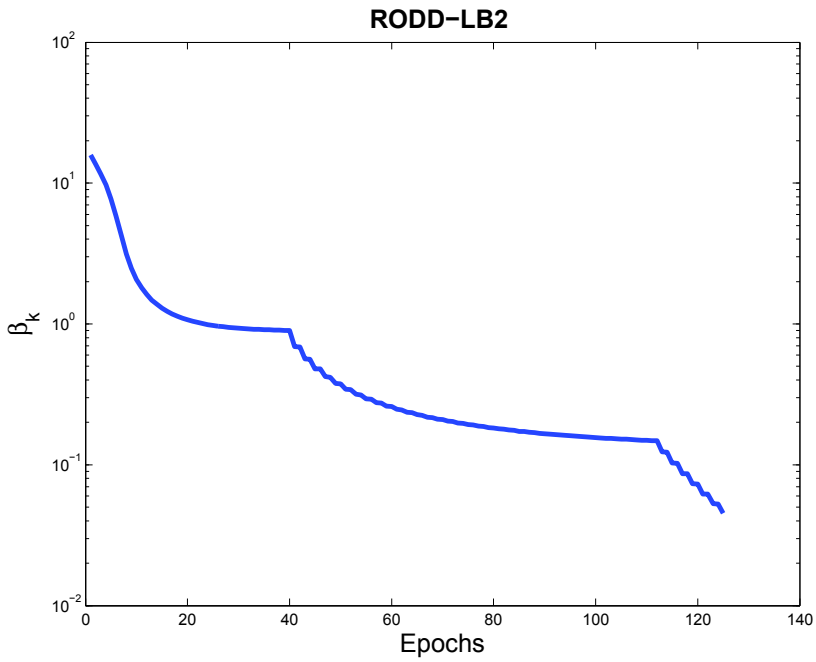

Figure (8.4) Graph of β_k for RODD-LB2

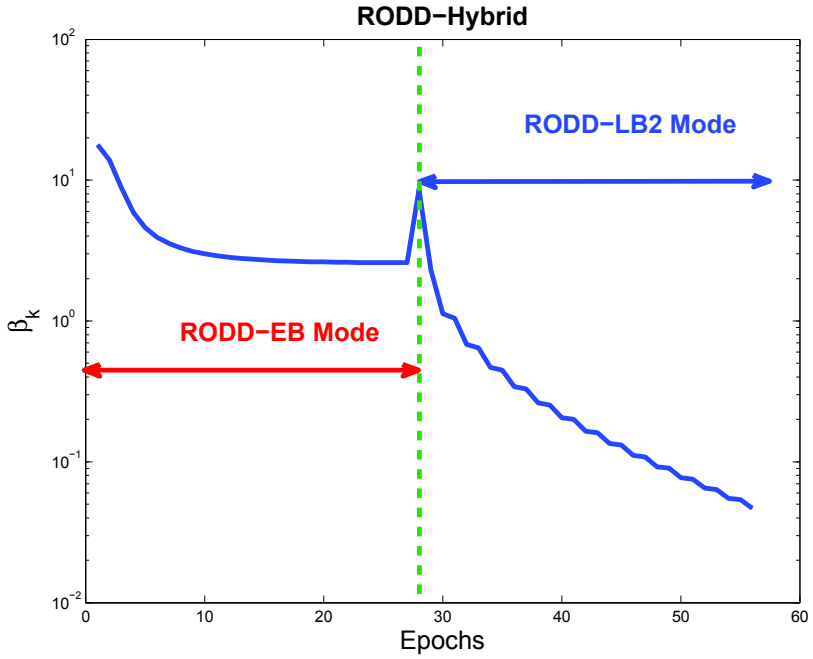

Figure (8.5) Graph of β_k for RODD-Hybrid

Figure (8.4) and Figure (8.5) shows the graph of β_k in the RODD-LB2 and RODD-Hybrid method. In the RODD-Hybrid, the transition from RODD-EB to RODD-LB2 occurred at $k = 29$ where there is a jump. This is due to the change of $\mathbf{D}_k$ in (7.14) from an ellipsoid to a polytope. The reason for this jump explained graphically in (7.13). Table (8.1) illustrates the stability analysis result for this example. RODD-Hyrbid has the fastest speed of convergence. RODD-Hybrid detected the GAS of the equilibrium point 1.5 times faster than RODD-LB2.

	LMI	LB2	EB	Hybrid
Result	Fail	Pass	Fail	Pass
Runtime(sec)	NA	80	837	57
Improvement	NA	$1.5x$	NA	Base

Table (8.1) Comparison of LMI and RODD Methods

In the following example, another Lure form system (higher dimensional) with GAS equilibrium point will be tested.

Example 8.2 Consider the following system

$$\mathbf{x}(k+1) = \mathbf{B}\tanh(\mathbf{W}\mathbf{x}(k) + \mathbf{b}) \tag{8.2}$$

where

$$\mathbf{B} = \begin{bmatrix} -0.144 & -0.192 & -0.224 & 0.534 & 0.231 & 0.810 & 0.214 & -0.304 & 0.726 & 0.249 \\ 1.284 & 0.706 & 0.686 & -0.857 & 1.233 & -1.118 & -0.649 & -0.886 & 0.584 & 1.332 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} -0.082 & -0.654 & -0.151 & -0.617 & -1.132 & 0.515 & 0.568 & -2.823 & -0.066 & 0.265 \\ -0.558 & -0.362 & -1.439 & -0.494 & -0.595 & 0.377 & -0.400 & -1.158 & 0.162 & 0.521 \end{bmatrix}^T$$

$$\mathbf{b} = \begin{bmatrix} 0.859 & -0.177 & -0.658 & 0.204 & 1.445 & -0.458 & 0.213 & 1.527 & 0.161 & -0.233 \end{bmatrix}^T$$

This is an example of two-layered RNN with 10 neurons in the hidden layer. This system has the GAS equilibrium point $\mathbf{z} = \begin{bmatrix} 0.8248 & 0.7799 \end{bmatrix}^T$. The LMI failed to detect the GAS of the equilibrium point, whereas all the RODD methods were able to detect stability. RODD-LB2 has the fastest speed of convergence for this example, which is due to the shape of reachable sets. The set $\mathbf{D}_0$ and $\mathbf{f}(\mathbf{D}_0)$ are represented in Figure (8.6) and Figure (8.7). In this case, linear boundaries can approximate the reachable set better than elliptical boundaries. Hence RODD-LB2 has the fastest speed of convergence. Although RODD-EB could detect the GAS of the equilibrium point, due to the shape of the reachable sets, this method is not as fast as RODD-LB2. RODD-Hybrid does not have the fastest speed of convergence for this example, because it starts with the RODD-EB mode, and that affects the overall speed of RODD-Hybrid method.

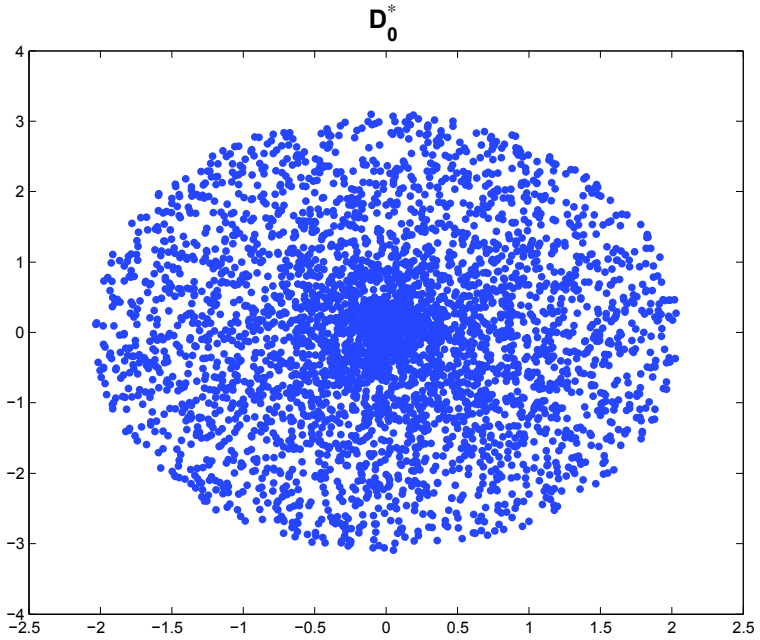

Figure (8.6) Original Data

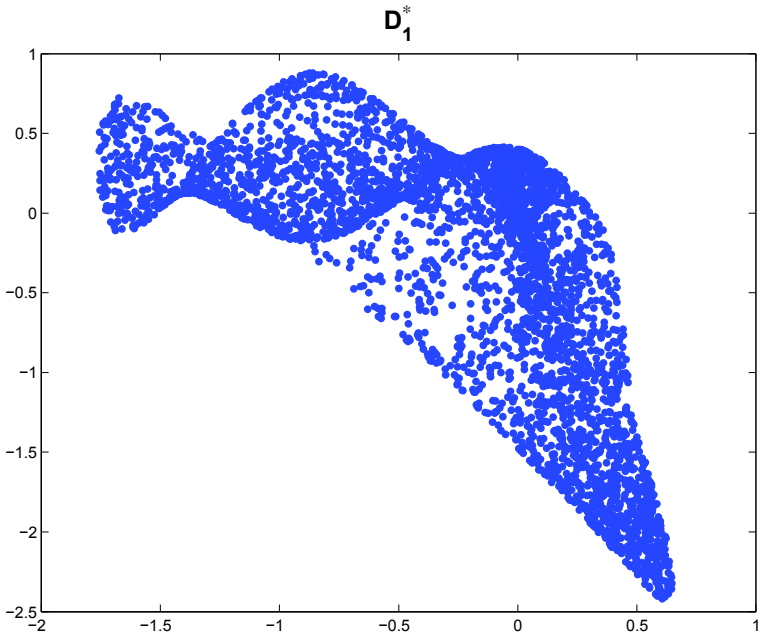

Figure (8.7) Original Data Updated One Time Step

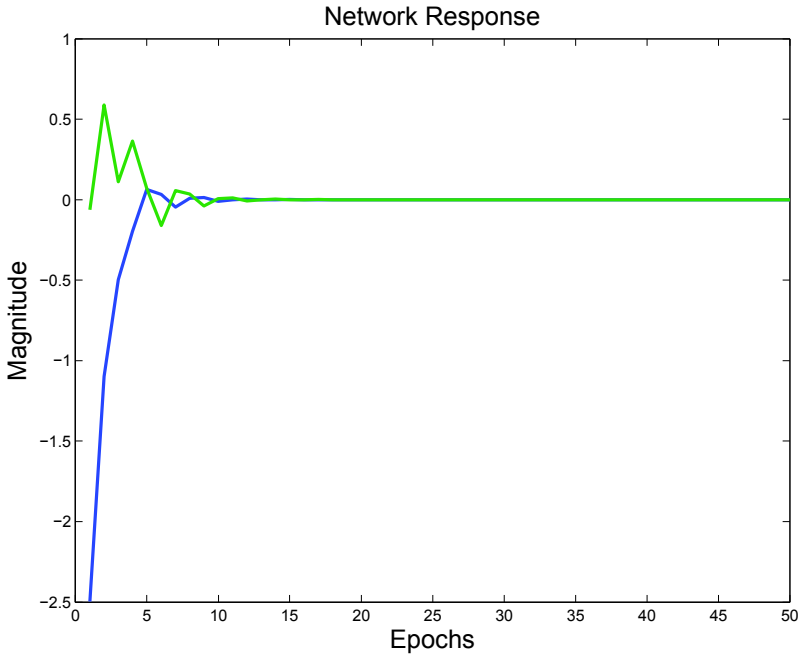

Figure (8.8) System Response

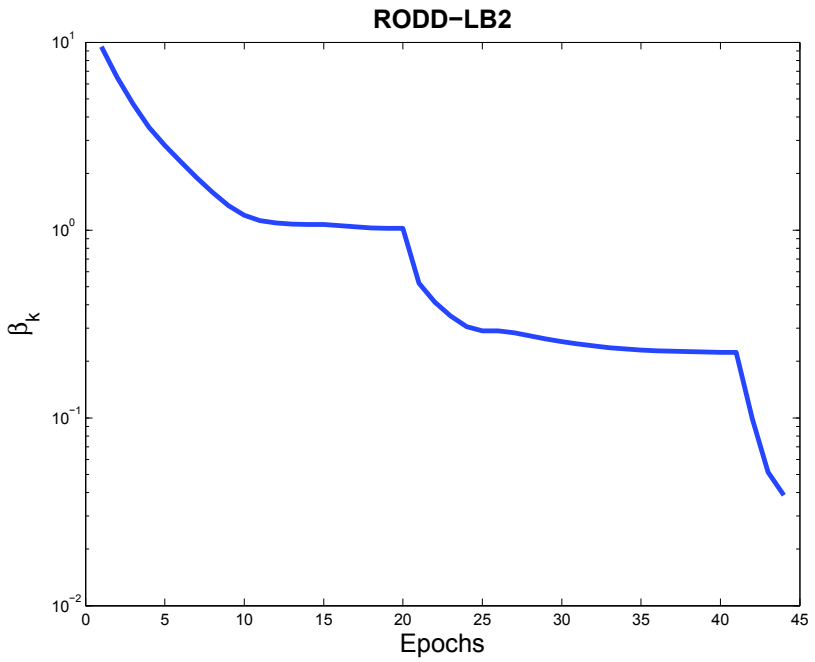

Figure (8.9) Graph of β_k for RODD-LB2

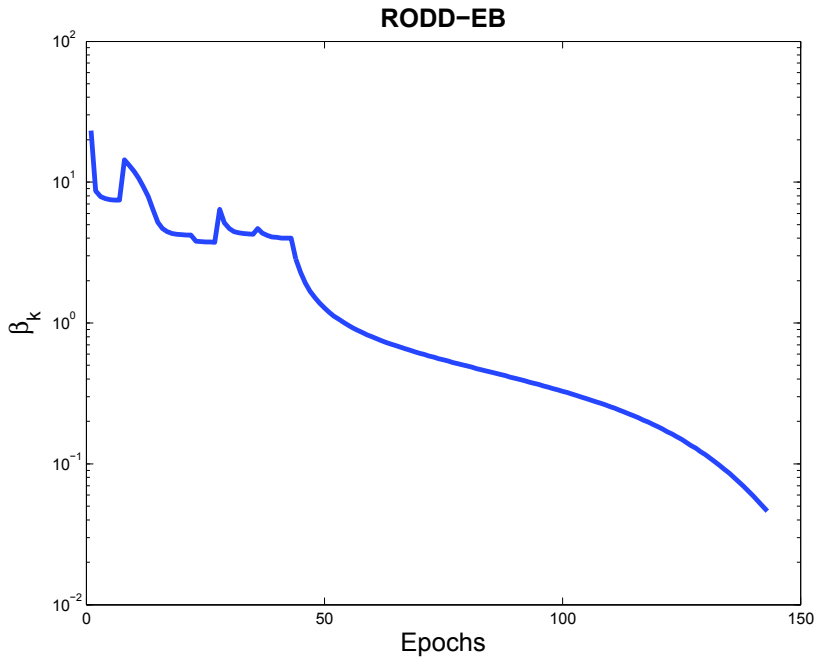

Figure (8.10) Graph of β_k for RODD-EB

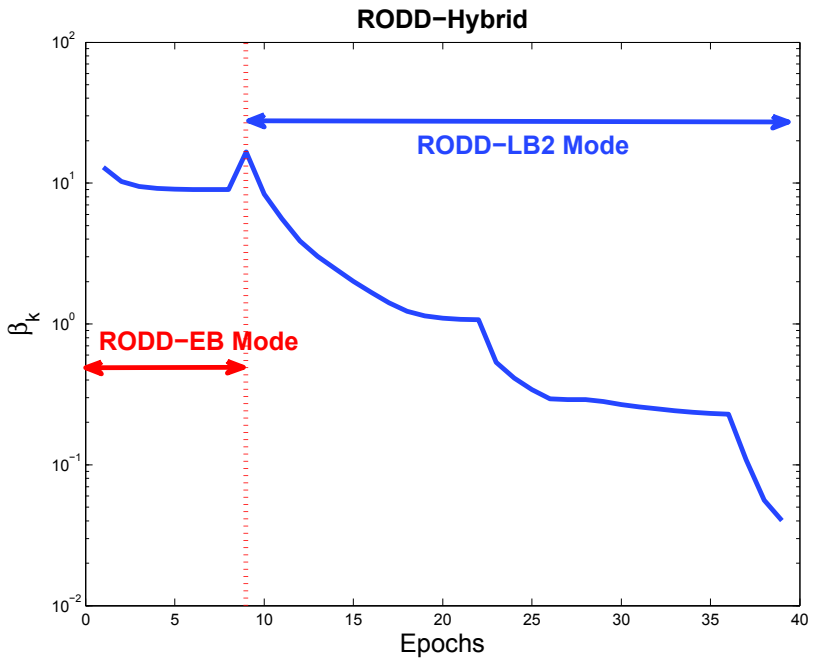

Figure (8.11) Graph of β_k for RODD-Hybrid

The graph of β_k for RODD-LB2, RODD-EB and RODD-Hybrid illustrated in Figure (8.9), Figure (8.10) and Figure (8.11). Note that for RODD-EB the graph of β_k is not always decreasing. This is because of the change of orientation of the $\mathbf{D}_k$. Table (8.2) illustrates the stability results for this example. RODD-LB2 has the fastest speed of convergence for this example. RODD-LB2 detects the GAS of the equilibrium point 6.8 times faster than RODD-EB and 1.1 times faster than RODD-Hybrid.

	LMI	LB2	EB	Hybrid
Result	Fail	Pass	Pass	Pass
Runtime(sec)	NA	35	241	40
Improvement	NA	Base	6.8x	1.1x

Table (8.2) Comparison of LMI and RODD Methods

In the following section, some systems with GAS equilibrium points will be provided that are not in the Lure form.

Non-Lure Model

The non-Lure models cannot be analyzed for stability with the LMI method. However, they can be represented in the state space form (5.1), and they can be analyzed with all the RODD methods. One of the main advantages of the RODD methods over the LMI method, is the flexibility of the these methods to study stability for a wider range of systems. The RODD methods require a system to be represented in state space form, and, as explained in chapter 2, the state space representation is one of the most general forms of system representation. We will show in the next chapter that MRAC and NARMA-L2 control problems cannot be modeled to be in Lure form, and their stability analysis will be done through RODD methods.

In the following examples, stability analysis will be investigated using RODD-LB2, RODD-EB and RODD-Hybrid.

Example 8.3 Consider a double pendulum system shown in Figure (8.12). In this example, we considered a double pendulum system with viscous friction. This is an example of a dynamical system with GAS equilibrium point. A response can be seen in Figure (8.13), which shows the state space simulation of this system for an arbitrary initial condition.

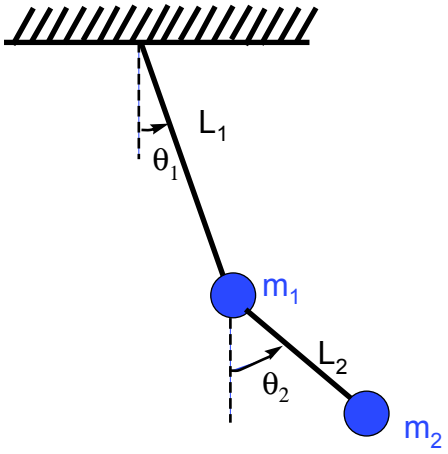

Figure (8.12) Double Pendulum

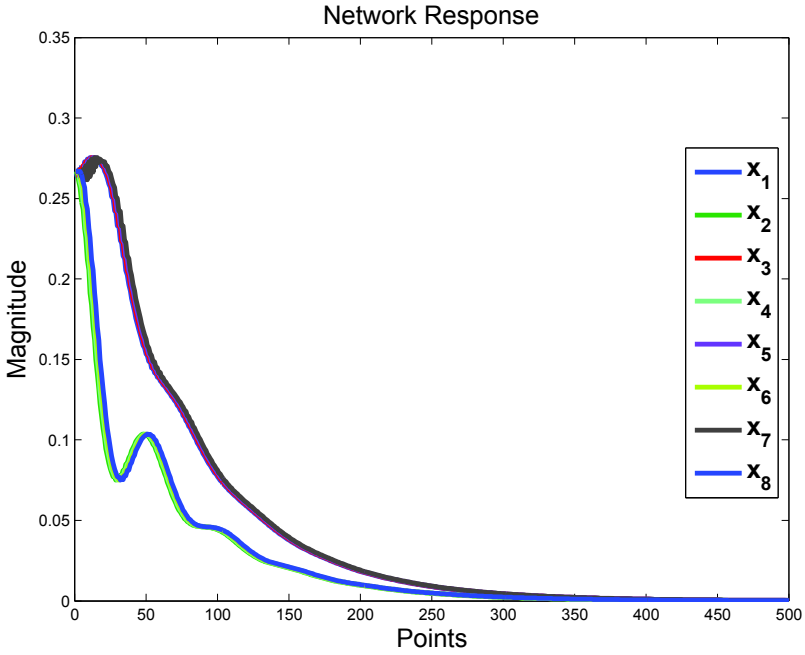

Figure (8.13) Network Response

In order to check the performance of proposed stability methods, we need to model the double pendulum system with an RNN model. Then the proposed stability analysis methods will be applied to the RNN model. The block diagram of this model is illustrated

in Figure (8.14). In this model, we used four output delays, 20 neurons in the hidden layer, with a hyperbolic tangent activation function. For the training data set, we generated 20 sequences with random initial conditions in Matlab/Simulink. The RNN model has been trained with the two measured outputs, angular position θ_1 and θ_2, and 10000 data points. The training was done with the neural network toolbox in Matlab ([BeHa12]).

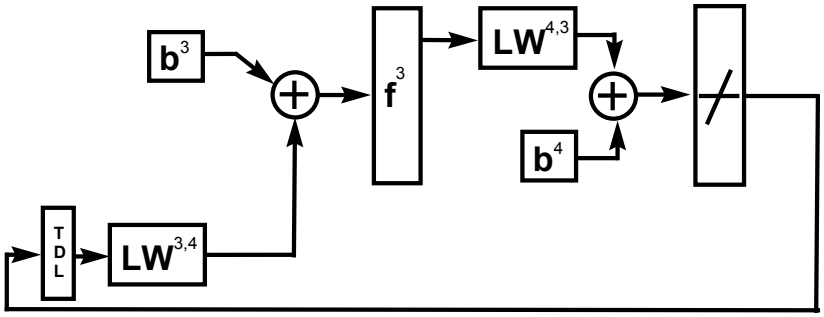

Figure (8.14) RNN Model of Double Pendulum

The network response after training, and the error between the target and the network response, are shown in Figure (8.15).

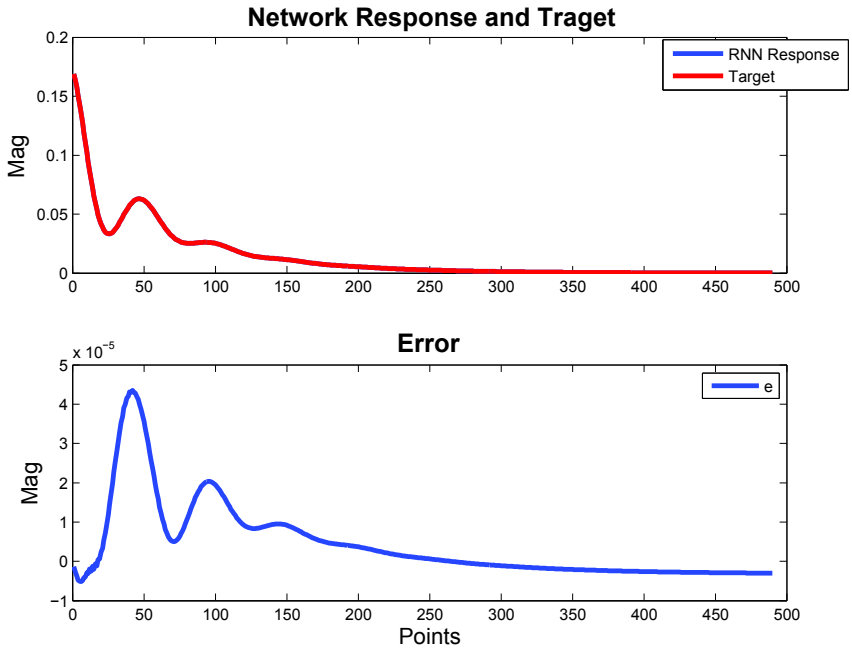

Figure (8.15) RNN Model of Double Pendulum After Training

The trained RNN, which is now equivalent to the double pendulum, can be written in state space form as follows:

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}\mathbf{x}(k) + \mathbf{b}_3) + b_4 \\ x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} \tag{8.3}$$

where $\mathbf{x}(k) = \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) \end{bmatrix}^T$ (x_1 and x_2 are the angular positions of the first and the second pendulum, x_3 and x_4 are the angular velocities of the first and the second pendulum) and

$$\mathbf{LW}^{3,4} = \begin{bmatrix} -0.12 & -0.10 & -0.01 & 0.09 \\ -0.02 & -0.08 & -0.05 & -0.11 \\ -0.14 & -0.14 & -0.18 & 0.03 \\ -0.14 & -0.10 & 0.12 & 0.07 \\ 0.08 & -0.01 & 0.09 & 0.00 \\ -0.26 & 0.08 & -0.07 & 0.01 \\ -0.04 & -0.03 & 0.05 & 0.05 \\ 0.03 & -0.03 & 0.11 & 0.02 \\ 0.06 & 0.14 & -0.00 & -0.07 \\ 0.20 & -0.07 & -0.05 & -0.07 \\ -0.09 & 0.10 & -0.21 & 0.03 \\ -0.12 & -0.09 & 0.15 & 0.06 \\ 0.18 & 0.19 & -0.15 & -0.01 \\ -0.06 & 0.00 & -0.10 & -0.00 \\ 0.14 & 0.07 & -0.09 & -0.03 \\ 0.16 & 0.15 & -0.13 & -0.06 \\ 0.10 & 0.03 & -0.07 & -0.09 \\ -0.08 & -0.03 & 0.07 & 0.00 \\ -0.06 & -0.09 & -0.04 & -0.00 \\ 0.12 & 0.22 & -0.12 & -0.04 \end{bmatrix}, \mathbf{LW}^{3,4} = \begin{bmatrix} -0.66 \\ -0.61 \\ -0.22 \\ -0.03 \\ -1.04 \\ -1.02 \\ 0.76 \\ -0.92 \\ -0.74 \\ 0.99 \\ 1.24 \\ -0.45 \\ 0.42 \\ 1.01 \\ 0.35 \\ 0.00 \\ -0.29 \\ -0.97 \\ -0.06 \\ -0.01 \end{bmatrix}^T, \mathbf{b}_3 = \begin{bmatrix} -0.00 \\ -0.07 \\ -0.27 \\ 0.02 \\ -0.00 \\ 0.00 \\ -0.05 \\ 0.02 \\ -0.05 \\ 0.00 \\ -0.01 \\ 0.15 \\ -0.04 \\ 0.14 \\ 0.00 \\ -0.03 \\ 0.08 \\ 0.09 \\ 0.03 \\ -0.13 \end{bmatrix}, b_4 = -0.01$$

The proposed stability methods can now be applied to the state equation given in (8.3). Figure (8.16) and Figure (8.17) show the graphs of β_k for RODD-LB2 and RODD-Hybrid. All the RODD methods were able to detect stability of the equilibrium point for this example. In this example, RODD-EB and RODD-Hybrid have the same graph for the β_k. This is because RODD-Hybrid starts with the RODD-EB mode, and, for this example RODD-Hybrid never goes to the RODD-LB2 mode.

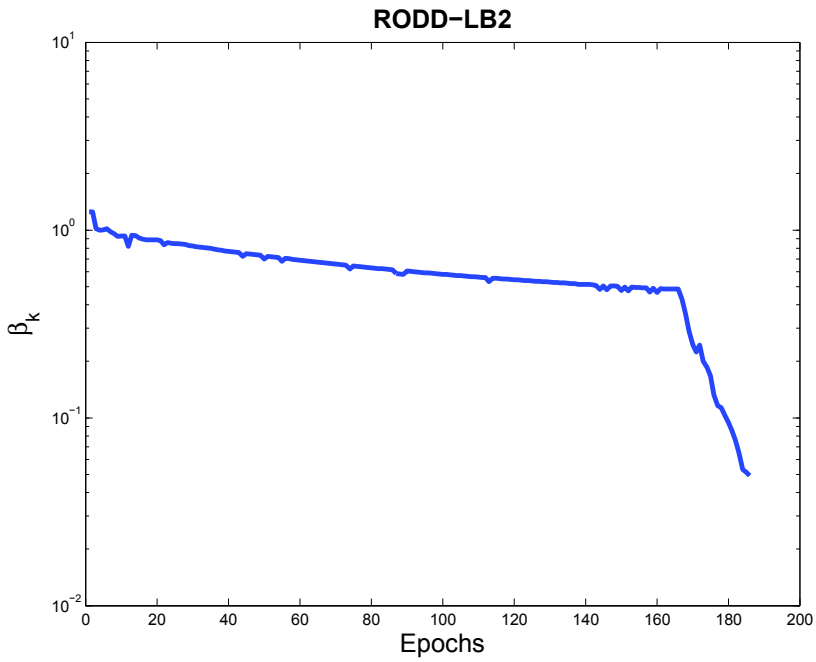

Figure (8.16) Graph of β_k for RODD-LB2

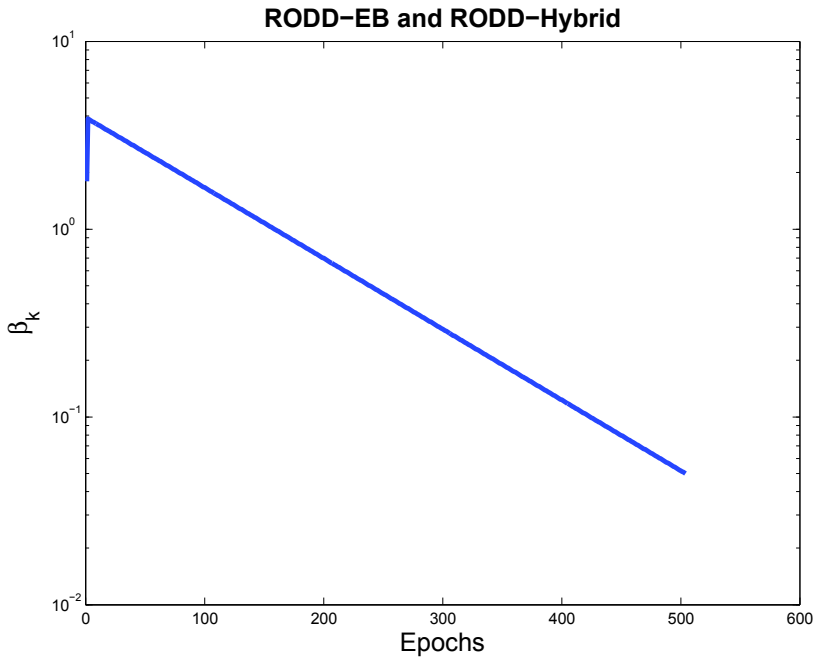

Figure (8.17) Graph of β_k for RODD-EB and RODD-Hybrid

Table (8.6) illustrates the result of the stability analysis for different methods. RODD-EB (RODD-Hybrid) is more efficient in detecting stability of the equilibrium point compared to RODD-LB2. For this example RODD-EB (RODD-Hybrid) is 1.07 times faster than RODD-LB2.

	LB2	EB	Hybrid
Result	Pass	Pass	Pass
Runtime(sec)	302	282	282
Improvement	$1.07x$	Base	Base

Table (8.3) Comparison RODD Methods

Example 8.4 Consider the following system

$$\begin{bmatrix} x(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}z(k) + \mathbf{LW}^{3,2}x(k) + \mathbf{b}_3) + b_4 \\ \mathbf{LW}^{2,1}\tanh(\mathbf{LW}^{1,2}z(k) + \mathbf{LW}^{1,4}x(k) + \mathbf{b}_1) + b_2 \end{bmatrix} \tag{8.4}$$

where

$$\mathbf{LW}^{4,3} = \begin{bmatrix} -2.115 & 0.462 & -2.459 & -0.304 & 0.3982 & -1.705 & 0.759 & 0.203 & 0.958 & 0.486 \end{bmatrix}$$

$$\mathbf{LW}^{2,1} = \begin{bmatrix} 0.297 & -0.71 & -0.687 & 0.572 & 0.445 & -1.326 & -1.499 & 1.016 & -0.027 & -0.16 \end{bmatrix}$$

$$\mathbf{LW}^{3,4} = \begin{bmatrix} 0.1319 & -0.999 & -0.354 & 0.1056 & 0.955 & 0.464 & 0.266 & -1.423 & 0.111 & 1.172 \end{bmatrix}^T$$

$$\mathbf{LW}^{3,4} = \begin{bmatrix} 0.378 & -0.107 & 0.212 & 0.468 & -1.616 & 0.014 & -0.037 & 0.403 & 0.305 & -1.243 \end{bmatrix}^T$$

$$\mathbf{LW}^{1,2} = \begin{bmatrix} -0.794 & -0.227 & 1.593 & 0.155 & 0.178 & -0.337 & -1.525 & -0.709 & -0.866 & 0.071 \end{bmatrix}^T$$

$$\mathbf{LW}^{1,4} = \begin{bmatrix} 0.155 & -0.182 & 0.731 & -0.347 & 0.234 & -0.867 & -0.982 & -0.293 & 0.824 & 0.194 \end{bmatrix}^T$$

$$\mathbf{b}_3 = \begin{bmatrix} -0.924 & -0.72 & 0.833 & -0.566 & -0.999 & 0.641 & -0.254 & -1.121 & 1.647 & -0.306 \end{bmatrix}^T$$

$$\mathbf{b}_1 = \begin{bmatrix} -0.445 & 0.325 & -0.02 & -1.888 & 0.338 & 0.337 & 0.424 & 0.112 & -0.118 & -2.686 \end{bmatrix}^T$$

$$b_2 = 1.548$$

$$b_4 = -0.774$$

This is an example of a system in the state space form with GAS equilibrium point $\mathbf{z} = \begin{bmatrix} -0.959 & -0.565 \end{bmatrix}^T$ (we will show in the next chapter that this is an example of MRAC Control). Since (8.4) is not directly in the Lure form, the LMI method cannot be used to investigate the stability of the equilibrium point, whereas this form is suitable for all of the RODD methods.

Due to the shape of the reachable sets, which are less elliptical and not symmetric, RODD-EB could not detect the GAS of the equilibrium point of this system. However, RODD-LB2 and RODD-Hybrid successfully detected the GAS of the equilibrium point with a fast rate of convergence. The set $\mathbf{D}_0$ and $\mathbf{f}(\mathbf{D}_0)$ are represented in Figure (8.18) and Figure (8.19). The fact that $\mathbf{f}(\mathbf{D}_0)$ does not have an elliptical shape indicates that RODD-EB may have some difficulties.

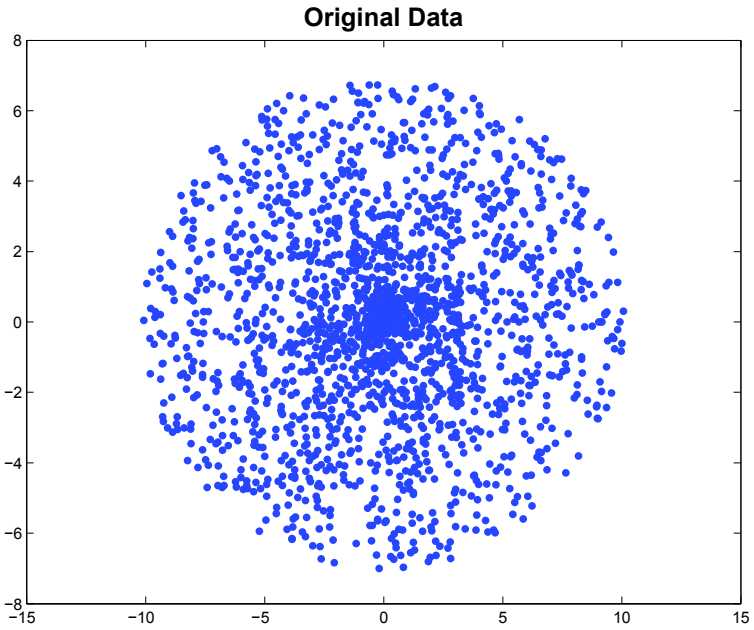

Figure (8.18) Original Data

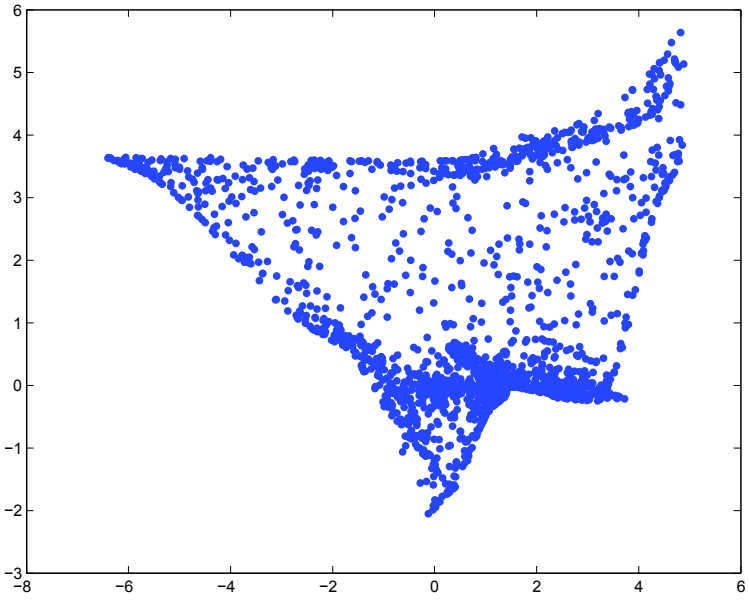

Figure (8.19) Original Data Updated One Time Step

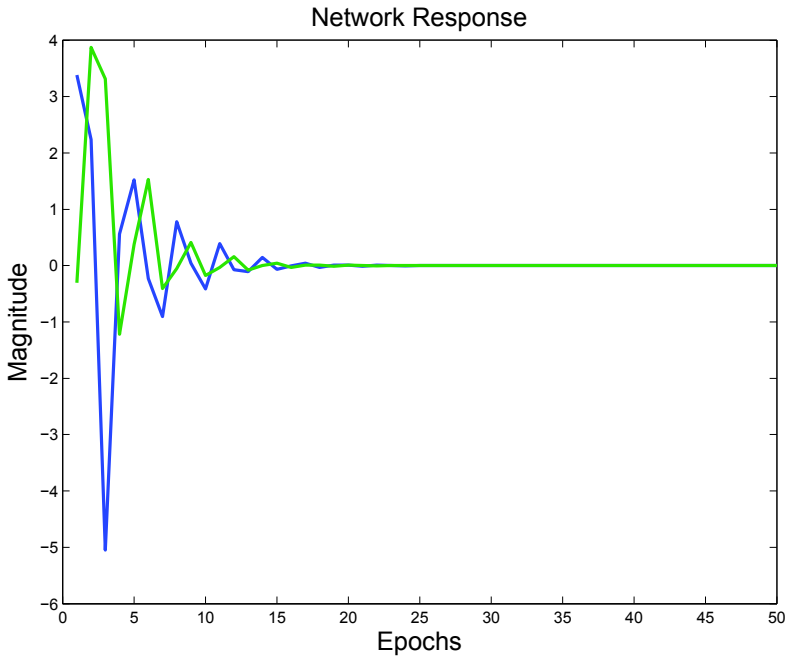

Figure (8.20) System Response

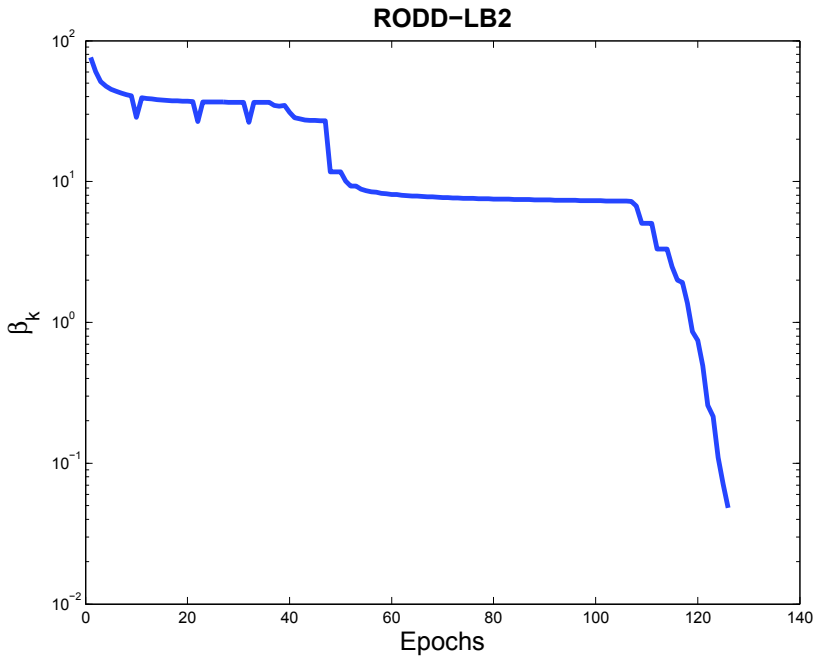

Figure (8.21) Graph of β_k for RODD-LB2

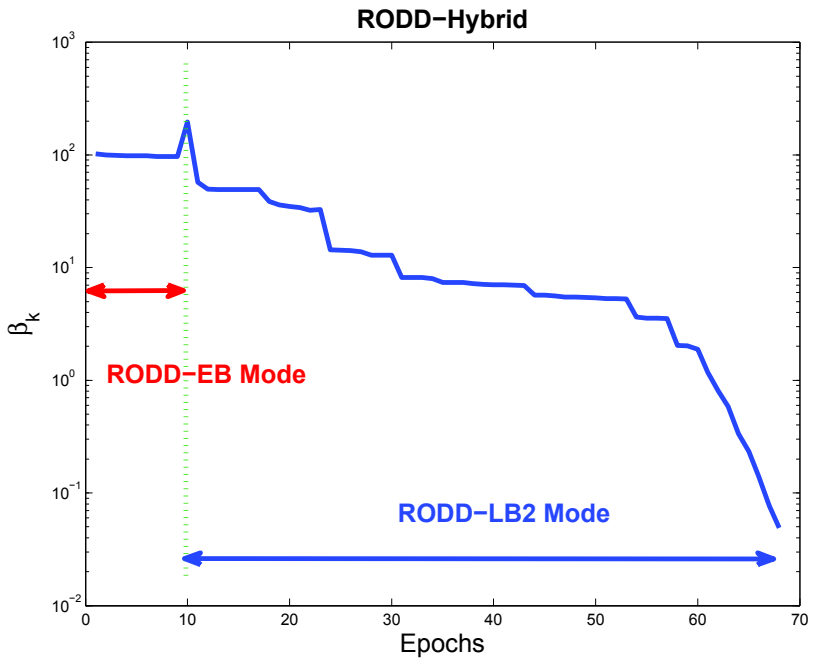

Figure (8.22) Graph of β_k for RODD-Hybrid

Table (8.4) illustrates the stability results for this example. RODD-Hybrid has the fastest speed of convergence for this example. In this example, RODD-Hybrid detects the GAS of the equilibrium point 1.5 times faster than RODD-LB2.

	LB2	EB	Hybrid
Result	Pass	Fail	Pass
Runtime(sec)	130	5495	83
Improvement	1.5x	NA	Base

Table (8.4) Comparison RODD Methods

Unstable Dynamical Systems

In the second category, the proposed method will be applied to some dynamical systems with unstable equilibrium points. Some of these test problems are in the Lure model form, and some are not. In the following section, the systems are in Lure model form.

Lure Model

As explained in Chapter 6, RODD methods can only confirm the GAS of the equilibrium point; they cannot confirm instability. This is due to the approximation errors in the reachable sets. In the following examples, we will apply the proposed stability method to some systems in the Lure form with unstable equilibrium points.

Example 8.5 Consider the following system

$$\mathbf{x}(k+1) = \mathbf{B}\tanh(\mathbf{W}\mathbf{x}(k)+\mathbf{b}) \tag{8.5}$$

where

$$\mathbf{B} = \begin{bmatrix} -0.885 & -0.077 & 0.344 & -1.051 & -0.256 & 0.847 & 1.431 & 0.840 \\ 0.380 & 2.226 & -1.409 & 0.675 & -1.354 & -0.335 & -0.035 & -1.650 \\ -0.158 & -2.163 & -0.452 & 1.487 & 0.335 & 0.387 & -0.061 & 1.648 \\ -0.300 & -0.788 & 0.327 & 1.198 & -0.133 & -0.219 & -0.262 & 0.442 \\ 3.110 & 0.931 & 0.528 & 0.358 & -1.447 & 0.061 & -0.154 & -1.352 \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} -0.693 & -0.716 & 0.724 & --1.444 & 0.681 & 0.251 & -1.625 & -0.073 \\ 0.113 & -1.317 & -0.610 & -0.126 & -0.467 & 0.585 & 1.034 & 1.043 \\ 0.453 & -1.015 & 0.186 & -0.427 & -1.414 & 0.148 & -1.251 & -0.313 \\ -0.148 & -0.166 & -0.319 & 0.510 & -1.672 & -2.141 & -0.418 & 0.731 \\ 0.240 & 0.783 & 0.813 & -0.798 & -1.062 & 1.679 & 0.420 & -0.353 \end{bmatrix}^T$$

$$\mathbf{b} = \begin{bmatrix} -0.508 & 0.483 & 1.464 & -0.786 & -0.929 & 0.884 & -1.405 & 0.245 \end{bmatrix}^T$$

This is an example of a system with an unstable equilibrium point. As expected, all the methods, LMI, RODD-LB2, RODD-EB and RODD-Hybrid, fail to detect the GAS of the equilibrium point. Figure (8.23) shows the network response for an arbitrary initial condition. This is enough to confirm instability of the equilibrium point for this system.

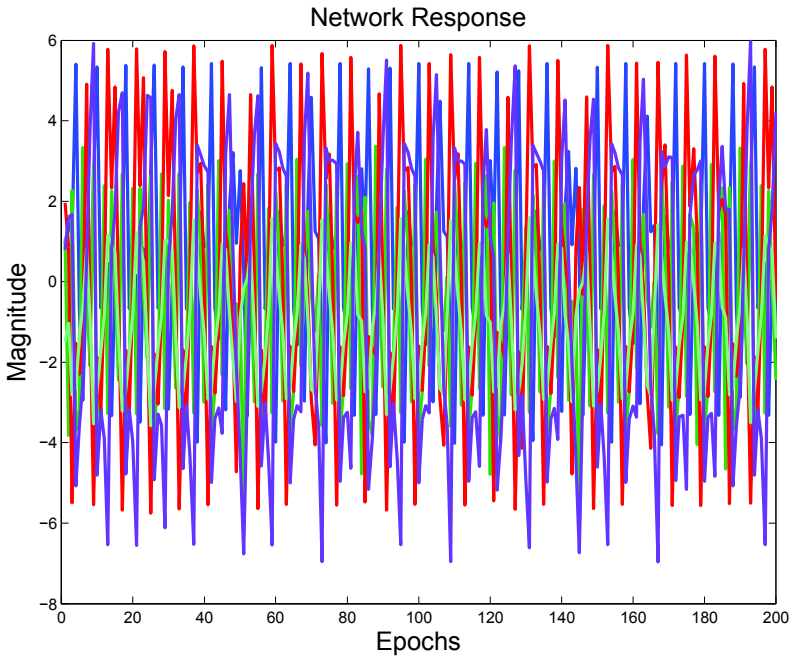

Figure (8.23) Network Response

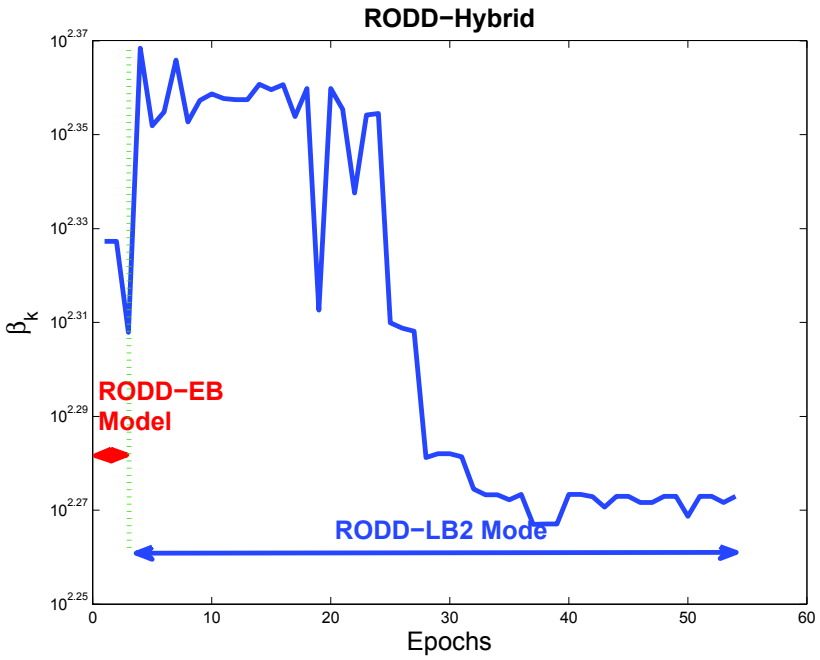

Figure (8.24) Graph of β_k for RODD-Hybrid

The graph of β_k is shown in Figure (8.24) for the RODD-Hybrid. At $k = 3$, the algorithm, switches from RODD-EB to RODD-LB2. Due to the instability of the equilibrium point, β_k failed to continue to decrees, and the algorithm stops because additional linear boundaries in RODD-LB2 can not make $\mathbf{D}_{k+1}$ smaller than $\mathbf{D}_k$.

Table (8.5) illustrates the results of stability analysis for different methods. The RODD-Hybrid method is more efficient in failing to confirm stability of the equilibrium point compared to the other RODD methods. RODD-Hybrid failed 31.6 times faster than RODD-LB2 and 2.5 times faster than RODD-EB.

	LMI	LB2	EB	Hybrid
Result	Fail	Fail	Fail	Fail
Runtime(sec)	NA	14450	1173	457
Improvement	NA	31.6*x*	2.5*x*	Base

Table (8.5) Comparison of LMI and RODD Methods

In the following section, some systems with unstable equilibrium points will studied that are not in the Lure form.

Non-Lure Model

As we explained in the previous section, the non-Lure model cannot be studied with LMI methods, whereas all the RODD methods can be used if the system can be written in state space form (5.1).

In the following example, stability analysis will be investigated with RODD-LB2, RODD-EB and RODD-Hybrid.

Example 8.6 Consider again the double pendulum system shown in Figure (8.12).

This time, we considered a double pendulum system without viscous friction. This system exhibits rich dynamic behavior with strong sensitivity to initial conditions. This is an example of a dynamical system with unstable equilibrium point. Figure (8.25) shows the network response for an initial condition which confirms the instability of the equilibrium point.

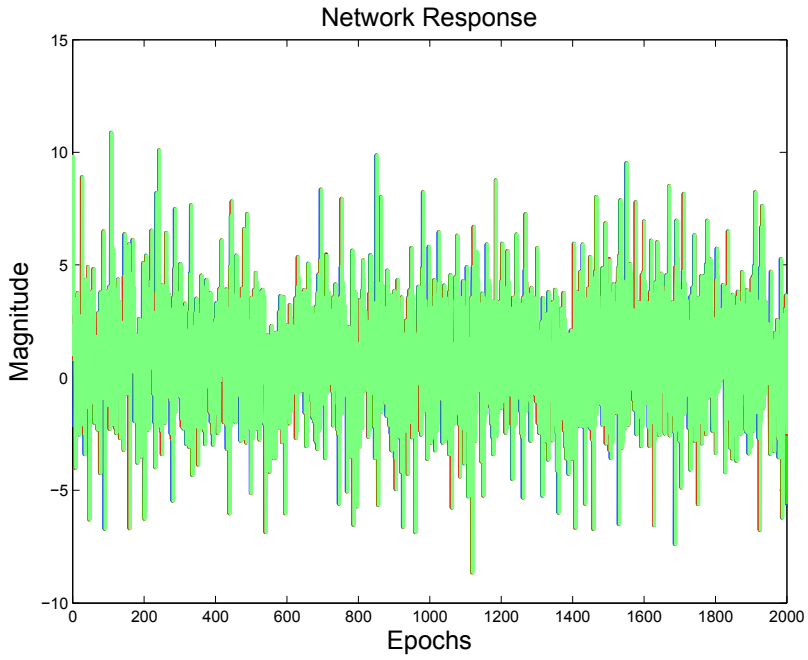

Figure (8.25) Network Response

In order to check the performance of the proposed stability methods, we modelled double pendulum system with an RNN model. The proposed methods will be applied to the RNN model. We used the same RNN model shown in Figure (8.14). In this model, we used four output delays, 20 neurons in the hidden layer, with hyperbolic tangent activation function. For the training data set, we generated 20 sequences with random initial conditions in Matlab/Simulink. The RNN model was trained with 40000 points. The training was done

with neural network toolbox in Matlab ([BeHa12]). Figure (8.26) shows the response of the RNN model after training and the error between the target and the network response. Although the RNN response does not perfectly match the true response, it has a similar characteristics. This is sufficient for testing our algorithms.

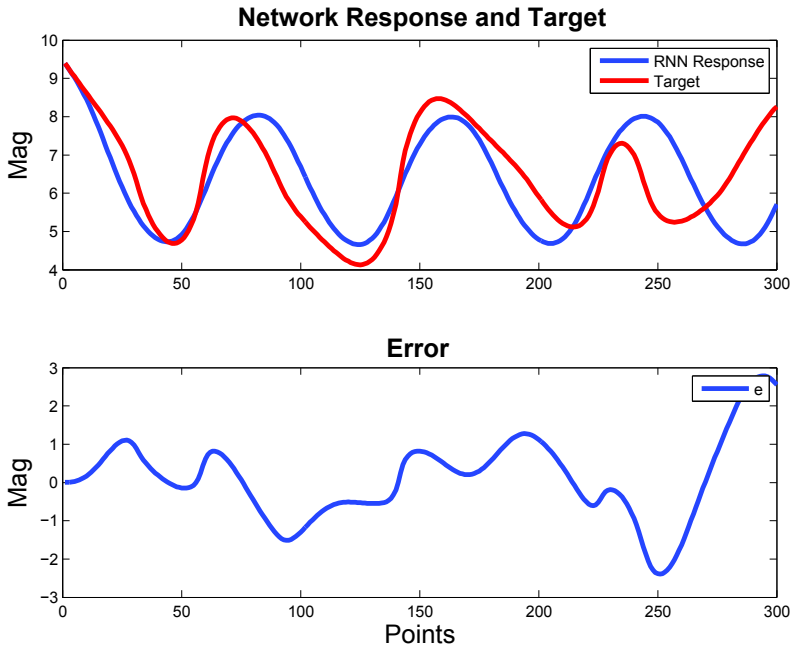

Figure (8.26) RNN Model of Double Pendulum After Training

The trained RNN double pendulum model can be written in state space form as follows:

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}\mathbf{x}(k) + \mathbf{b}_3) + b_4 \\ x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} \tag{8.6}$$

where $\mathbf{x}(k) = \begin{bmatrix} x_1(k) & x_2(k) & x_3(k) & x_4(k) \end{bmatrix}^T$ (x_1 and x_2 are the angular positions of the first

and the second pendulum, x_3 and x_4 are the angular velocities of the first and the second pendulum) and

$$\mathbf{LW}^{3,4} = \begin{bmatrix} 1.47 & 2.49 & 0.83 & -0.31 \\ -1.17 & -2.29 & -0.38 & -0.83 \\ -0.27 & 2.42 & -0.01 & -1.19 \\ -0.91 & 1.97 & -0.22 & -1.90 \\ 1.78 & 1.18 & -2.94 & -1.35 \\ 0.05 & -1.41 & -0.79 & 0.56 \\ 0.89 & -0.25 & -2.20 & 1.77 \\ 2.98 & -1.45 & -0.24 & -1.30 \\ 4.70 & 3.82 & -3.29 & -5.24 \\ -1.45 & 0.12 & 0.26 & -2.25 \\ -0.95 & 0.67 & -1.75 & -1.04 \\ -2.38 & -0.00 & 2.41 & 1.99 \\ 5.53 & -0.92 & -0.42 & -4.17 \\ -4.34 & -1.04 & -0.12 & -3.73 \\ 1.76 & -2.88 & -0.34 & -0.72 \\ -0.91 & 2.64 & -0.10 & 0.26 \\ 2.27 & 2.30 & 2.14 & 2.47 \\ -4.29 & 0.01 & -0.39 & 3.06 \\ -0.55 & -2.93 & -4.52 & -1.12 \\ -2.00 & 1.73 & 1.74 & 0.51 \end{bmatrix}, \mathbf{LW}^{3,4} = \begin{bmatrix} 0.00 \\ 0.00 \\ -0.19 \\ 0.43 \\ -0.25 \\ -0.75 \\ 1.35 \\ -3.34 \\ -1.55 \\ 0.21 \\ -0.31 \\ 0.53 \\ 3.59 \\ -0.86 \\ -0.17 \\ 0.17 \\ -1.84 \\ -0.12 \\ -0.97 \\ -0.66 \end{bmatrix}^T, \mathbf{b}_3 = \begin{bmatrix} -2.79 \\ 2.30 \\ -2.63 \\ 1.51 \\ -1.73 \\ 2.06 \\ -0.27 \\ -0.02 \\ -0.01 \\ -0.10 \\ -0.10 \\ -2.35 \\ -0.01 \\ -3.63 \\ 1.01 \\ 1.00 \\ 3.60 \\ -2.01 \\ -3.58 \\ -2.46 \end{bmatrix}, b_4 = -0.03$$

The graph of β_k for RODD-Hybrid is shown in Figure (8.27). At $k = 123$ the algorithm switches from RODD-EB to RODD-LB2. Due to the instability of the equilibrium point, β_k would not get small enough to confirm stability; additional linear boundaries in RODD-LB2 cannot make $\mathbf{D}_{k+1}$ smaller than $\mathbf{D}_k$.

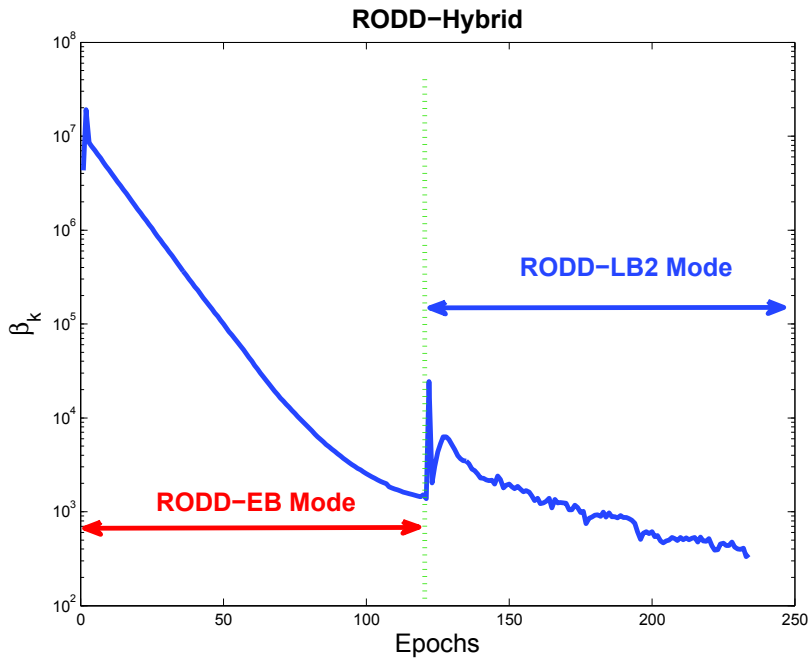

Figure (8.27) Graph of β_k for RODD-Hybrid

Table (8.6) illustrates the result of the stability analysis for different methods. The RODD-LB2 method is more efficient in failing to verify stability of the equilibrium point compared to the other RODD methods. For this example, RODD-EB takes longer to stop, which affects the speed of RODD-Hybrid. This is because RODD-Hybrid always starts with RODD-EB.

	LB2	EB	Hybrid
Result	Fail	Fail	Fail
Runtime(sec)	8165	14400	13218
Improvement	Base	1.7x	1.6x

Table (8.6) Comparison RODD Methods

In the following example, another system in non-Lure form with unstable equilibrium point will be studied.

Example 8.7 Consider the following system:

$$\begin{bmatrix} x(k+1) \\ \mathbf{z}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}\mathbf{z}(k) + \mathbf{LW}^{3,2}x(k) + \mathbf{b}_3) + b_4 \\ \mathbf{LW}^{2,1}\tanh(\mathbf{LW}^{1,2}\mathbf{z}(k) + \mathbf{LW}^{1,4}x(k) + \mathbf{b}_1) + b_2 \\ z_1(k) \end{bmatrix} \tag{8.7}$$

where $\mathbf{z}(k) = \begin{bmatrix} z_1(k) & z_2(k) \end{bmatrix}$

$$\mathbf{LW}^{4,3} = \begin{bmatrix} -1.349 & 3.034 & 0.725 & -0.063 & 0.714 & -0.205 & -0.124 & 1.489 & 1.409 & 1.417 \end{bmatrix}$$

$$\mathbf{LW}^{2,1} = \begin{bmatrix} 0.537 & 1.833 & -2.258 & 0.862 & 0.318 & -1.307 & -0.433 & 0.342 & 3.578 & 2.769 \end{bmatrix}$$

$$\mathbf{LW}^{3,4} = \begin{bmatrix} 0.1319 & -0.999 & -0.354 & 0.1056 & 0.955 & 0.464 & 0.266 & -1.423 & 0.111 & 1.172 \end{bmatrix}^T$$

$$\mathbf{LW}^{3,4} = \begin{bmatrix} 0.671 & -1.207 & 0.717 & 1.630 & 0.488 & 1.0347 & 0.726 & -0.303 & 0.293 & -0.787 \\ 0.888 & -1.147 & -1.068 & -0.809 & -2.944 & 1.438 & 0.325 & -0.754 & 1.370 & -1.711 \end{bmatrix}^T$$

$$\mathbf{LW}^{1,2} = \begin{bmatrix} -0.102 & -0.241 & 0.319 & 0.312 & -0.864 & -0.030 & -0.164 & 0.627 & 1.093 & 1.109 \\ -0.863 & 0.077 & -1.214 & -1.113 & -0.006 & 1.532 & -0.769 & 0.371 & -0.225 & 1.117 \end{bmatrix}^T$$

$$\mathbf{LW}^{1,4} = \begin{bmatrix} -1.089 & 0.032 & 0.552 & 1.100 & 1.544 & 0.085 & -1.491 & -0.742 & -1.061 & 2.305 \\ -0.615 & 0.748 & -0.192 & 0.888 & -0.764 & -1.402 & -1.422 & 0.488 & -0.177 & -0.196 \end{bmatrix}^T$$

$$\mathbf{b}_3 = \begin{bmatrix} -0.839 & 1.354 & -1.072 & 0.961 & 0.124 & 1.436 & -1.960 & -0.197 & -1.207 & 2.908 \end{bmatrix}^T$$

$$\mathbf{b}_1 = \begin{bmatrix} 0.840 & -0.888 & 0.100 & -0.544 & 0.303 & -0.600 & 0.490 & 0.739 & 1.711 & -0.194 \end{bmatrix}^T$$

$$b_2 = -2.138$$

$$b_4 = 0.825$$

This is an example of a system in state space form with unstable equilibrium point. As we will show in the next chapter, this is an example of MRAC Control. Since (8.7) is not directly in the Lure form, the LMI method cannot be used to investigate the stability of the equilibrium point, whereas this form is suitable for all of the RODD methods. As expected, all the stability methods, RODD-LB2, RODD-EB and RODD-Hybrid, fail to detect

the GAS of the equilibrium point. A typical response of this system is shown in Figure (8.28).

Figure (8.29) shows the graph of β_k for RODD-Hybrid. Due to the instability of the equilibrium point, β_k would not get small enough to prove the stability of the equilibrium point. The algorithm stops when additional linear boundaries in RODD-LB2 can not make $\mathbf{D}_{k+1}$ smaller than $\mathbf{D}_k$. Table (8.7) illustrates the results of the stability analysis with different methods. RODD-Hybrid method is more efficient in failing to show stability of the equilibrium point compared with the other RODD methods. RODD-Hybrid failed to verify stability 13.4 times faster than RODD-LB2 and 11.1 times faster than RODD-EB.

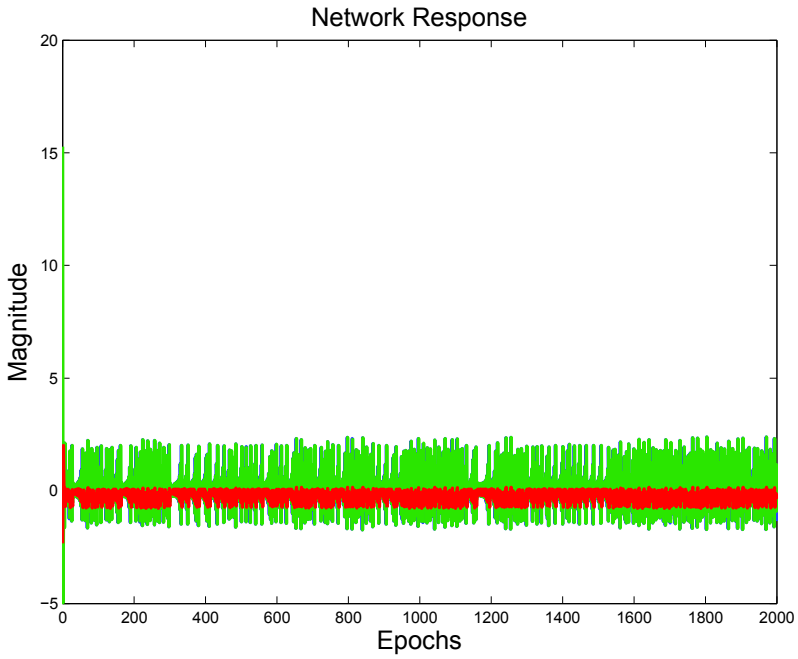

Figure (8.28) Network Response

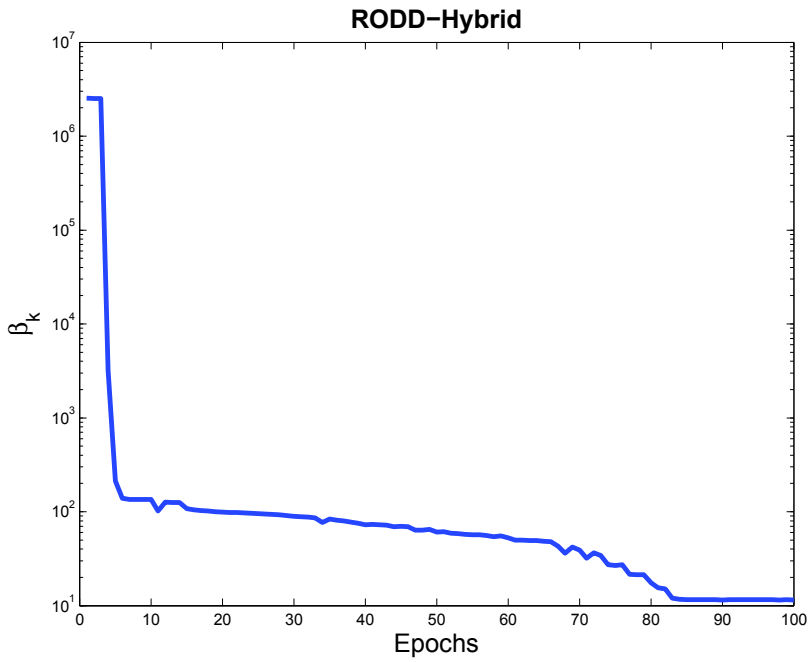

Figure (8.29) Graph of β_k for RODD-Hybrid

.

	LB2	EB	Hybrid
Result	Fail	Fail	Fail
Runtime(sec)	5249	4357	391
Improvement	13.4x	11.1x	Base

Table (8.7) Comparison RODD Methods

Conclusion

In this chapter, several test problems are used to investigate the performance of our proposed algorithms. The test problems are divided into stable and unstable groups. In each group, we provided some systems in Lure form and some systems in non-Lure form.

The main reason to provide systems in non-Lure form is to show that the LMI method cannot investigate the stability of these systems, whereas RODD methods can investi-

gate the stability of any system in state space form. This shows that RODD methods can be applied to a wider range of systems.

The tests described in this chapter demonstrate that the RODD-Hybrid method provides the most efficient operation. Often, at the initial steps of the algorithm, the reachable set is not well-approximated by elliptical contours. The hybrid method is able to switch to LB2 mode in these cases. In later stages, as the algorithm approaches the equilibrium point, the reachable set for many dynamical systems become approximately elliptical. At these points the hybrid method can switch to the EB mode.

In the next chapter, we will demonstrate the performance of RODD methods on MRAC and NARMA-L2 control systems which are not in Lure form.

9 CONTROL PROBLEMS

Chapter Overview

The previous chapters investigated different methods for stability analysis of RNNs. We developed three algorithms, RODD-LB2, RODD-EB and RODD-Hybrid, for detecting stability. In Chapter 8, the proposed methods have been applied to some test problems to check the efficiency of each algorithm. In this chapter, the proposed methods will be used to design stable RNN controllers for some real systems.

RNNs have been applied successfully in the system identification and control of dynamical systems ([HaDe99], [HuSb92], [HaDe02]). Rather than attempt to analyze many different neural network based controllers, we will concentrate on Model Reference Adaptive Control [NaPa90] and NARMA-L2 Control [NaMu97].

These controllers will be used to design a stable control systems, and stability will be investigated using the proposed algorithms.

Model Reference Adaptive Control

The Model Reference Adaptive Control (MRAC) system is shown in Figure (9.1). The neural network plant model is trained to match the plant response. Then the neural network controller is trained so that the closed loop system response matches the response of the reference model.

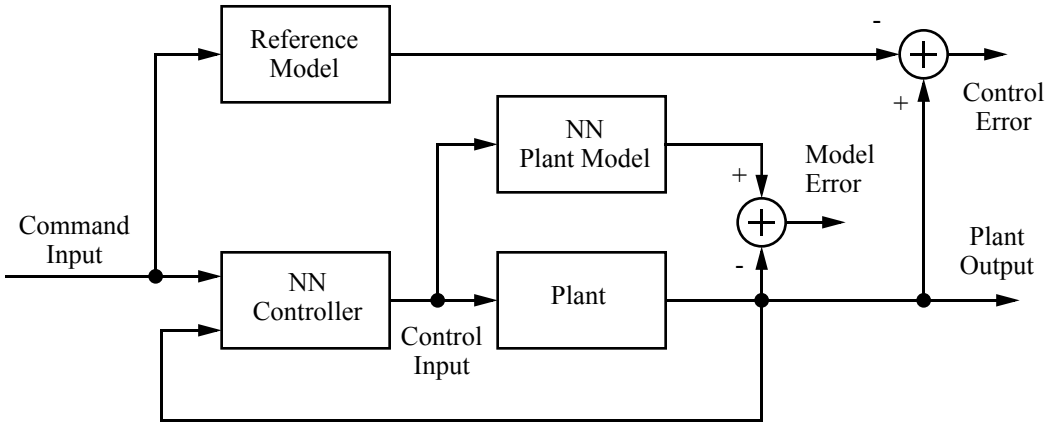

Figure (9.1) MRAC Architecture

In order to study the stability of the MRAC system, it is required that the overall system be represented in state space form.

In the following section, we will model the MRAC system in the state space form. Then we will apply the RODD methods to study the stability of the overall system.

Neural Network Model of MRAC

To perform stability analysis using RODD methods, the neural network model of the MRAC structure shown in Figure (9.2) needs to be converted into the state space form.

There are three signals that are input to Tapped Delayed Lines (TDL) in Figure (9.2). The first one is the reference signal $r(t)$, the second one is the input $u(t)$ and the third one the output $y(t)$.

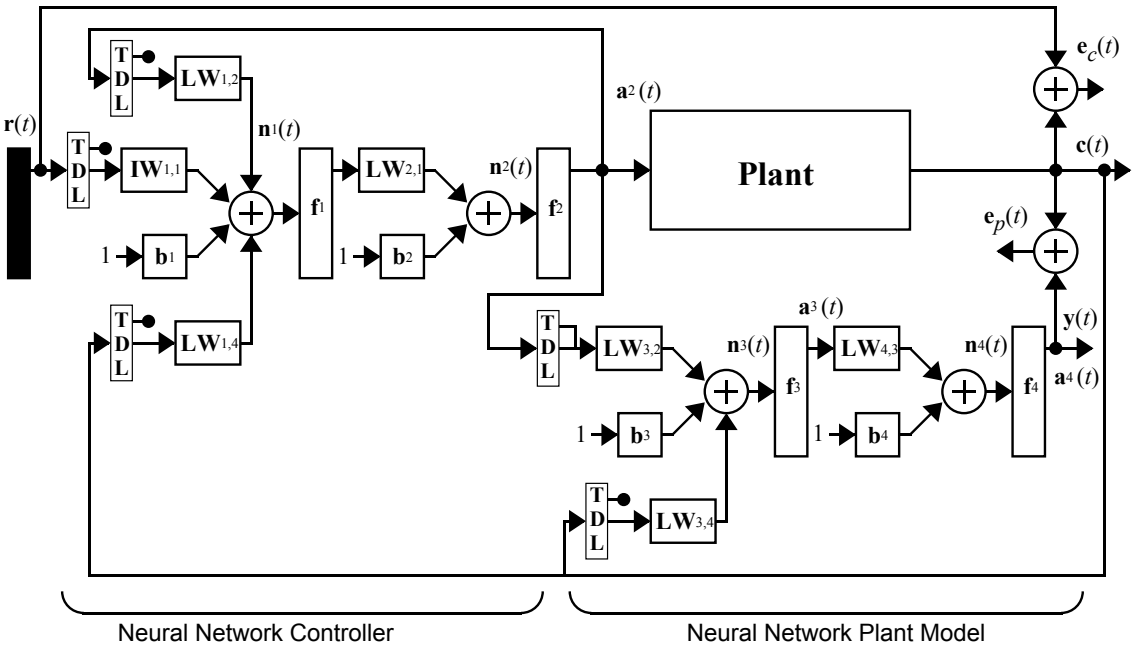

Figure (9.2) Neural Network Model of MRAC

Let m_c be the number of neurons in the RNN controller, m_p be the number of neurons in the RNN plant identifier, n_t be the maximum number of delays for the reference signal, n_u be the maximum number of input delays, n_y be the maximum number of output delays, n_r be the number of reference signals, n_i be the number of inputs and n_o be the number of outputs.

Define $\mathbf{T}(k) = \left[\mathbf{t}_1(k)\ \mathbf{t}_2(k)\ \ldots\ \mathbf{t}_{n_t}(k)\right]^T$ as the state vector for the reference signal. The state vector for the reference signal can be written in terms of previous reference signals as follows:

$$\begin{aligned}\mathbf{t}_1(k) &= \mathbf{r}(k-1)\\ \mathbf{t}_2(k) &= \mathbf{r}(k-2)\\ &\ldots\\ \mathbf{t}_{n_t}(k) &= \mathbf{r}(k-n_t)\end{aligned}$$

where $\mathbf{r}(k) \in \mathbf{R}^{n_r \times 1}$ .The state space representation of the reference signal can be written as

$$\mathbf{T}(k+1) = \begin{bmatrix}\mathbf{r}(k)\\ \mathbf{r}(k-1)\\ \mathbf{r}(k-2)\\ \ldots\\ \mathbf{r}(k-n_t+1)\end{bmatrix} \tag{9.1}$$

Similarly, define $\mathbf{Z}(k) = \left[\mathbf{z}_1(k)\ \mathbf{z}_2(k)\ \ldots\ \mathbf{z}_{n_u}(k)\right]^T$ as the state vector of the RNN controller. The state vector for the RNN controller can be written in terms of previous inputs as follows

$$\begin{aligned}\mathbf{z}_1(k) &= \mathbf{u}(k-1)\\ \mathbf{z}_2(k) &= \mathbf{u}(k-2)\\ &\ldots\\ \mathbf{z}_{n_u}(k) &= \mathbf{u}(k-n_u)\end{aligned}$$

According to Figure (9.2), and considering (9.1), the state space representation of the RNN controller can be written as

$$\mathbf{Z}(k+1) = \begin{bmatrix} \mathbf{u}(k) \\ \mathbf{u}(k-1) \\ \mathbf{u}(k-2) \\ \cdots \\ \mathbf{u}(k-n_u+1) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{LW}^{2,1}\tanh(\mathbf{LW}^{1,2}\mathbf{Z}(k) + \mathbf{LW}^{1,4}\mathbf{X}(k) + \mathbf{IW}^{1,1}\mathbf{T}(k) + \mathbf{b}^1) + \mathbf{b}^2 \\ \mathbf{z}_1(k) \\ \mathbf{z}_2(k) \\ \cdots \\ \mathbf{z}_{n_u-1}(k) \end{bmatrix} \tag{9.2}$$

where $\mathbf{u}(k) \in \mathbf{R}^{n_i \times 1}$, $\mathbf{LW}^{1,2} \in \mathbf{R}^{n_i \times m_c}$, $\mathbf{LW}^{2,1} \in \mathbf{R}^{m_c \times (n_i n_u)}$, $\mathbf{LW}^{1,4} \in \mathbf{R}^{m_c \times (n_y n_o)}$, $\mathbf{b}_1 \in \mathbf{R}^{m_c \times 1}$ and $\mathbf{b}_1 \in \mathbf{R}^{n_i \times 1}$.

Similarly, define $\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}_1(k) & \mathbf{x}_2(k) & \cdots & \mathbf{x}_{n_y}(k) \end{bmatrix}^T$ as the state vector of the RNN plant model. The state vector for the RNN plant model can be written in terms of previous outputs as follows

$$\begin{aligned} \mathbf{x}_1(k) &= \mathbf{y}(k-1) \\ \mathbf{x}_2(k) &= \mathbf{y}(k-2) \\ &\cdots \\ \mathbf{x}_{n_y}(k) &= \mathbf{y}(k-n_y) \end{aligned}$$

Hence the state space representation of the RNN plant model can be written as

$$\mathbf{X}(k+1) = \begin{bmatrix} \mathbf{y}(k) \\ \mathbf{y}(k-1) \\ \mathbf{y}(k-2) \\ \cdots \\ \mathbf{y}(k-n_y+1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}\mathbf{Z}(k) + \mathbf{LW}^{3,2}\mathbf{X}(k) + \mathbf{b}^3) + \mathbf{b}^4 \\ \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \\ \cdots \\ \mathbf{x}_{n_y-1}(k) \end{bmatrix} \quad (9.3)$$

where $\mathbf{y}(k) \in \mathbf{R}^{n_i \times 1}$, $\mathbf{LW}^{4,3} \in \mathbf{R}^{n_o \times m_p}$, $\mathbf{LW}^{3,2} \in \mathbf{R}^{m_p \times (n_i n_u)}$, $\mathbf{LW}^{3,4} \in \mathbf{R}^{m_p \times (n_y n_o)}$, $\mathbf{b}_3 \in \mathbf{R}^{m_p \times 1}$ and $\mathbf{b}_4 \in \mathbf{R}^{n_o \times 1}$.

The augmentation of (9.1), (9.2) and (9.3) is the state space representation of the overall system. The overall state space representation can be written as

$$\begin{bmatrix} \mathbf{T}(k+1) \\ \mathbf{X}(k+1) \\ \mathbf{Z}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{r}(k) \\ \mathbf{t}_1(k) \\ \mathbf{t}_2(k) \\ \cdots \\ \mathbf{t}_{n_t-1}(k) \\ \mathbf{LW}^{4,3}\tanh(\mathbf{LW}^{3,4}\mathbf{Z}(k) + \mathbf{LW}^{3,2}\mathbf{X}(k) + \mathbf{b}^3) + \mathbf{b}^4 \\ \mathbf{x}_1(k) \\ \mathbf{x}_2(k) \\ \cdots \\ \mathbf{x}_{n_y-1}(k) \\ \mathbf{LW}^{2,1}\tanh(\mathbf{LW}^{1,2}\mathbf{Z}(k) + \mathbf{LW}^{1,4}\mathbf{X}(k) + \mathbf{IW}^{1,1}\mathbf{T}(k) + \mathbf{b}^1) + \mathbf{b}^2 \\ \mathbf{z}_1(k) \\ \mathbf{z}_2(k) \\ \cdots \\ \mathbf{z}_{n_u-1}(k) \end{bmatrix} \quad (9.4)$$

Equation (9.4) is the state space representation of the MRAC. This is suitable for stability analysis with RODD methods. In the following example, stability of the MRAC for a robot arm problem will be investigated using RODD methods.

Robot Arm Problem

Consider the robot arm shown in Figure (9.3). The objective is to design a stable RNN based control system so that the robot arm will follow a certain trajectory. In order to accomplish this goal, we first need to model the robot arm system using an Nonlinear-Auto-Regressive with Exogenous input (NARX) model. Then, an RNN controller needs to be designed. The stability of the overall system will be confirmed by the RODD methods.

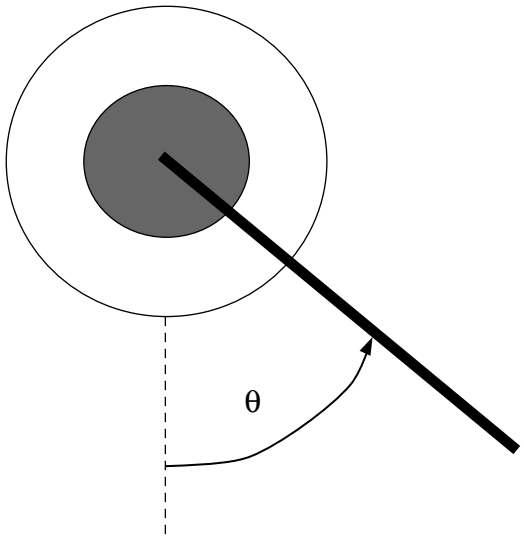

Figure (9.3) Robot Arm [BeHa12]

The NARX model shown in Figure (9.4) is used to model the robot arm. In this model, we used two input delays, two output delays, 10 neurons in the hidden layer, with a hyperbolic tangent activation function. For the training data set, we generated a sequence with 1463 data points using Matlab/Simulink. The training was done with the neural net-

work toolbox in Matlab ([BeHa12]). Figure (9.5) shows the input and target data that was used to train the network. The input is the voltage (between -10 and 10 volts) and the target is the angular position.

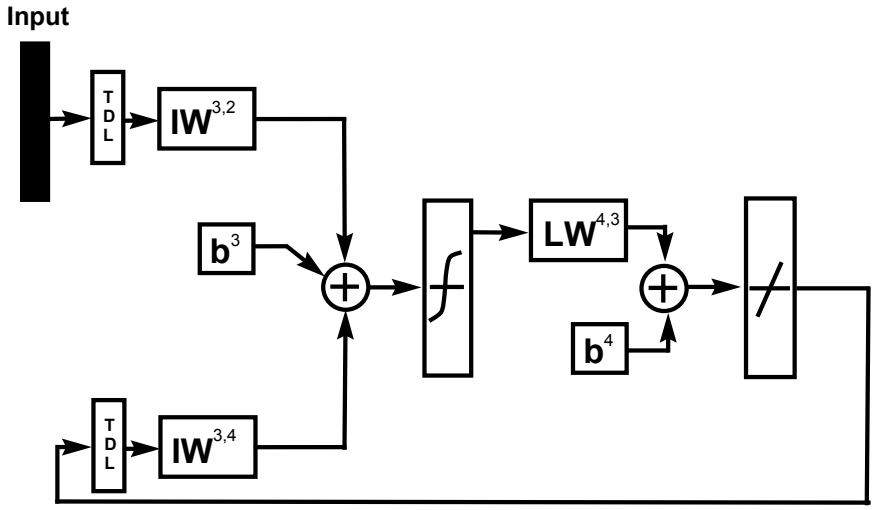

Figure (9.4) NARX Model of Robot Arm

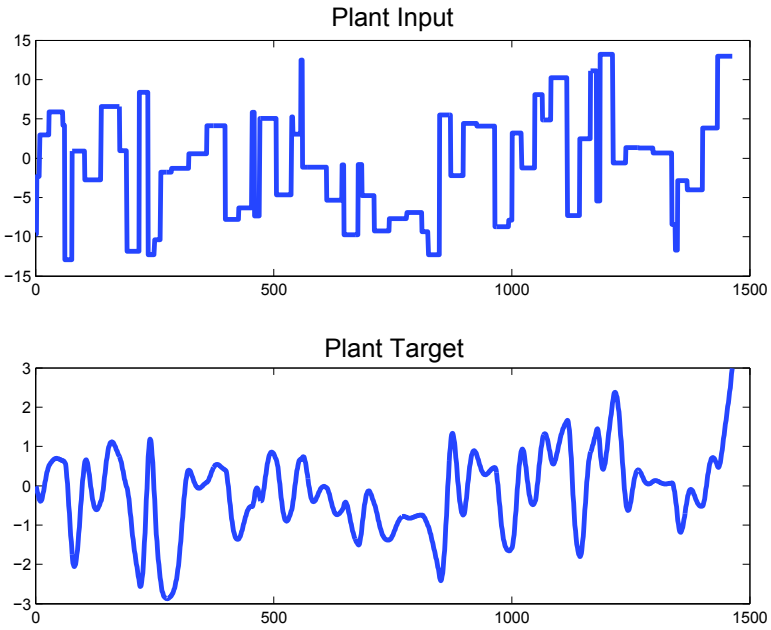

Figure (9.5) Training Data Set

Figure (9.6) and Figure (9.7) show the response of the network versus the target and the error between them after training. The trained network, which is now equivalent to the robot arm, can be written in state space form (derived in (9.3)) as follows:

$$\mathbf{X}(k+1) = \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{4,3}\tanh(\mathbf{IW}^{3,2}\mathbf{Z}(k) + \mathbf{IW}^{3,4}\mathbf{X}(k) + \mathbf{b}^3) + b^4 \\ x_1(k) \end{bmatrix} \quad (9.5)$$

where

$$
\begin{aligned}
\mathbf{LW}^{4,3} &= \begin{bmatrix} 0.82 & -1.64 & -3.90 & -0.00 & -0.00 & -2.36 & -1.02 & -0.73 & 1.12 & 0.00 \end{bmatrix} \\
\mathbf{IW}^{3,4} &= \begin{bmatrix} -0.45 & -0.35 & -0.48 & 0.28 & -0.64 & -0.26 & -0.58 & -0.65 & 0.70 & 0.27 \\ -0.00 & -0.02 & 0.26 & -0.82 & -0.20 & 0.04 & 0.37 & -0.33 & -0.43 & 0.55 \end{bmatrix}^T \\
\mathbf{IW}^{3,2} &= \begin{bmatrix} -0.00 & -0.00 & -0.00 & -0.06 & 0.06 & -0.00 & -0.00 & 0.00 & 0.00 & -0.00 \\ -0.00 & -0.00 & -0.00 & 0.00 & -0.36 & 0.25 & -0.00 & -0.00 & 0.00 & 0.00 \end{bmatrix}^T \\
\mathbf{b}^3 &= \begin{bmatrix} 1.53 & 1.31 & 0.75 & 1.29 & 0.06 & -0.75 & -0.48 & -0.86 & 1.80 & -0.39 \end{bmatrix}^T \\
b^4 &= -0.39
\end{aligned}
\tag{9.6}
$$

After deriving the RNN model of the robot arm in (9.5), an RNN controller needs to be designed. The RNN controller is another NARX model. The RNN controller and the RNN robot arm will cascade to form a four layer network as shown in Figure (9.2). (The weight of the RNN robot arm will not be adjusted during the training of the RNN controller.) For the training of the RNN controller, we used one delay for the reference input $n_r = 2$, one delay for the control signal $n_u = 2$, two output delays $n_y = 2$, 10 neurons in the hidden layer, with hyperbolic tangent activation function. The overall network has been trained with 2200 data points.

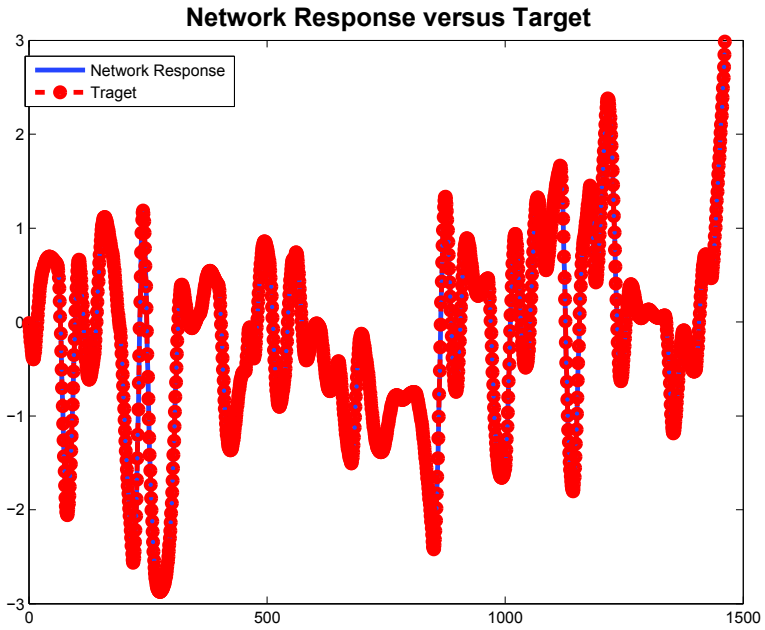

Figure (9.6) Network Response and Target after Training

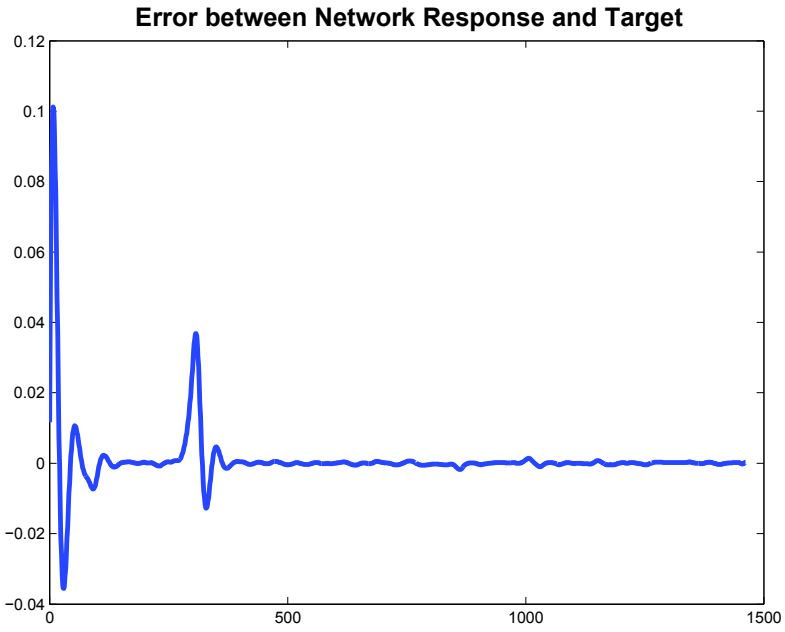

Figure (9.7) Error between Target and Network Response After Training

The target data has been generated in Matlab/Simulink using a first order reference model.

Figure (9.8) illustrates the reference input and reference output that have been used to train the RNN controller. The training was done with the neural network toolbox in Matlab ([BeHa12]). The network response after training is shown in Figure (9.9).

According to (9.4) the overall RNN system can be written as

$$\begin{bmatrix} \mathbf{T}(k+1) \\ \mathbf{X}(k+1) \\ \mathbf{Z}(k+1) \end{bmatrix} = \begin{bmatrix} r(k) \\ t_1(k) \\ \mathbf{LW}^{4,3}\tanh(\mathbf{IW}^{3,2}\mathbf{Z}(k) + \mathbf{IW}^{3,4}\mathbf{X}(k) + \mathbf{b}^3) + b^4 \\ x_1(k) \\ \mathbf{LW}^{2,1}\tanh(\mathbf{LW}^{1,2}\mathbf{Z}(k) + \mathbf{LW}^{1,4}\mathbf{X}(k) + \mathbf{IW}^{1,1}\mathbf{T}(k) + \mathbf{b}^1) + \mathbf{b}^2 \\ z_1(k) \end{bmatrix} \tag{9.7}$$

where $\mathbf{T}(k) = \begin{bmatrix} r(k-1) & r(k-2) \end{bmatrix}^T$, $\mathbf{X}(k) = \begin{bmatrix} y(k-1) & y(k-2) \end{bmatrix}^T$,

$\mathbf{Z}(k) = \begin{bmatrix} z(k-1) & z(k-2) \end{bmatrix}^T$ and

$$\mathbf{LW}^{2,1} = \begin{bmatrix} 0.04 \\ -10.21 \\ -17.08 \\ -23.79 \\ 21.59 \\ 25.95 \\ -19.50 \\ 0.04 \\ -17.12 \\ -0.05 \end{bmatrix}^T, \mathbf{LW}^{1,2} = \begin{bmatrix} -2.22 & -0.94 \\ 0.01 & 0.01 \\ 8.45 & 7.60 \\ 0.02 & 0.00 \\ -0.03 & -0.00 \\ -0.02 & -0.01 \\ 0.01 & 0.00 \\ 1.76 & 0.91 \\ -8.38 & -7.51 \\ 1.41 & 4.47 \end{bmatrix}, \mathbf{LW}^{1,4} = \begin{bmatrix} 2.41 & 1.12 \\ -0.23 & -1.10 \\ 7.91 & -7.96 \\ 5.20 & -5.06 \\ -5.07 & 4.88 \\ -5.95 & 5.69 \\ -2.63 & 2.40 \\ -0.86 & -1.19 \\ 6.94 & -7.05 \\ 0.77 & -1.68 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} -11.26 \\ 0.00 \\ -0.62 \\ 1.37 \\ 1.15 \\ -0.05 \\ 0.00 \\ 9.85 \\ 0.62 \\ 6.37 \end{bmatrix}$$

$$\mathbf{IW}^{1,1} = \begin{bmatrix} 4.46 & -0.30 & 0.06 & -0.66 & 0.58 & 0.54 & -0.29 & -0.98 & -0.03 & -5.83 \\ 9.92 & 0.10 & -0.80 & 0.18 & -0.21 & -0.17 & -0.01 & -1.05 & -0.79 & -7.57 \end{bmatrix}^T$$

$$b^2 = 4.76$$

and $\mathbf{LW}^{4,3}$, $\mathbf{LW}^{3,4}$, $\mathbf{IW}^{3,2}$, $\mathbf{b}^3$ and b^4 are the parameters of the robot arm model, which are not adjusted during the RNN controller training.

The state space representation shown in is suitable for stability analysis using RODD methods. The response of the overall system, RNN controller and RNN robot arm, after training is shown in Figure (9.10). This figure shows the response of the state space equation for an arbitrary initial condition without reference input. All the states converge to the origin. In order to confirm stability of the equilibrium point of , RODD methods need to be applied to this equation. Figure (9.11) shows the graph of β_k for RODD-LB2. Since the equilibrium point of this system is stable, the graph of β_k could become small enough to stop the algorithm.

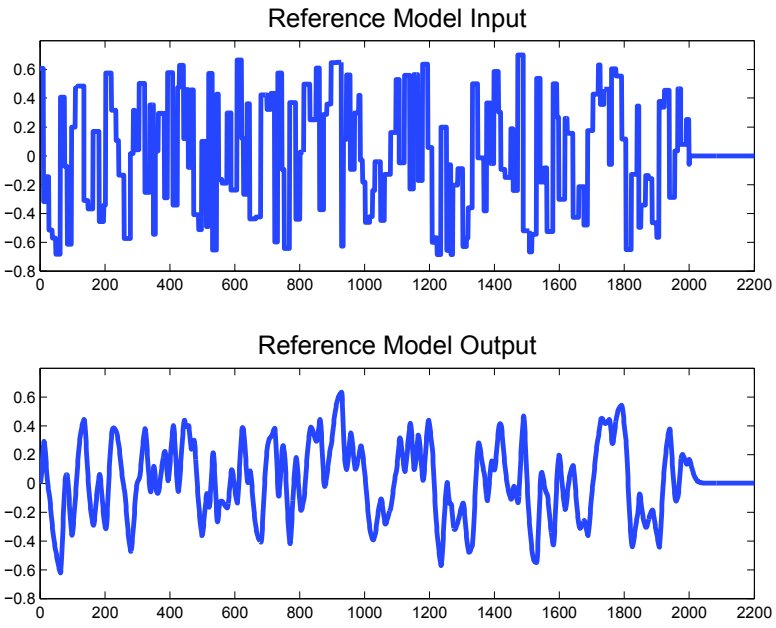

Figure (9.8) Reference Input and Output

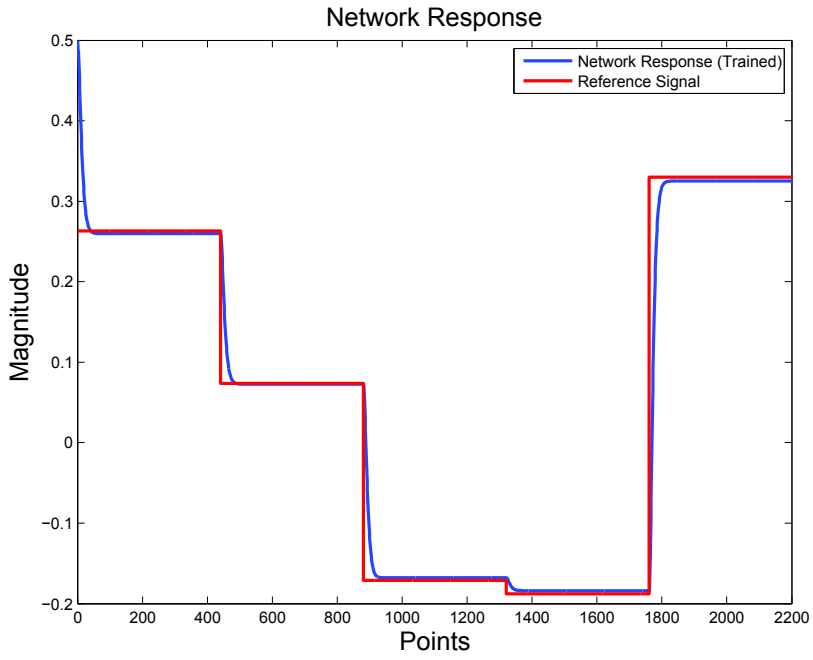

Figure (9.9) Reference Signal versus Network Response After Training

In this example, it is important to mention that the initial set $\mathbf{D}_0$ in all of the RODD methods needs to be placed in region in which the network has been trained. Otherwise, the stability results would not be accurate. Table (9.1) illustrates the result of the stability analysis for different methods. Neither RODD-EB nor RODD-Hyrbid were able to confirm stability of the equilibrium point. This might be due to the non-elliptical shape of the reachable set. RODD-LB2 is more efficient in detecting stability of the equilibrium point compared to RODD-EB and RODD-Hybrid..

	LB2	EB	Hybrid
Result	Pass	Fail	Fail
Runtime(sec)	2020	14400	1115
Improvement	Base	NA	NA

Table (9.1) Comparison of RODD Methods

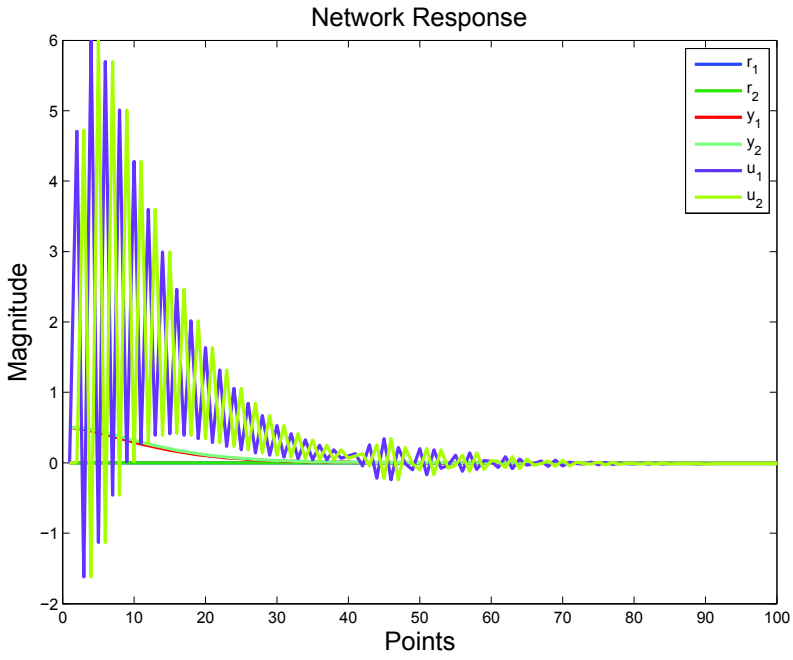

Figure (9.10) MRAC Response After Training

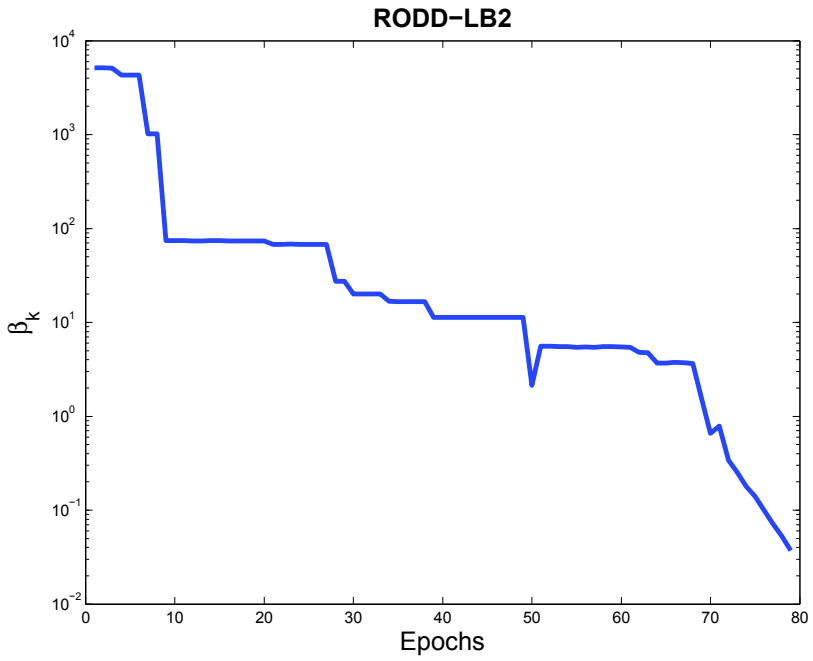

Figure (9.11) Graph of β_k for RODD-LB2

The following section discusses the RNN-based NARMA-L2 controller. The stability of this controller will be investigated using RODD methods.

NARMA-L2 Control

The NARMA-L2 model is an approximation of the NARX model explained in Chapter 2 and derived in (2.34). This model is very useful in control, since the control signals at each time step can be solved linearly with respect to the reference signal. The first step in using NARMA-L2 control is to identify the system to be controlled. The NARMA-L2 approximate model is given by

$$\begin{aligned} y(k+d) = {} & f(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1)) + \\ & g(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1))u(k) \end{aligned} \tag{9.8}$$

where f and g are nonlinear functions. Once the NARMA-L2 model of the plant is derived, the NARMA-L2 controller is just a rearrangement of the NARMA-L2 model. This is because of (9.8) where the control signal can be solved linearly with respect to the reference input. The control signal should be calculated such that the system response follows the reference input.

Let $r(k)$ be the reference input. Considering the NARMA-L2 model (9.8), the control system $u(k)$ can be derived as

$$u(k) = \frac{r(k+1) - f(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1))}{g(y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1))} \tag{9.9}$$

This control signal $u(k)$ will guarantee to make the system response follow the desired trajectory $r(k)$ if the exact knowledge of f and g are available. However, this knowledge is not available, and so we must approximate f and g. Figure (9.12) illustrates the block dia-

gram of the NARMA-L2 control scheme.

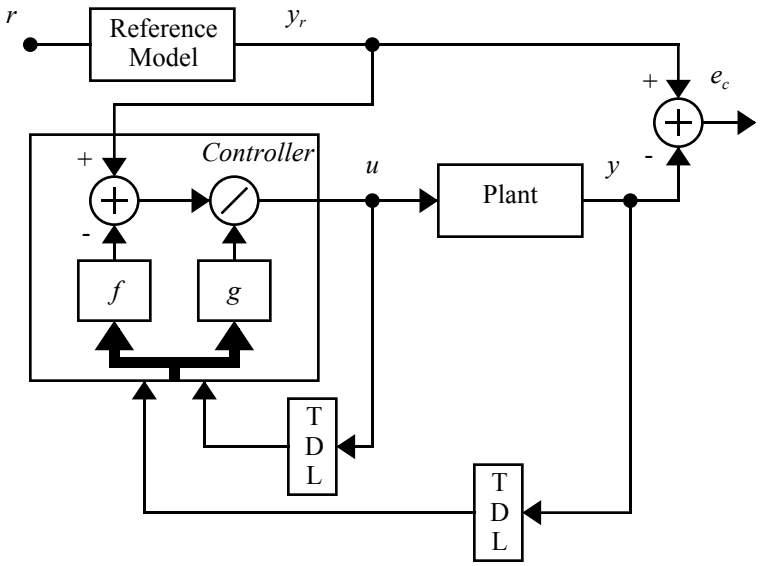

Figure (9.12) NARMA-L2 Controller

In the following section, we will use neural network approximations for f and g. The RNN-based NARMA-L2 controller will be represented in state space form, which is suitable for stability analysis using RODD methods.

Neural Network Model of NARMA-L2 Control

The nonlinear functions f and g in (9.9) can be approximated by neural networks. An RNN-based NARMA-L2 model is illustrated in Figure (9.13). Since the NARMA-L2 controller is just a rearrangement of the plant model, controlling the NARMA-L2 model of the plant gives a perfect result. To avoid this, and, to verify the performance of the NARMA-L2 control, the plant will be modeled using a NARX model. The error between the actual plant and the NARX model can verify the performance of the NARMA-L2 controller.

In order to model the RNN-based controller and RNN-based plant, let m_c be the number of neurons in the RNN-based NARMA-L2 controller, m_p be the number of neu-

rons in the RNN-based NARX plant, n_u be the maximum number of input delays and n_y be the maximum number of output delays. Unlike the MRAC, which is a multi-input-multi-output controller, the NARMA-L2 control is a single-input-single-output control. Figure (9.14) shows the block diagram of the NARMA-L2 controller with the NARX model of the plant.

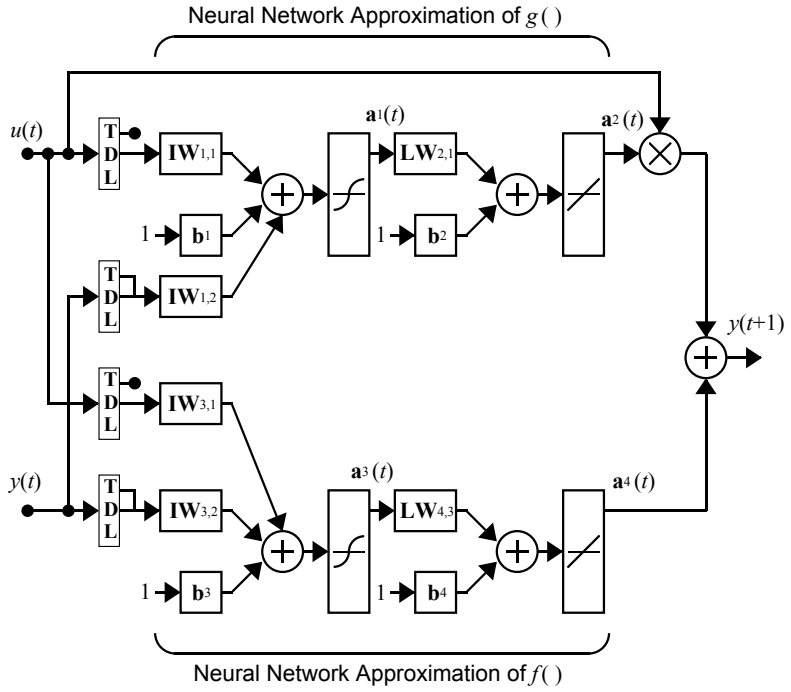

Figure (9.13) RNN-Based NARMA-L2 Model

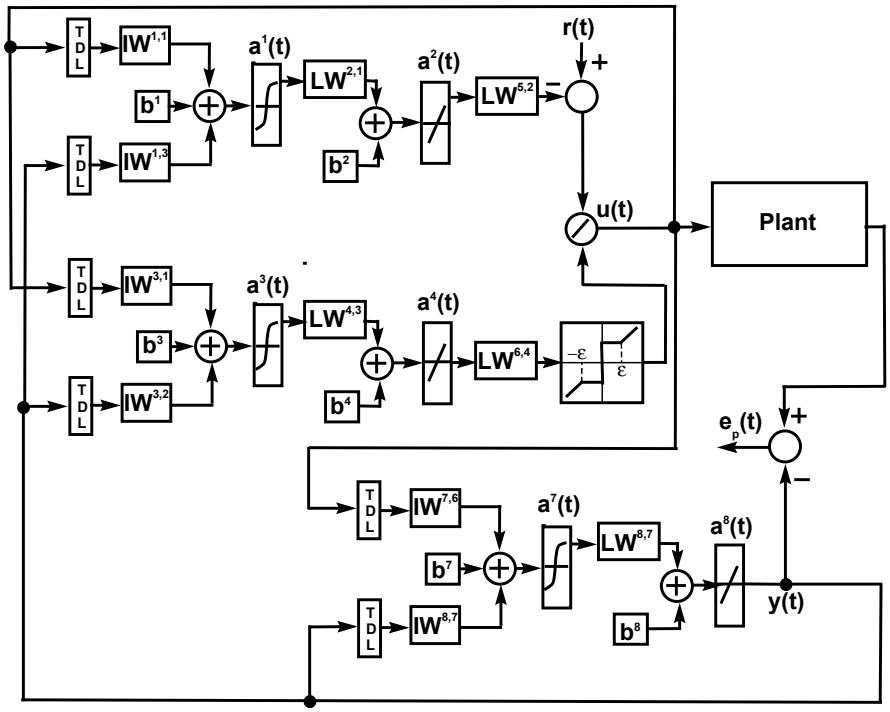

Figure (9.14) NARMA-L2 Controller and NARX Plant

Define $\mathbf{z}(k) = \left[z_1(k)\ z_2(k)\ \ldots\ z_{n_u}(k)\right]^T$ as the state vector of the RNN controller and

$\mathbf{x}(k) = \left[x_1(k)\ x_2(k)\ \ldots\ x_{n_y}(k)\right]^T$ as the state vector of the RNN plant model. The state vector for the RNN controller can be written in terms of previous inputs as follows:

$$\begin{aligned} z_1(k) &= u(k-1) \\ z_2(k) &= u(k-2) \\ &\cdots \\ z_{n_u}(k) &= u(k-n_u) \end{aligned} \tag{9.10}$$

and the state vector for the RNN plant model can be written in terms of previous outputs as follows:

$$\begin{aligned} x_1(k) &= y(k-1) \\ x_2(k) &= y(k-2) \\ &\cdots \\ x_{n_u}(k) &= y(k-n_y) \end{aligned} \tag{9.11}$$

According to (9.10) the state space representation of the RNN controller can be written as:

$$\mathbf{z}(k+1) = \begin{bmatrix} u(k) \\ u(k-1) \\ u(k-2) \\ \cdots \\ u(k-n_u+1) \end{bmatrix} =$$

$$= \begin{bmatrix} \dfrac{r(k) - \mathbf{LW}^{5,2}(\mathrm{LW}^{2,1}\tanh(\mathbf{IW}^{1,3}\mathbf{z}(k) + \mathbf{IW}^{1,1}\mathbf{x}(k) + \mathbf{b}^1) + b^2)}{\mathbf{LW}^{6,4}(\mathbf{LW}^{4,3}\tanh(\mathbf{IW}^{3,1}\mathbf{z}(k) + \mathbf{IW}^{3,2}\mathbf{x}(k) + \mathbf{b}^3) + b^4)} \\ z_1(k) \\ z_2(k) \\ \cdots \\ z_{n_u-1}(k) \end{bmatrix} \tag{9.12}$$

where $r(k) \in \mathbf{R}$, $\mathrm{LW}^{5,2} \in \mathbf{R}$, $\mathbf{LW}^{2,1} \in \mathbf{R}^{1 \times m_c}$, $\mathbf{IW}^{1,3} \in \mathbf{R}^{m_c \times n_y}$, $\mathbf{IW}^{1,1} \in \mathbf{R}^{m_c \times n_u}$, $\mathbf{LW}^{6,4} \in \mathbf{R}$, $\mathbf{LW}^{4,3} \in \mathbf{R}^{1 \times m_c}$, $\mathbf{IW}^{3,1} \in \mathbf{R}^{m_c \times n_u}$, $\mathbf{IW}^{3,2} \in \mathbf{R}^{m_c \times n_y}$, $\mathbf{b}^1 \in \mathbf{R}^{m_c \times 1}$, $\mathbf{b}^3 \in \mathbf{R}^{m_c \times 1}$, $b^2 \in \mathbf{R}$ and $b^4 \in \mathbf{R}$

According to (9.11) the state space representation of RNN plant identifier can be written as:

$$\mathbf{x}(k+1) = \begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \\ \cdots \\ y(k-n_y+1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{8,7}\tanh(\mathbf{IW}^{7,6}\mathbf{z}(k) + \mathbf{IW}^{8,7}\mathbf{x}(k) + \mathbf{b}^7) + b^8 \\ x_1(k) \\ x_2(k) \\ \cdots \\ x_{n_u-1}(k) \end{bmatrix} \quad (9.13)$$

where $y(k) \in \mathbf{R}$, $\mathbf{LW}^{8,7} \in \mathbf{R}^{1 \times m_p}$, $\mathbf{IW}^{7,6} \in \mathbf{R}^{m_p \times n_u}$, $\mathbf{IW}^{8,7} \in \mathbf{R}^{m_p \times n_y}$, $\mathbf{b}^7 \in \mathbf{R}^{m_p \times 1}$ and $b^8 \in \mathbf{R}$.

The augmentation of (9.12) and (9.13) derives the state space representation of the overall system. The overall state space representation can be written as

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \mathbf{z}(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{8,7}\tanh(\mathbf{IW}^{7,6}\mathbf{z}(k) + \mathbf{IW}^{8,7}\mathbf{x}(k) + \mathbf{b}^7) + b^8 \\ x_1(k) \\ x_2(k) \\ \cdots \\ x_{n_u-1}(k) \\ \dfrac{r(k) - \mathbf{LW}^{5,2}(\mathbf{LW}^{2,1}\tanh(\mathbf{IW}^{1,1}\mathbf{z}(k) + \mathbf{IW}^{1,3}\mathbf{x}(k) + \mathbf{b}^1) + b^2)}{\mathbf{LW}^{6,4}(\mathbf{LW}^{4,3}\tanh(\mathbf{IW}^{3,1}\mathbf{z}(k) + \mathbf{IW}^{3,2}\mathbf{x}(k) + \mathbf{b}^3) + b^4)} \\ z_1(k) \\ z_2(k) \\ \cdots \\ z_{n_u-1}(k) \end{bmatrix} \quad (9.14)$$

Equation (9.14) is the state space representation of the RNN-based NARMA-L2 controller with an RNN-based plant. This is suitable for stability analysis using RODD methods. In the following example, a NARMA-L2 control will be designed for a magnetic levitation problem and the stability will be investigated using RODD methods.

Magnetic Levitation Problem

Magnetic levitation is a system which is used to levitate a magnet using the electromagnetic force generated by an electromagnet. The objective is to design a stable RNN-based NARMA-L2 controller that levitates the magnet and makes it track a specified position trajectory. The input to this system is the electromagnet current and the output is the position of the magnet. The schematic view of the magnetic levitation problem is illustrated in Figure (9.15).

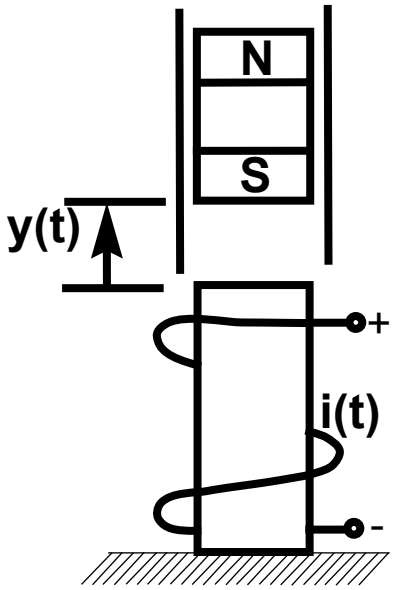

Figure (9.15) Magnetic Levitation [BeHa12]

In order to accomplish this goal, the magnetic levitation dynamics need to be modelled using an RNN. Then, an RNN-based NARMA-L2 controller needs to be designed. The stability of the overall system will be confirmed using RODD methods.

For the modeling of the magnetic levitation, we used the NARX model shown in Figure (9.16). In this model, we used two input delays, two output delays, 10 neurons in

the hidden layer, with a hyperbolic tangent activation function. For the training data set, we combined 4 sequences with different frequencies. The combination of sequences with low, high and middle frequencies enables us to derive a model which is valid for a wider operating range. Each sequence has been generated for a duration of 50 seconds with a sampling rate of 50 milliseconds The training data has been generated in Matlab/Simulink and the training has been done with the neural network toolbox in Matlab ([BeHa12]). Figure (9.17) shows the input and the target data which have been used to train the network.

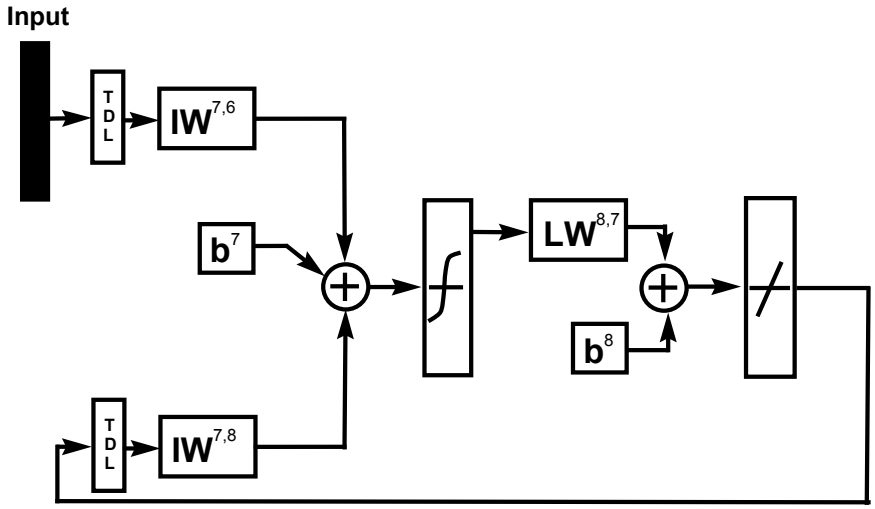

Figure (9.16) NARX Model of Magnetic Levitation

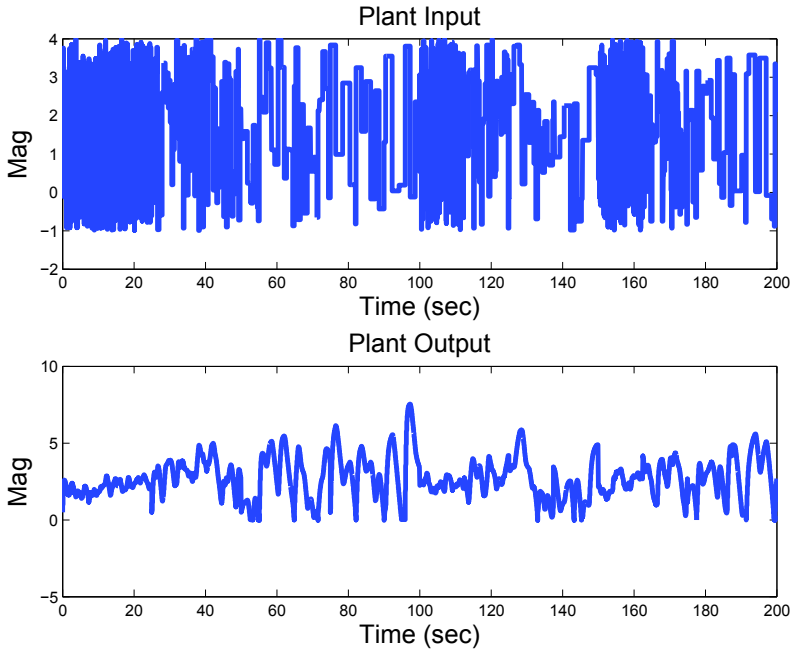

Figure (9.17) Sample Input and Output Signals for Identification

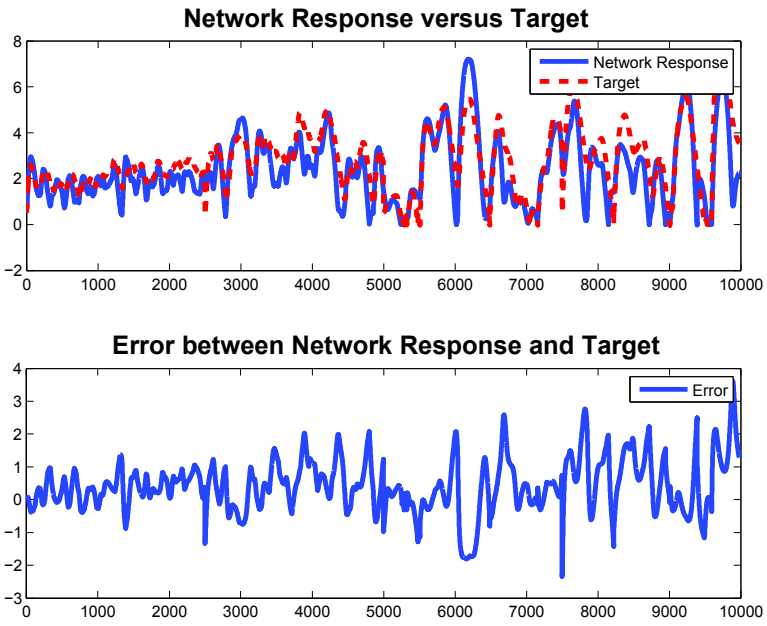

Figure (9.18) Network Response versus Target After Training

Figure (9.18) shows the response of the network versus the target and the error between them after training. The trained network, which is now equivalent to the magnetic levitation system, can be written in state space form as follows:

$$\mathbf{x}(k+1) = \begin{bmatrix} y(k) \\ y(k-1) \end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{8,7}\tanh(\mathbf{IW}^{7,6}\mathbf{z}(k) + \mathbf{IW}^{7,8}\mathbf{x}(k) + \mathbf{b}^{7}) + b^{8} \\ x_1(k) \end{bmatrix} \quad (9.15)$$

where

$$\begin{aligned}
\mathbf{LW}^{8,7} &= \begin{bmatrix} 0.50 & -1.04 & 0.75 & 0.29 & 0.00 & 0.00 & 57.10 & -0.01 & 15.38 & -7.25 \end{bmatrix} \\
\mathbf{IW}^{7,8} &= \begin{bmatrix} -13.11 & 1.75 & 44.00 & 14.96 & 15.28 & 7.99 & -0.07 & 17.80 & 0.18 & -0.23 \\ 81.93 & -8.62 & -140.30 & -15.33 & 19.39 & 5.93 & 0.08 & -7.37 & -0.17 & 0.21 \end{bmatrix}^{T} \\
\mathbf{IW}^{7,6} &= \begin{bmatrix} 1.44 & 0.94 & -0.85 & 0.06 & -18.57 & -11.43 & 0.01 & 2.27 & -0.03 & 0.05 \\ -0.58 & -1.67 & 0.70 & 0.01 & -7.08 & 0.36 & -0.02 & -3.37 & 0.06 & -0.09 \end{bmatrix}^{T} \\
\mathbf{b}^{7} &= \begin{bmatrix} -0.79 & -0.46 & 3.13 & -0.06 & -16.02 & -6.92 & 6.60 & -2.10 & -5.85 & 1.17 \end{bmatrix}^{T} \\
b^{8} &= 4.67
\end{aligned} \quad (9.16)$$

We also verified that the derived RNN model of the magnetic levitation system is stable. The response of this model for an arbitrary initial condition without input can been seen in Figure (9.19). The confirmation of the stability of this model is achieved by applying RODD-LB2 to the state space equation of the magnetic levitation problem derived in (9.15). Figure (9.20) shows the graph of β_k for RODD-LB2. Since the equilibrium point of this system is stable, β_k could become small enough to stop the algorithm.

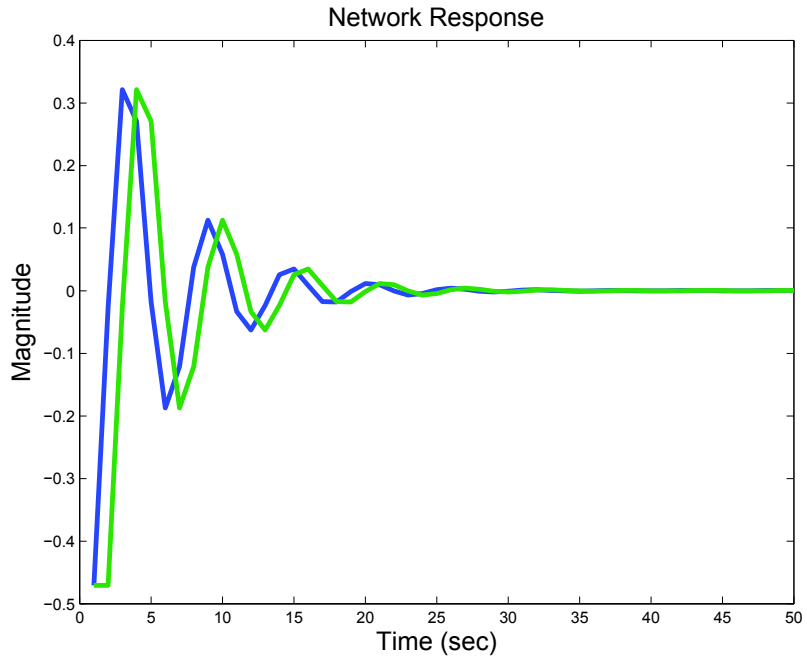

Figure (9.19) Network Response

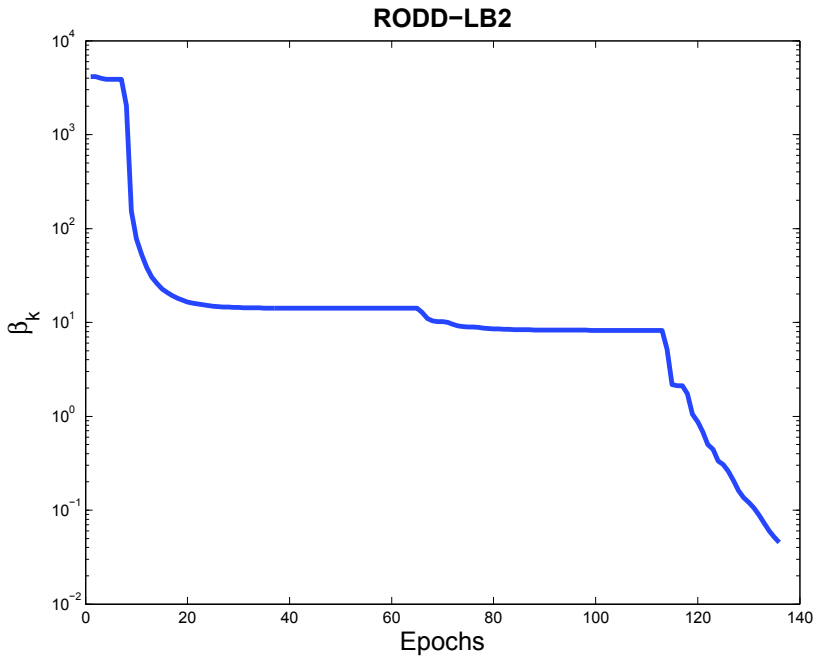

Figure (9.20) Graph of β_k for RODD-LB2

The next step is designing an RNN-based NARMA-L2 controller. This can be done by finding the NARMA-L2 model of the magnetic levitation system. The rearrangement of this model gives the NARMA-L2 controller. Figure (9.21) shows the block diagram of the NARMA-L2 model which has been used to model the magnetic levitation system. In this model, we used two input delays, five output delays, 10 neurons in the hidden layers, with a hyperbolic tangent activation function. For the training data set, we used the same data shown in Figure (9.17). The training was done with neural network toolbox in Matlab ([BeHa12]).

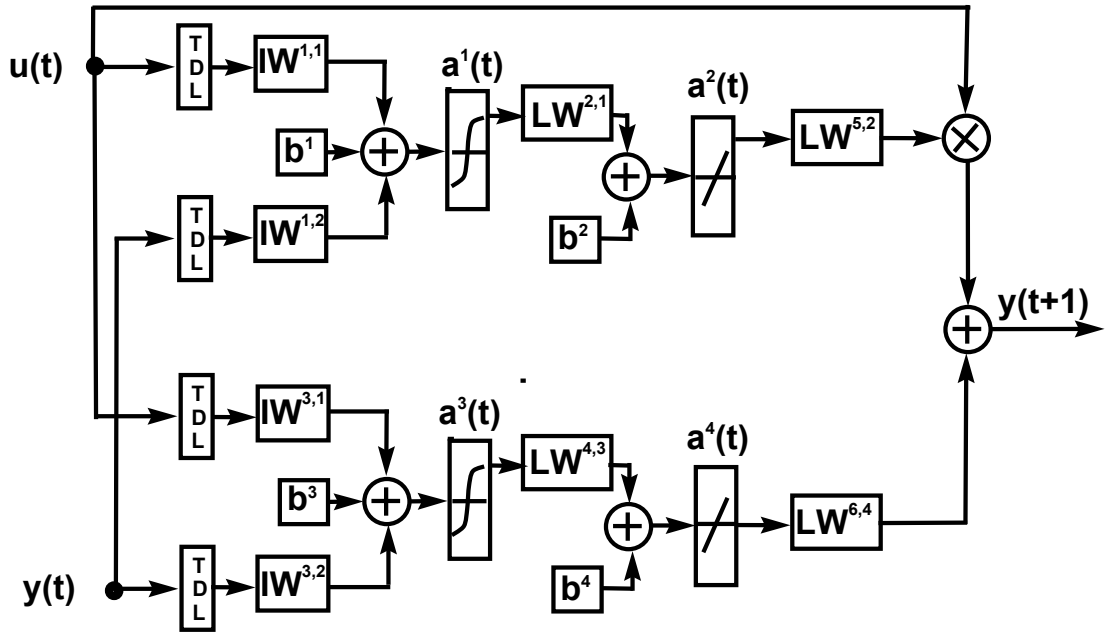

Figure (9.21) NARMA-L2 Model of Magnetic Levitation

After deriving the RNN-based NARMA-L2 model of magnetic levitation, the over-all closed loop control system with NARX plant model (shown in Figure (9.14)) can be written in the state space form as follows:

$$\begin{bmatrix}\mathbf{x}(k+1)\\ \mathbf{z}(k+1)\end{bmatrix} = \begin{bmatrix} \mathbf{LW}^{8,7}\tanh(\mathbf{IW}^{7,6}\mathbf{z}(k)+\mathbf{IW}^{8,7}\mathbf{x}(k)+\mathbf{b}^7)+b^8 \\ x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ \dfrac{r(k)-\mathbf{LW}^{5,2}(\mathbf{LW}^{2,1}\tanh(\mathbf{IW}^{1,1}\mathbf{z}(k)+\mathbf{IW}^{1,3}\mathbf{x}(k)+\mathbf{b}^1)+b^2)}{\mathbf{LW}^{6,4}(\mathbf{LW}^{4,3}\tanh(\mathbf{IW}^{3,1}\mathbf{z}(k)+\mathbf{IW}^{3,2}\mathbf{x}(k)+\mathbf{b}^3)+b^4)} \\ z_1(k) \end{bmatrix} \tag{9.17}$$

where

$$\mathbf{LW}^{2,1} = \begin{bmatrix}-0.81 & -0.20 & 1.70 & -0.00 & -1.53 & -0.00 & 0.16 & -2.10 & 0.02 & 0.00\end{bmatrix}$$

$$\mathbf{IW}^{4,3} = \begin{bmatrix}0.92 & 0.67 & 0.45 & 0.90 & 0.50 & -0.07 & 0.09 & -0.59 & 0 - 1.07 & -0.06\end{bmatrix}$$

$$\mathbf{IW}^{1,3} = \begin{bmatrix} 0.24 & -0.52 & -0.07 & 1.38 & -0.10 & 3.18 & 0.21 & -0.01 & 0.43 & 0.50 \\ -0.05 & 0.14 & -0.00 & 1.75 & 0.01 & -0.56 & -0.10 & -0.01 & -1.00 & 1.24 \\ 0.58 & 0.36 & 0.81 & -1.41 & -0.38 & -0.57 & -0.47 & -0.63 & -0.63 & 1.17 \\ -0.68 & -0.59 & -0.51 & -0.12 & -0.29 & -1.72 & -1.13 & 1.12 & -1.38 & 1.18 \\ 0.74 & 0.68 & -0.80 & 0.24 & 1.59 & -1.15 & -1.05 & 0.30 & 0.44 & 1.03 \end{bmatrix}^T$$

$$\mathbf{IW}^{3,2} = \begin{bmatrix} 1.47 & 0.97 & 0.30 & 1.57 & -0.22 & 0.08 & -0.30 & -0.29 & -0.24 & 0.97 \\ -0.67 & -0.10 & 1.08 & -0.33 & 0.63 & -1.90 & 0.82 & -0.09 & 1.27 & -1.32 \\ 0.68 & -0.98 & 1.37 & -0.96 & -0.97 & -0.12 & -1.57 & -0.53 & -0.72 & 1.19 \\ 1.19 & -0.16 & 1.03 & -1.27 & -1.60 & 0.07 & 1.22 & 1.34 & 1.04 & -0.82 \\ -0.83 & 1.78 & 0.19 & -0.07 & 0.92 & 1.09 & 0.28 & -1.64 & -1.07 & 0.37 \end{bmatrix}^T$$

$$\mathbf{b}^1 = \begin{bmatrix}2.38 & 1.99 & -0.91 & -2.48 & -0.23 & 0.47 & -0.78 & 0.36 & -0.51 & 0.76\end{bmatrix}^T$$

$$\mathbf{b}^3 = \begin{bmatrix}2.21 & -1.85 & -1.34 & 0.77 & 0.21 & -0.16 & 0.82 & -1.53 & -1.65 & 2.17\end{bmatrix}^T$$

$$\mathbf{LW}^{5,2} = -0.32$$

$$\mathbf{LW}^{6,4} = 7.76e^{-5}$$

$$b^2 = -0.86$$

$$b^4 = 0.88$$

$\mathbf{IW}^{1,1}, \mathbf{IW}^{3,1} = \mathbf{0}$ and $\mathbf{LW}^{8,7}, \mathbf{IW}^{7,6}, \mathbf{IW}^{8,7}, \mathbf{b}^{7}, b^{8}$ are given in (9.16).

The NARMA-L2 control signal could become unbounded due to the term in the denominator (see (9.9)). This will affect the stability analysis, because RODD methods can only be applied to state equations which are bounded. For this reason, we constrain the denominator. The constraint block shown in Figure (9.14) ignores the denominator for small control signal $[-\varepsilon, \varepsilon]$ ($\varepsilon = 0.1$ has been chosen for this example).

The state space representation derived in (9.17) is suitable for stability analysis using RODD methods. Figure (9.22) shows the response of the state equation of (9.17) for an arbitrary initial condition without reference input. All the states converge to the origin.

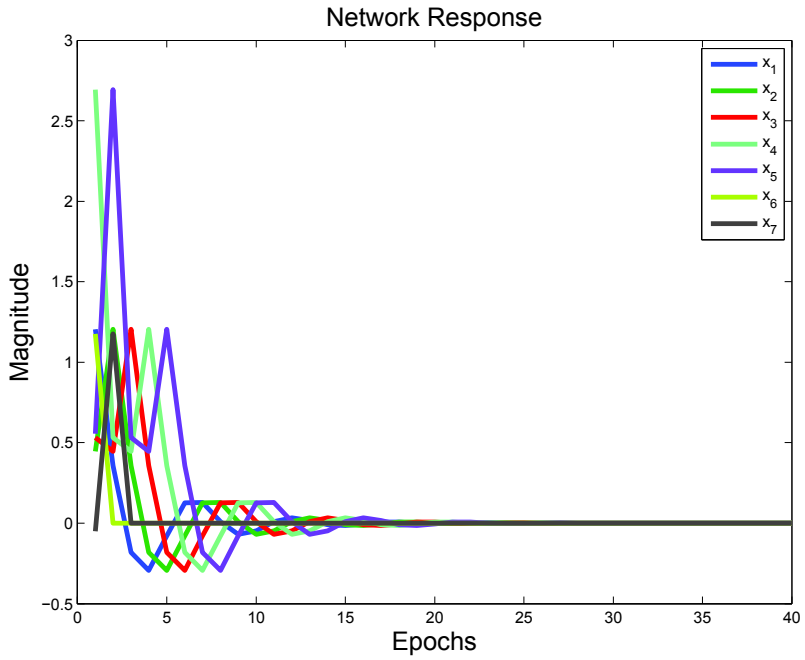

Figure (9.22) Network Response

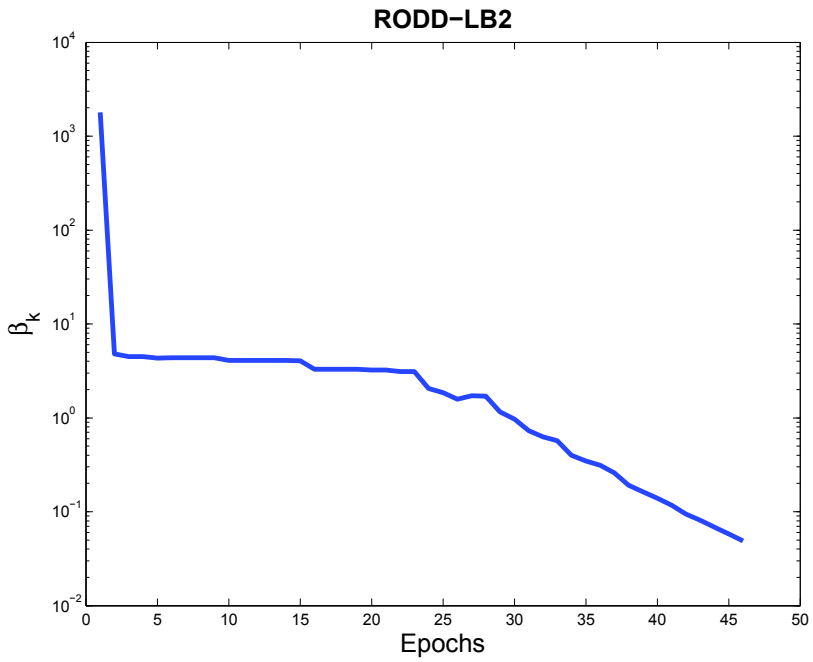

Figure (9.23) Graph of β_k for RODD-LB2

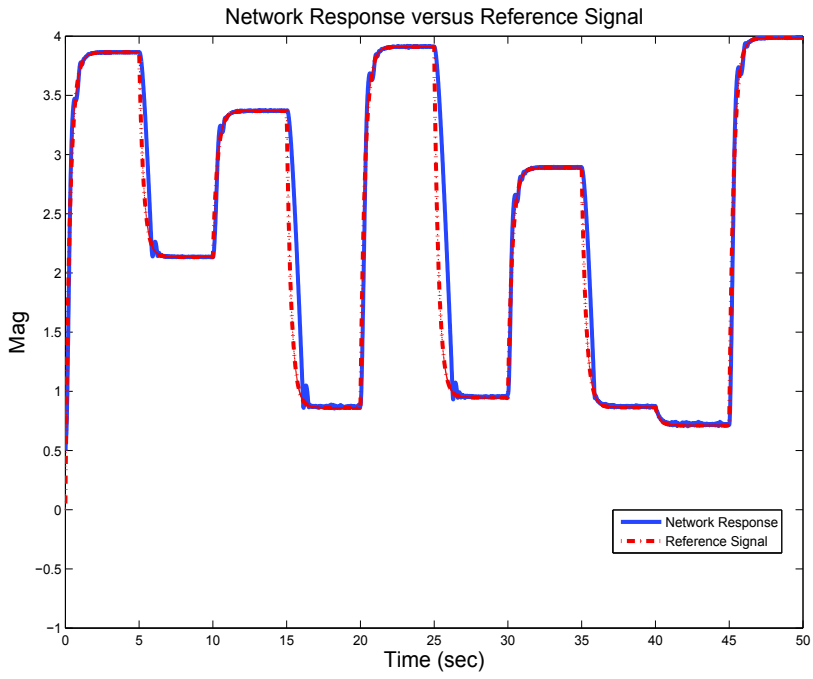

Figure (9.24) Reference Signal versus Network Response

In order to confirm stability of the equilibrium point of (9.17), RODD-LB2 has been applied to this equation. Figure (9.23) shows the graph of β_k for RODD-LB2. Since the equilibrium point of this system is stable, the graph of β_k could become small enough to stop the algorithm. Figure (9.24) shows the performance of the NARAM-L2 in following a reference signal.

The initial set $\mathbf{D}_0$ is placed in the region for which the network has been trained. Table (9.2) illustrates the result of the stability analysis for different methods. Neither RODD-EB nor RODD-Hyrbid were able to confirm stability of the equilibrium point. This might be due to the non-elliptical shape of reachable set. RODD-LB2 is more efficient in detecting stability of the equilibrium point compared to RODD-EB and RODD-Hybrid.

	LB2	EB	Hybrid
Result	Pass	Fail	Fail
Runtime(sec)	271	14400	14400
Improvement	Base	NA	NA

Table (9.2) Comparison of RODD Methods

Conclusion

In this chapter, two RNN-based controllers, MRAC and NARMA-L2, have been investigated. We picked two examples, robot arm and magnetic levitation problems, to check the performance of these controllers. In both cases, the RODD-LB2 method was able to demonstrate stability.

10 CONCLUSIONS

The objective of this research was to develop an efficient algorithm to detect global asymptotic stability of RNNs and then to use the stable RNNs for the purpose of control and system identification. Several methods exist for stability analysis of RNNs, but they are either conservative or suffer from a slow rate of convergence. In this study, we developed three algorithms which address these problems.

In the first section of this chapter, we will discuss the motivation behind this study and then will present a summary of the research. Finally, we will conclude the chapter by describing future work.

Motivation

The main motivation behind this research originated from the previous study by Hagan *et al* [HaDe02], in which they designed a NARMA-L2 controller for magnetic levitation. Magnetic levitation is an example of a highly nonlinear system for which the NARMA-L2 controller obtained excellent performance. NARMA-L2 is an RNN-based

controller that is capable of dealing with severe process nonlinearity. The performance of this controller has been verified through various simulations and generally demonstrates good performance. However, the NARMA-L2 controller occasionally becomes unstable. For this reason, we focused our efforts to derive an efficient algorithm to determine the largest space of stable parameters for a given RNN. In order to achieve this goal, we divided the study into two phases. The first phase of the research was devoted to the modeling of RNNs, and the second phase of the research was devoted to stability analysis.

Summary

In the modeling phase, we started with the most general system representations: State-space and I/O. A discrete-time dynamical system can either be represented by a state-space model or an I/O model. One may choose either model, depending on the application. The state-space model and the I/O model are equivalent under certain conditions. These conditions and the procedure for transforming the state-space model into the I/O model and back were presented in Chapter 2.

After studying the most general dynamical system representations, we concentrated on the stability analysis of RNN models. In our research, we found that the state-space model was convenient for the stability analysis of RNNs. Different examples of state-space models were introduced, e.g., NL_q and the Lur'e model. Then the BPRNN network was introduced as a special type of RNN that was neither in the state-space form nor in the I/O form. Hence, the BPRNN was transformed to the Lur'e model through a special technique called the state-space extension method. The Lur'e model consists of a linear part and a nonlinear part that satisfies certain sector conditions.

The transformation of the BPRNN to the Lur'e model, and the demonstration that the transformed BPRNN model satisfies the sector condition, laid the groundwork for the derivation of the absolute stability criterion for the BPRNN. After several experiments, we found that the stability criteria derived for absolute stability is conservative, and there is a need to derive a better criterion.

A less conservative stability criterion was then investigated in Chapter 6. The RODD-LB1 method can approximate a larger space of stable parameters for a given RNN. Although the RODD-LB1 method gives a less conservative stability criterion, it suffers from a slow rate of convergence. We then developed an extension to RODD-LB1, RODD-LB2. The RODD-LB2 method is an efficient version of RODD-LB1 and can detect stability in less time. Both RODD-LB1 and RODD-LB2 use a linear approximation of the reachable sets. However, the reachable sets can also be approximated by quadratic functions. Next, we developed RODD-EB, which uses a quadratic approximation of the reachable sets. We found in our experiments that for dynamical systems with elliptical reachable sets, the RODD-EB method converges faster than RODD-LB1 and RODD-LB2. The need for an accurate and fast converging method for detecting stability for reachable sets with arbitrary shapes leads us to propose a new algorithm, the RODD-Hybrid method, which is a combination of RODD-LB2 and RODD-EB. All the new RODD algorithms were described in Chapter 7. The manner in which the efficiency of the RODD algorithms depends on the shape of the reachable sets is discussed at the end of Chapter 7.

Chapter 8 and Chapter 9 were devoted to some examples verifying the efficiency of the proposed algorithms. The test problems in Chapter 8 consisted of stable and unstable

systems in a variety of forms. The control problems in Chapter 9 were used to check if the proposed methods can detect the stability of control systems with RNN-based controllers and RNN-based plant models.

In the following section, we will discuss the future work for this research.

Future Work

In this research, different methods for the stability analysis of RNNs were investigated. We developed three efficient algorithms for detecting stability: RODD-LB2, RODD-EB and RODD-Hybrid. The proposed methods opened up many interesting theoretical and practical research possibilities. The methods which we developed in this work can be applied to solve interesting problems. In the following section, some of the these problems are listed.

Maintaining Stability During RNN Training

One of the main challenges with RNNs is training. The potential instability of RNNs complicates their training. The spurious valleys in the training error surface that were studied in [JeHa09] are an immediate consequence of the potential instability. However the new efficient stability algorithms open up the possibility of maintaining stability during RNN training. This will guarantee smoothness of the error surface and improve training performance.

Approximation of Region of Attraction

One of the great feature of the RODD methods is the fact that the initial set $\mathbf{D}_0$ can be selected arbitrarily. Moreover, this set can be placed at any desired location. In this research work, we were looking for GAS equilibrium points. However, RODD methods can

be applied to nonlinear systems with multiple stable equilibrium points. In this case, each stable equilibrium point has a region of attraction. By changing the size and location of $\mathbf{D}_0$ we could use RODD methods to approximate the region of attraction of selected equilibrium points.

REFERENCES

[SuVa96] J. A.K. Suykens, Joos P. L Vandewalle and Bart L. R. De Moor, *Artificial Neural Networks for Modelling and Control of Non-linear Systems*, Kluwer Academic Publication, Boston/Dorcrecht/London, 1996.

[NaTa73] K. S. Narendra, James H. Taylor, *Frequency Domain Criteria for Absolute Stability*, New York and London Press, 1973.

[BoFe94] S. Boyd, L. E. Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory,* SIAM Studies in Applied Mathematics; vol. 15. ISBN: 0-89871-334-X, 1994.

[YaLe04] V. A Yakubovich, G. A. Leonov and A. K. Gelig, *Stability of Stationary Sets in Control Systems with Discontinuous Nonlinearities,* St. Petersburg State Univ., Russia, Series A, 2004.

[Khal01] H. K. Khalil, *Nonlinear Systems,* 3th ed., N.J.: Prentice-Hall, Dec., 2001.

[SuVa97-1] J. A. K. Suykens, J. Vandewalle and B. L. R. De Moor, "NLq theory: A Neural Control Framework with Global Asymptotic Stability Criteria," *Neural Networks*, vol. 10, no. 4, pp. 615-637, Jun., 1997.

[SuVa97-2] J. A. K. Suykens, J. Vandewalle and B. L. R. De Moor, "NLq theory: checking and imposing stability of recurrent neural networks for nonlinear modeling," *IEEE Trans. Signal Process.* vol. 45, no. 11, pp. 2682-2691, Nov. 1997.

[Tana96] K. Tanaka, "An Approach to Stability Criteria of Neural-Network Control Systems," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 629-642, May 1996.

[BaPr02] N. E. Barbanov and D. V. Prokhorov, "Stability Analysis of Discrete-Time Recurrent Neural Networks," *IEEE Trans. Neural Netw.*, vol. 13, no.2, pp. 292-303, Mar. 2002.

[BaPr03] N. E. Barbanov and D. V. Prokhorov, "A New Method for Stability Analysis of Nonlinear Discrete-Time Systems," *IEEE Trans. Autom. Control*, vol. 48, vo.12, pp. 2250-2255, Dec. 2003.

[NaMu97] K. S. Narendra and S. Mukhopadhyay, "Adaptive Control Using Neural Networks and Approximate Models," *IEEE Trans. Neural Netw*, vol. 8, pp. 475-485, May 1997.

[Nade01] N. Sadegh, "Minimal Realization of Non-Linear Systems Described by Input-Output Difference Equations," *IEEE Trans. Autom. Control*, vol. 46, no. 5, pp. 698-710, May 2001.

[Jaku77] V. A. Jakubovic, “The S-procedure in Nonlinear Control Theory,” *Vestnik*, Univ. Vestnik Leningrad, no. 1, vol. 4, pp. 73-93, 1977.

[CaNa95] J. B. D. Cabrera and K. S. Narendra, “On the Control of Nonlinear Discrete-Time Systems, Part I: State Variables Accessible”, Elect. Eng. Yale Univ., CT, Tech. Rep. TR. 9515, Dec. 1995.

[Buck78] R. Creighton Buck, *Advanced Calculus*, 3th ed., New York: McGraw-Hill 1978.

[Hans95] B. G. Hanson, *General Systems Theory-Beginning with Whole*s, Taylor & Francis,York Univ., Toronto, Ontario, Canada, 1995.

[MeTa75] M.D. Mesarovic and Y. Takahara, *General Systems Theory: Mathematical Foundations*, Academic Press, Math. in Sci. and Eng., vol 113, Feb. 1975.

[WaZo06] L. Wang and Z. Xu, “Sufficient and Necessary Conditions for Global Exponential Stability of Discrete-Time Recurrent Neural Networks,” *IEEE Trans. on Circuits and Syst*., vol. 53, no. 6, pp. 1373-1380, 2006.

[Tana96] K. Tanaka, “An Approach to Stability Criteria of Neural-Network Control Systems,” *IEEE Trans. Neural Netw.*, vol. 7, No.3 pp. 629-642, 1996.

[Liu07] M. Liu, “Delayed Standard Neural Networks Models for Control Systems” *IEEE Trans. Neural Netw.*, vol. 18, no. 5, pp.1376-1391, Sep. 2007.

[FaKi96] Y. Fang and T. G. Kincaid, “Stability Analysis of Dynamical Neural Networks,” *IEEE Trans. Neural Netw*, vol. 7, no.4 pp. 996-1006, 1996.

[Cher93] F. L. Chernousko, *State Estimation for Dynamic Systems*, CRC press, Institute for Problems in Mechanics, Russian Academy of Sciences 1993.

[Bana22] S. Banach, S*ur les opérations dans les ensembles abstraits et leur application aux équations intégrales*, PhD. dissertation, Math Dep. Univ. Lvov, 1922.

[Mosh05] N. Moshtagh, “Minimum *Volume Enclosing Ellipsoids,”* tech. report, School of Eng. and Applied Science, Univ. Pennsylvania, 2005.

[Elya98] S. Elyadi, “Stability Theory,” *An Introduction to Difference Equations*, Springer, 3th Ed., Dec. 1998.

[Ross80] K. A. Ross, *Elementary Analysis: The theory of Calculus*, Springer, Mar., 1980.

[GoYa02] C. J. Goh and X. Q. Yang, *Duality in Optimization and Variational Inequalities*, Taylor & Francis, London, New York, May 2002.

[Meye00] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra,* SIAM, 2000.

[Jafar11] R. Jafari, *Direct and Converse Lyapunov Theorems for Discrete Dynamical Systems*, Creative Component, Math. Dep., Oklahoma State Univ., Apr., 2011.

[JeHa09] J. Horn, O. D. Jesus and M. Hagan, "Spurious Valleys in the Error Surface of Recurrent Networks - Analysis and Avoidance," *IEEE Trans. Neural Netws,* vol. 20, no. 4, pp. 686-700, Apr. 2009.

[JaDh07] R. Dhaouadi, R. Jafari "Adaptive PID Neuro-Controller for a Nonlinear Servomechanism*", IEEE Int. Symp. Ind. Electron., pp. 157-162, Jun.,* 2007.

[HaDe02] M. Hagan, H. B. Demuth and O. D. Jesus, "An Introduction to the Use of Neural Networks in Control Systems*", International Journal of Robust and Nonlinear Control,* vol. 12, no. 11, pp. 959-985, Sep. 2002.

[HaDe99] M. Hagan, H. B. Demuth, "Neural Networks for Control*", American Control Conference*, pp. 1642-1656, 1999.

[HuSb92] K. J. Hunt, D. Sbarbaro, R.Zbikowski and P. J. Gawthrop, "Neural Networks for Control System*", Automatica,* vol. 28, pp. 1083-1112, 1992.

[NaPa90] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks*", IEEE Trans. on Neural Networks,* vol. 1, pp. 4-27, 1990.

[BeHa12] M. H. Beale, M. T. Hagan and H. B. Demuth, Neural Network Toolbox, User's Guide, R2012a

[con2ver] http://www.mathworks.com/matlabcentral/fileexchange/7894-con2vert-constraints-to-vertices

APPENDIX A

NARMA-L1

NARMA-L1 is an approximation of the NARX model using the Taylor series expansion. Let's rewrite (2.21) as

$$y(k+1) = F(\mathbf{Y}_n(k), \mathbf{U}_n(k)) \tag{A.1}$$

where $\mathbf{Y}_n(k) = [y(k), y(k-1), \ldots, y(k-n+1)]$,

$\mathbf{U}_n(k) = [u(k), u(k-1), \ldots, u(k-n+1)]$ and $F = hog : \mathbf{R}^{2n} \rightarrow \mathbf{R}$.Using the Taylor series expansion we can expand (A.1) about the operating point

$(y(k), y(k-1), \ldots, y(k-n+1), 0, \ldots, 0) = (\mathbf{Y}_n(k), \mathbf{0})$ as follows:

$$\begin{aligned} y(k+1) = {} & F(\mathbf{Y}_n(k), \mathbf{0}) + \frac{\partial}{\partial u(k)} F(\mathbf{Y}_n(k), \mathbf{U}_n(k))\Big|_{\mathbf{U}_n(k)=0} \cdot u(k) + \\ & \frac{\partial}{\partial u(k-1)} F(\mathbf{Y}_n(k), \mathbf{U}_n(k))\Big|_{\mathbf{U}_n(k)=0} \cdot u(k-1) + \ldots + \\ & \frac{\partial}{\partial u(k-n+1)} F(\mathbf{Y}_n(k), \mathbf{U}_n(k))\Big|_{\mathbf{U}_n(k)=0} \cdot u(k-n+1) + \mathbf{R}_1(\mathbf{Y}_n(k), \mathbf{U}_n(k)) \end{aligned} \tag{A.2}$$

where $\mathbf{R}_1(\mathbf{Y}_n(k), \mathbf{U}_n(k))$ is the remainder with the following upper bound

$$\mathbf{R}_1(\mathbf{Y}_n(k), \mathbf{U}_n(k)) \leq \frac{M_1 \|\mathbf{U}_n(k)\|^2}{2} \tag{A.3}$$

$\mathbf{M}_1$ is the maximum matrix norm of the Hessian matrix $\frac{\partial}{\partial U}\left(\frac{\partial f}{\partial U}\right)^T$ when evaluated over

the compact domain $K_Y \times K_U \subset \mathbf{R}^n \times \mathbf{R}^n$. By the right choice of $\mathbf{U}_n(k)$ we can make the upper bound in (A.3) very small. For example, if the objective is to have the remainder to be less than ε then $|u(k)|$ must be chosen less than δ such that $\delta \leq \sqrt{\frac{2\varepsilon}{\mathbf{M}_1 n}}$. Considering a small upper bound for the remainder and denoting

$g_i(\mathbf{Y}_n(k)) \equiv \frac{\partial}{\partial u(k-i)} F(\mathbf{Y}_n(k), \mathbf{U}_n(k))\big|_{\mathbf{U}_n(k)=0}$, $0 \leq i \leq n-1$, (A.2) may be approximated by

$$y(k+1) \cong F(\mathbf{Y}_n(k), \mathbf{0}) + \left[g_0(\mathbf{Y}_n(k)) \; g_1(\mathbf{Y}_n(k)) \; \ldots \; g_{n-1}(\mathbf{Y}_n(k)) \right] \begin{bmatrix} u(k) \\ u(k-1) \\ \ldots \\ u(k-n+1) \end{bmatrix} \quad \text{(A.4)}$$

$$y(k+1) \cong \bar{F}(y(k), y(k-1), \ldots, y(k-n+1)) + \sum_{i=0}^{n-1} f_i(y(k), y(k-1), \ldots, y(k-n+1))u(k-i)$$

where $\bar{F}, f_i : \mathbf{R}^n \to \mathbf{R}$

Equation (A.4) is the approximation of the NARX model given in (2.21), which is called the NARMA-L1 model. This model has the advantage that the output depends linearly on the control signals. So, control signals can easily be calculated at each time step from the reference point. Although the approximation error can always be made small by the right design of a controller, the NARMA-L1 model derived in (A.4) is only a good approximation if the operating point of the NARX model does not change from the origin. In this case the approximation will not be good, and a different method for calculating the control signals will need to be used.

Lemma A.1 *Let* $0 < c < r$ *and* r *be as defined in (4.21),then*

$$\frac{\tanh r + \tanh c}{r + c} \geq \frac{\tanh(r) - \tanh c}{r - c} \tag{A.5}$$

Proof: As proven in chapter 4 $\frac{\tanh(s)}{s}$ is a monotonically increasing in the right half plane. Hence, for $c \in [0, r]$

$$\frac{\tanh c}{c} \geq \frac{\tanh r}{r} \tag{A.6}$$

Multiplying both sides of the above inequality by $2rc$ yields

$$\begin{gathered} 2r\tanh c \geq 2c\tanh r \\ r\tanh c + r\tanh c \geq c\tanh r + c\tanh r \\ r\tanh c - c\tanh r \geq c\tanh r - r\tanh c \end{gathered} \tag{A.7}$$

Adding $r\tanh r - c\tanh c$ to both sides of (A.7) yields,

$$\begin{gathered} r\tanh c - c\tanh r + r\tanh r - c\tanh c \geq c\tanh r - r\tanh c + r\tanh r - c\tanh c \\ (r - c)\tanh c + (r - c)\tanh r \geq (r + c)\tanh r - (r + c)\tanh c \\ (r - c)[\tanh r + \tanh c] \geq (r + c)[\tanh r - \tanh c] \end{gathered} \tag{A.8}$$

Dividing both sides of (A.8) by $(r - c)(r + c)$ proves the lemma. □

Theorem A.1 *Pick* $c > 0$ *and define a new function* $g_c : \mathbf{R} \to \mathbf{R}$ *as*

$$g_c(s) = \begin{cases} \dfrac{f(s) - f(c)}{s - c} & s - c \neq 0 \\ f'(c) & s - c = 0 \end{cases} \tag{A.9}$$

where $f(s) = \tanh(s)$, *then* g_c *satisfies the following conditions*

1. g_c *is of class* C^1.

2. $\lim_{s \to -\infty} g_c(s) = \lim_{s \to \infty} g_c(s) = 0$

3.There exists unique $c' \in [-c, 0]$ *(*$c > 0$*) such that* $g_c(c') = 1 - \tanh^2 c'$

$$4.\begin{cases} g_c'(s) > 0 & ; \quad s < c' \\ g_c'(s) < 0 & ; \quad s > c' \\ g_c'(s) = 0 & ; \quad s = c' \end{cases}$$

Proof: The proof includes the following sections:

1. To show that g_c is of class C^1, it is required to prove that $g_c'(s)$ exists and it is continuous. By the definition of the derivative,

$$\begin{aligned} g_c'(s) &= \lim_{s \to c} \frac{g_c(s) - g_c(c)}{s - c} \\ &= \lim_{s \to c} \frac{\frac{f(s) - f(c)}{s - c} - f'(c)}{s - c} \\ &= \lim_{s \to c} \frac{f(s) - f(c) - (s - c) f'(c)}{(s - c)^2} \end{aligned}$$

Applying L'Hôpital's rules rule yields

$$g_c'(s) = \lim_{s \to c} \frac{f'(s) - f'(c)}{2(s - c)}$$

Since the above equation is not defined when $s \to c$, so another L'Hôpital's rule yields

$$g_c'(s) = \frac{f''(s)}{2} \tag{A.10}$$

The above equation guarantees the existence of $g_c'(s)$. The next step is to show that $g_c'(s)$ is continuous. To show the continuity of $g_c'(s)$, it is enough to show the following equation is true

$$
\begin{aligned}
g_c'(c) &= \lim_{s \to c} g_c'(s) \\
&= \lim_{s \to c} \frac{(s-c)f'(c) - [f(s) - f(c)]}{(s-c)^2}
\end{aligned}
\tag{A.11}
$$

By the definition, the derivative of g_c at c is defined as

$$
\begin{aligned}
g_c'(c) &= \lim_{s \to c} \frac{g_c(s) - g_c(c)}{s-c} \\
&= \lim_{s \to c} \frac{g_c(s) - g_c(c)}{s-c} \\
&= \lim_{s \to c} \frac{f(s) - f(c) - (s-c)f'(c)}{(s-c)^2}
\end{aligned}
\tag{A.12}
$$

By applying the L'Hôpital's rule to above equation twice, $g_c'(c)$ will be derived as follows

$$
g_c'(c) = \frac{f''(c)}{2} \tag{A.13}
$$

By applying L'Hôpital's rule to the right hand part of (A.11) the following equation will be derived

$$
\begin{aligned}
\lim_{s \to c} \frac{(s-c)f'(c) - [f(s) - f(c)]}{(s-c)^2} &= \lim_{s \to c} \frac{(s-c)f''(s) + f'(s) - f'(s)}{2(s-c)} \\
&= \frac{f''(c)}{2}
\end{aligned}
\tag{A.14}
$$

Equation (A.13) and (A.14) prove (A.11). This is enough to say that $g_c'(s)$ is continuous at $s = c$. Since c can be any number in the domain, $g_c'(s)$ is a continuous function in the entire domain.

2. The proof to this part can be achieved by inspection. The function $g_c(s)$ goes to zero as $s \to -\infty$ because the numerator is bounded and the denominator goes to $-\infty$. Therefore, the quotient goes to zero. The case when $s \to \infty$ can be proven similarly.

3. The function $f(s) = \tanh(s)$ is concave up for $s < 0$. This is enough to say that the slope of the tangent line at $s = -c$ is less than the slope of the cord connecting two points $(c, f(c))$ and $(-c, -f(c))$. In other words,

$$f'(-c) < g_c(-c) \tag{A.15}$$

Also the slope of the tangent line at the origin is greater than the slope of all the cords pivoted at $(c, f(c))$. Furthermore,

$$f'(0) > g_c(0) \tag{A.16}$$

Since both f and g_c are continuous functions, and $f' - g_c$ changes sign in the interval $[-c, 0]$, by the Intermediate Value Theorem (IVT), there is a point $c' \in [-c, 0]$ such that

$$g_c(c') = f'(c') \tag{A.17}$$

Equation (A.17) proves this part of the theorem.

4. This part of the proof is more important than the previous parts, because the results will be used directly to prove Lemma A.2. The derivative of $g_c'(s)$ at $s - c \neq 0$ is defined as

$$\begin{aligned} g_c'(s) &= \frac{(s-c)\left[f'(s) - \dfrac{f(s)-f(c)}{s-c}\right]}{(s-c)^2} \\ &= \frac{f'(s) - g_c(s)}{s-c} \end{aligned} \tag{A.18}$$

Figure (A.1) shows the graph of $g_c(s)$ for a positive c. It can be observed from the graph that the entire domain can be divided into four regions.

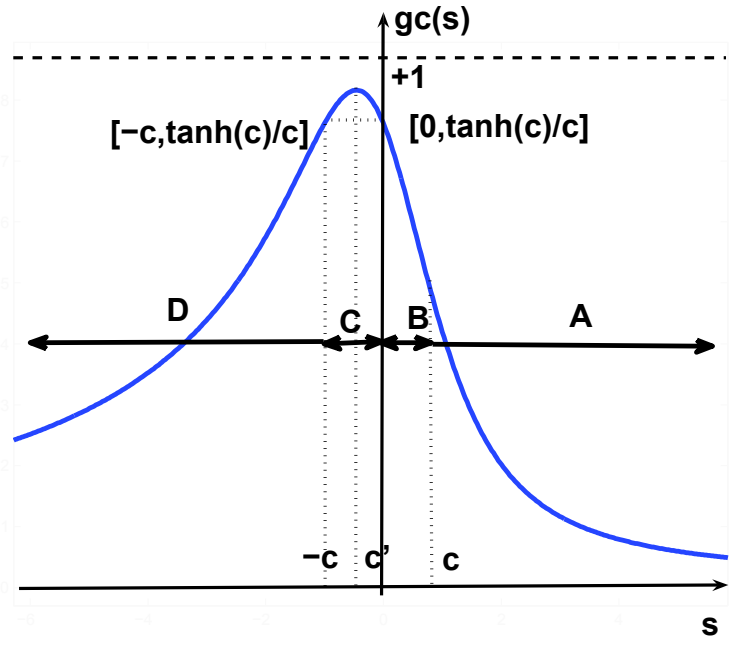

Figure (A.1) $g_c(s)$ divided into four regions

Region A: In this region, $s > c$ and the function $f(s) = \tanh(s)$ is concave down. As can be observed from Figure (A.2), the slope of the tangent line at any point in this region is always less than the slope of the cords pivoted at $(c, f(c))$. In other words, $f'(s) < g_c(s)$. So the numerator of (A.18) is negative and the denominator is positive. Moreover

$$g_c'(s) < 0 \qquad \text{(A.19)}$$

Region B: In this region $0 < s < c$ and the function $f(s) = \tanh(s)$ is concave down. The difference between this case and previous case is that s lies on the left side of

c. As can be observed from Figure (A.2), the slope of the tangent line to the function at any point in this region is always greater than any cords pivoted at $(c, f(c))$. In other words, $f'(s) > g_c(s)$. So, the numerator of (A.18) is positive and the denominator is negative. Therefore

$$g_c'(s) < 0 \tag{A.20}$$

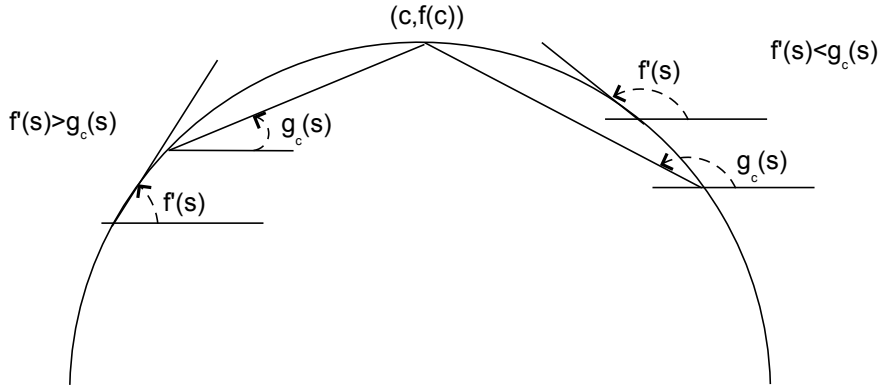

Figure (A.2) $\tanh(s)$ is concave down for $s > 0$

Region C: In this region, $-c < s < 0$. Since the argument to define the sign of g_c' in the region $s \in [-c, c']$ and $s \in [c', 0]$ is similar, the proof is only given for the case when $s \in [-c, c']$. Denote point A on the graph of f at $(c, f(c))$ by (a, b). Consider a point, B, in the assumed interval on the graph of f and denote it as (c, d). The line BC is the tangent line at a point in the interval. This line crosses the cord with one end at $(c, f(c))$ and the other end at $(c', f(c'))$. Denote the cross point as C. In order to simplify the proof, let us shift the coordinate to the point C. The slope of the line AB is equal to

$$\begin{aligned} g_c(s) &= \frac{f(c)-f(s)}{c-s} \\ &= \frac{b-d}{a-c} \end{aligned} \tag{A.21}$$

The slope of the tangent line, BC, at s in the interval is equal to $\frac{d}{c}$. The claim here is as follows

$$\frac{d}{c} < \frac{b-d}{a-c} \tag{A.22}$$

The line AC has the following equation

$$y(s) = \frac{b}{a}s$$

Because of the location of the point B, $d - \frac{b}{a}c > 0$. Due to the location of the points A, B and C and the reason that f is monotonically increasing, the following inequalities can be derived

$$\begin{aligned} &c < a \\ &d < b \\ &c < 0 \\ &d < 0 \\ &a > 0 \end{aligned} \tag{A.23}$$

Beginning with $d - \frac{b}{a}c > 0$ and considering (A.23) the following inequalities are always valid

$$
\begin{aligned}
d &> \frac{b}{a}c \\
ad &> bc \\
ad - cd &> bc - cd \\
d(a-c) &> c(b-d) \\
\frac{d}{c} &< \frac{b-d}{a-c}
\end{aligned}
\tag{A.24}
$$

The final inequality in (A.24) proves the claim in (A.22).

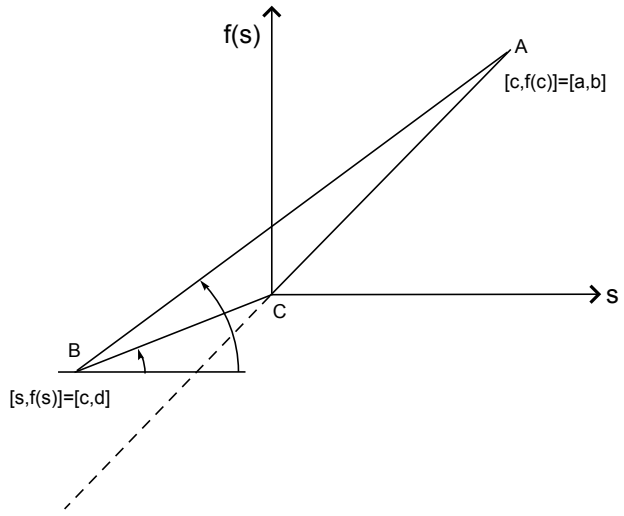

Figure (A.3) Geometric representation of $g_c(s)$ and $f'(s)$ for $s \in [-c, c']$

Region D: In this region, $s < -c$, the function f is concave up. Similar to the graphical representation given in Figure (A.2), it can be observed that the tangent line to the function at any point in this region is below the cord with one end pivoted at $(c, f(c))$ and the other end at $(s, f(s))$. In other words, $f'(s) < g_c(s)$. Moreover, the numerator of (A.18) is negative and the denominator is negative as well. Furthermore

$$g_c'(s) > 0 \tag{A.25}$$

Merging the results from the four regions proves the final part of the theorem.

Lemma A.2 *Let* $0 < c < r$ *and* r *to be defined in (4.21). Defining the new functions*

$$\begin{aligned} f_1(s) &\equiv \frac{\tanh(s+c) - \tanh(c)}{s} \\ f_2(s) &\equiv 1 - \tanh^2(s+c) \\ f_3(s) &= f_2(s) - f_1(s) \end{aligned} \tag{A.26}$$

If $|s + c| < r$ *for* $\forall s \in R$ *then*

$$\begin{cases} f_3(s) < 0 \;\; ; & -r - c < s < s_1 \\ f_3(s) > 0 \;\; ; & s_1 < s < 0 \\ f_3(s) < 0 \;\; ; & 0 < s < r + c \end{cases} \tag{A.27}$$

where s_1 *is the unique maximum point of* f_1.

Proof: According to (A.18), there is a relationship between f_1 and g_c. For the case when $s - c \neq 0$ the relationship can be written as follows

$$\begin{aligned} g_c(s+c) &= \frac{\tanh(s+c) - \tanh(c)}{s} \\ &= f_1(s) \end{aligned} \tag{A.28}$$

In Theorem A.1 part 4 it has been proved that c' is the unique maximum point of $g_c(s)$. Equivalently, $c' - c$ would be the unique maximum point of $g_c(s + c)$ for $\forall c > 0$. By the lemma assumption, s_1 is the unique maximum point of $f_1(s)$ .Additionally since $g_c(s + c) = f_1(s)$ for $s - c \neq 0$, the two maximum points should be identical. In other words

$$c' - c = s_1 \tag{A.29}$$

There is another relationship between $g_c'(s + c)$ and $f_3(s)$. The relationship can be derived

by evaluating (A.18) at $s + c$. So

$$g_c'(s+c) = \frac{1 - \tanh^2(s+c) - \dfrac{\tanh(s) - \tanh(c)}{s-c}}{s} = \frac{f_3(s)}{s} \tag{A.30}$$

The sign of $f_3(s)$ depends on the sign of $g_c'(s+c)$ and s. Consider the following three cases:

1. For $s < c' - c$, the derivative of function $g_c(s+c)$ is greater than zero because $c' - c$ is the unique maximum point for $g_c'(s+c)$. The function $g_c(s+c)$ is continuous so its derivative on the left side of the maximum point must be positive. Furthermore, by (A.30)

$$f_3(s) < 0 \text{ for } s < c' - c \tag{A.31}$$

2. For $c' - c < s < 0$ the derivative of function $g_c(s+c)$ is less than zero. The reason for this is similar to case 1. The function $g_c(s+c)$ is a continuos function and the derivative on the right hand side of the maximum point is negative. Hence,

$$f_3(s) > 0 \quad \text{for } c' - c < s < 0 \tag{A.32}$$

3. For $s > 0$ the derivative of function $g_c(s+c)$ is still less than zero for the same reason given in case 2. In other words

$$f_3(s) < 0 \quad \text{for } s > 0 \tag{A.33}$$

The results derived in (A.31), (A.32), (A.33) plus the equality in (A.29) proves the lemma.

□

Corollary A.1 *When* $c = 0$, $f_3(s) \le 0$ for $\forall s \in R$.

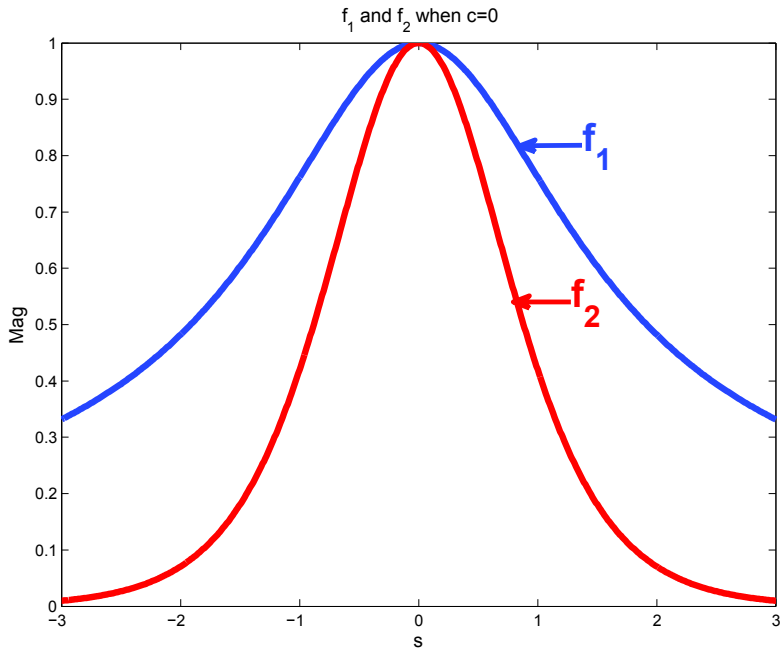

Figure (A.4) When $c = 0$, $f_3(s) \le 0$ for $\forall s \in \mathbf{R}$.

Proof: The proof is exactly the same as the proof given for the Lemma A.2 with the difference that when $c = 0$ then $s_1 = 0$. According to (A.27), $f_3(s) \le 0$ for $\forall s \in R$.

□

VITA

Reza Jafari

Candidate for the Degree of

Doctor of Philosophy

Thesis: STABILITY ANALYSIS OF RECURRENT NEURAL –BASED CONTROLLERS

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy/Education in electrical engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2012.

Completed the requirements for the Master of Science in Applied Mathematics at Oklahoma State University, Stillwater , Oklahoma in 2012.

Completed the requirements for the Master of Science in Mechatronics at American University of Sharjah, Sharjah , United Arab Emirates in 2005.

Experience:
Graduate Research Assistant, Department of Electrical Engineering, Oklahoma State University, Teaching Assistant, Department of Mechanical and Aerospace & Department of Electrical Engineering, Oklahoma State University, Project Management, Modal & Vibration Analysis & Control for embedded/real-time systems, Real System Modeling utilizing Simulink; Digital and Analog Sensors, Theory and Design; MATLAB / Simulink; SFunction; dSpace, Labview; Software Development; Feed Forward and Feedback Control for Nonlinear Systems; Computer Programming; Software Development; Finite Element Method & Analysis (ANSYS); Linear & Non-Linear Convex Optimization; Adaptive Control, Designing smart sensor suing Neural Networks and FPGA for Mobile Robots. Offer industrial expertise and strengths in electrical and computer engineering complemented with a passion for research and development with a concentration in analysis of dynamical systems and design of modern controllers based on recurrent neural networks. Additional research in fuzzy logic and neural networks design for pattern recognition.

Professional Memberships:
Institute of Electrical and Electronics Engineering in Control Systems
Institute of Electrical and Electronics Engineering in Computer Society
Institute of Electrical and Electronics Engineering in Neural Networks
Institute of Electrical and Electronics Engineering in Robotics and Automation Society

Name: Reza Jafari Date of Degree: May, 2012

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study:

STABILITY ANALYSIS OF RECURRENT NEURAL-BASED CONTROLLERS

Pages in Study: 234 Candidate for the Degree of Doctor of Philosophy

Major Field: Electrical Engineering

Scope and Method of Study:

In the last two decades, more attention has been given to Recurrent Neural Networks (RNNs) and their applications. The ability of RNNs to solve complex problems in control, system identification, signal processing, communication, pattern recognition, etc. is well understood. Although RNNs are more powerful than feedforward networks, this comes at the expense of more difficult training and the potential for instabilities. It has become more and more important to have efficient methods for determining the stability of RNNs. The purpose of this research is to develop improved methods for determining the stability of RNNs.

Findings and Conclusions:

The main contribution of this research is the development of an efficient algorithm to detect global asymptotic stability of RNNs and then use the stable RNNs for the purpose of control and system identification. Three Lyapunov-based algorithms, RODD-LB2, RODD-EB and RODD-Hybrid, are developed to detect global asymptotic stability of RNNs. We found that the RODD-Hybrid method generally produced the fastest and the most consistent results. The applications of the proposed algorithms have been applied to neural network control problems.

ADVISER'S APPROVAL: Dr. Martin Hagan