Louisiana Tech University
Louisiana Tech Digital Commons

Doctoral Dissertations Graduate School

Spring 2005

Intelligent control of nonlinear systems with actuator saturation using neural networks

Wenzhi Gao

Follow this and additional works at: https://digitalcommons.latech.edu/dissertations

Part of the Electrical and Computer Engineering Commons

NOTE TO USERS

This reproduction is the best copy available.

UMI®

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

INTELLIGENT CONTROL OF NONLINEAR SYSTEMS WITH ACTUATOR SATURATION USING NEURAL NETWORKS

by

Wenzhi Gao, M.S.

**A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy**

**COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY**

March 2005

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

UMI Number: 3164471

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 3164471

Copyright 2005 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

Feb. 24 2005
Date

We hereby recommend that the dissertation prepared under our supervision by Wenzhi Gao

entitled Intelligent Control of Nonlinear Systems with Actuator Saturation Using Neural Networks

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

[illegible]
Supervisor of Dissertation Research

Richard J. Greechie
Head of Department

CAM
Department

Recommendation concurred in:

Lihua Zou

Richard J. Greechie

Vir Phoha [illegible]

Advisory Committee

Approved:
[illegible]
Director of Graduate Studies

[illegible]
Dean of the College

Approved:
[illegible]
Dean of the Graduate School

GS Form 13
(5/03)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ABSTRACT

Common actuator nonlinearities such as saturation, deadzone, backlash, and hysteresis are unavoidable in practical industrial control systems, such as computer numerical control (CNC) machines, xy-positioning tables, robot manipulators, overhead crane mechanisms, and more. When the actuator nonlinearities exist in control systems, they may exhibit relatively large steady-state tracking error or even oscillations, cause the closed-loop system instability, and degrade the overall system performance. Proportional-derivative (PD) controller has observed limit cycles if the actuator nonlinearity is not compensated well. The problems are particularly exacerbated when the required accuracy is high, as in micropositioning devices. Due to the non-analytic nature of the actuator nonlinear dynamics and the fact that the exact actuator nonlinear functions, namely operation uncertainty, are unknown, the saturation compensation research is a challenging and important topic with both theoretical and practical significance.

Adaptive control can accommodate the system modeling, parametric, and environmental structural uncertainties. With the universal approximating property and learning capability of neural network (NN), it is appealing to develop adaptive NN-based saturation compensation scheme without explicit knowledge of actuator saturation nonlinearity. In this dissertation, intelligent anti-windup saturation compensation schemes in several scenarios of nonlinear systems are investigated. The nonlinear systems

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

studied within this dissertation include the general nonlinear system in Brunovsky canonical form, a second order multi-input multi-output (MIMO) nonlinear system such as a robot manipulator, and an underactuated system-flexible robot system. The abovementioned methods assume the full states information is measurable and completely known.

During the NN-based control law development, the imposed actuator saturation is assumed to be unknown and treated as the system input disturbance. The schemes that lead to stability, command following and disturbance rejection is rigorously proved, and verified using the nonlinear system models. On-line NN weights tuning law, the overall closed-loop performance, and the boundedness of the NN weights are rigorously derived and guaranteed based on Lyapunov approach. The NN saturation compensator is inserted into a feedforward path. The simulation conducted indicates that the proposed schemes can effectively compensate for the saturation nonlinearity in the presence of system uncertainty.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author [illegible]

Date Feb. 25 2005

GS Form 14
(5/03)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To Guoquan Zhang, Fendy Gao and my dear parents

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

TABLE OF CONTENTS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

LIST OF TABLES

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

LIST OF FIGURES

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

ACKNOWLEDGMENTS

I would like to express great appreciation to Dr. Rastko Selmic for his knowledge transfer, advice, support, inspiring work and encouragement. I also appreciate the opportunity of having been allowed to work as a teaching and research assistant under his supervision. I attribute most of my research career advancement to his insightful guidance and suggestions.

I would like to thank Dr. Lihe Zou, Dr. Vir V. Phoha, Dr. Huaijin Gu and Dr. Richard J. Greechie, for serving on my advisory committee and reviewing my dissertation. I also want to thank Dr. Zou and Dr. Greechie for their countless help and kind suggestions on my career development. I appreciate that Dr. Gu's guidance on the research of adaptive signal processing. In addition, I wish to thank all of the friends in Ruston who helped me and made my life here memorable.

Finally, I thank my wife Guoquan Zhang, our daughter Fendy, and my dear parents in China, for their love, patience, encouragement and unselfishness accompanying me always during my academic pursuits.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 1

INTRODUCTION

1.1 Preface

Saturation, deadzone, backlash, and hysteresis, are the most common actuator nonlinearities in practical control systems. Saturation nonlinearity is unavoidable in most actuators. Categories of saturation nonlinearities include constraints of the magnitude and the rate of actuator inputs. These limits maybe due to restrictions deliberately placed on the actuators to avoid damage to a system and/or physical limitations on the actuators themselves. When an actuator has reached such an input limit, it is said to be "saturated", since efforts to further increase the actuator output would not result in any variation in the output. Due to the non-analytic nature of the actuator nonlinear dynamics and the fact that the exact actuator nonlinear functions, namely operation uncertainty, are unknown, such systems present a challenge to the control design engineer (Narendra (1991)), and provide an application field for adaptive control, sliding control and neural network-based control. If the saturation exists, proportional-derivative (PD) controller has observed limit cycles; it could lead to a phenomenon called integrator windup (Lewis, Abdallah, and Dawson (1993)), (Hu and Lin (2001)). Integrator windup occurs when a

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

system has actuator saturation and an integrator in its controller. The effect causes the control signal saturates the actuator, and a further increase of the control signal will_not lead to a faster response of the system. If integration of error continues in this case, the integrator value becomes very large, without having any effect on the plant. The control error has to be of the opposite sign for a long time to bring the integrator back to its steady-state value. Some form of anti-windup mechanism must be implemented in the PID controller.

To tackle this problem, Astrom and Wittenmark (1996) developed the general actuator saturation compensator scheme; Hanus and Peng (1992) addressed a controller based on the conditional technique; Walgama and Sternby (1990) developed an observer-based anti-windup compensator; Niu (1998) designed a robust anti-windup controller based on the Lyapunov approach to accommodate the constraints and disturbance; Chan and Hui (1998) investigated the actuator saturation stability issues related to the number of the integrators in the plant; Hu and Lin (2001) proposed a systematic controller design to compensate the saturation nonlinearity for continuous and discrete linear systems. Annaswamy *et al.* (2001) addressed an adaptive controller to accommodate saturation constraints in the presence of time delays, which is applicable to 1st, 2nd, and n-th order linear plants. Atherton (1995) presented tracking windup and modified tracking windup scheme for linear system based on the measured actuator output.

In some seminal recent work, several rigorously derived adaptive schemes have been given for actuator nonlinearity compensation (Tao and Kokotovic (1996)). Compensation for non-symmetric deadzone is considered in (Selmic and Lewis (2000)), (Tao and Kokotovic (1994)), and (Tao and Kokotovic (1995)) for linear systems and in

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(Recker and. Kokotovic (1991)) for nonlinear systems in Brunovsky form with known nonlinear functions. Backlash compensation is addressed in (Desoer and Sharhruz (1986)), (Tao and Kokotovic (1995)), and hysteresis in (Selmic and Lewis (2000)) and (Tao and Kokotovic (1996)).

Much has been written on intelligent control using neural networks (NNs) (Miller, Sutton and Werbos (1990)), (Barron (1993)), (Commuri and Lewis (1995)), (Narendra and Parthasarathy (1990)). With the universal approximation property and learning capability (Lewis, Yesildirek, and Liu (1996)), NNs have proved to be a powerful tool to control complex dynamic nonlinear systems with parameter uncertainty. Although persistent problems (Lewis, Yesildirek, and Liu (1996)), such as *ad hoc* controllers, lack of rigorous stability proof, approximation of non-smooth functions and off-line weights initialization requirement still exist, NN has been widely used in adaptive and robust adaptive control. The common control strategies with regards to NN are direct adaptive NN control method with guaranteed stability (Choi, Lee and Kim (2001)), indirect adaptive NN control based on identification (Narendra (1991)), and dynamic inverse NN control (Lewis, Campos and Selmic (2002). In general, NN is used to estimate the unknown nonlinear dynamics and/or function and to compensate for them. Unlike the standard adaptive control schemes, NN can also cope with a nonlinear system that is linearly unparameterizable. Recently, a large amount of research (Lewis, Yesildirek, and Liu (1996)), (Lewis *et al.* (1993)), (Lewis *et al.* (1999)), (Lin *et al.* (2001)), (Polycarpou (1996)), (Recker *et al.* (1991)), (Selmic and Lewis (2000)), (Chen and Khalil (1992)) has used NN to synthesize the feedback linearization for the feedback

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

linearizable system (Isidori (1989)) and to incorporate the Lyapunov theory in order to ensure the overall system stabilization, command following and disturbance rejection.

Most saturation approaches mentioned above focus on the linear plant and assume that the saturation is symmetric and that the actuator output is measurable. This dissertation proposes several novel NN-based saturation control schemes for general nonlinear systems. The approaches are applied to the feedback linearizable (Nam (1999)) nonlinear plants, with a general model of actuator saturation assuming that the actuator output is not necessarily measurable. NN weights are tuned on-line, and the overall system performance is guaranteed based on Lyapunov function approach. The convergence of the NN learning process and the boundedness of the NN weights estimation error are all rigorously proven. The simulation results regarding to the related nonlinear dynamics model are provided.

1.2 Actuator Saturation

In control engineering, the most commonly used actuators are continuous drive devices, along with some incremental drive actuators, like the stepper motors (Astrom and Wittenmark (1996)). Saturation nonlinearity with its maximum and minimum operation limits is unavoidable in such actuator devices.

As shown in Figure 1.1, τ is the actuator output, u is the control input. We study actuator saturation that appears in the nonlinear system plant, and the way of its compensation. Compensation technique is based on NN learning capabilities, and is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

different from existing techniques (Astrom and Wittenmark (1995)), (Annaswamy *et al.* (2001)), (Atherton (1995)), (Walgama and Sternby (1990)).

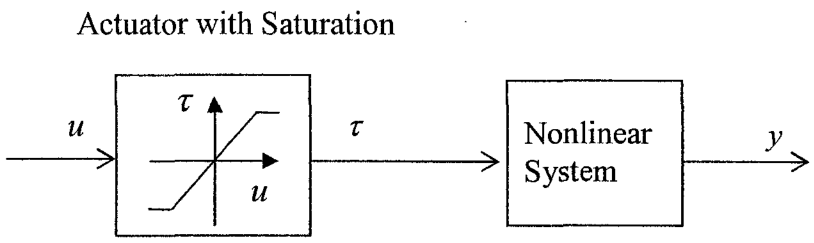

Figure 1.1. Nonlinear plant with actuator saturation.

Figure 1.2 shows the linear saturation $\tau = sat(u)$, where τ and u are scalars. In general, τ and u are vectors. It is commonly assumed that the saturation has a linear rather than nonlinear form, or that there is unity of form between τ and u, which can easily be mathematically modeled. $\tau_{\max}$ and $\tau_{\min}$ are the actuator operation limits. In this dissertation, we assume that saturation nonlinearity is unknown.

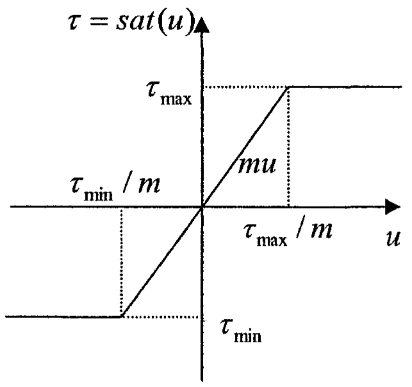

Figure 1.2. Symmetric saturation nonlinearity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Assuming ideal saturation, mathematically, the output of the actuator $\tau(t)$ is given by

$$\tau(t) = \begin{cases} \tau_{\max} : & u(t) \geq \tau_{\max} / m \\ mu(t) : & \tau_{\min} / m < u(t) < \tau_{\max} / m \\ \tau_{\min} : & u(t) \leq \tau_{\min} / m \end{cases}, \tag{1.1}$$

where $\tau_{\max}$ is the chosen positive, and $\tau_{\min}$ is the negative saturation limits. If $u(t)$ falls outside the range of the actuator, actuator saturation occurs, and the control input $u(t)$ can not be fully implemented by the actuator. The control that can not be implemented by the actuator, denoted as $\delta(t)$, is given by

$$\delta(t) = \tau(t) - u(t) = \begin{cases} \tau_{\max} - u(t) : & u(t) \geq \tau_{\max} / m \\ (m-1)u(t) : & \tau_{\min} / m < u(t) < \tau_{\max} / m \\ \tau_{\min} - u(t) : & u(t) \leq \tau_{\min} / m \end{cases}. \tag{1.2}$$

From (1.2), the nonlinear actuator saturation can be described using $\delta(t)$ (Annaswamy *et al.* (2001)), (Atherton (1995)), (Johnson and Calise (2001)), (Kosut (1983)). In this dissertation, NN is used to approximate the function $\delta(t)$.

An overview of normal saturation compensation strategies (Tharayil and Alleyne (2001)) is presented next.

Variable Limit PI-Controller: This was developed by (Safaric, *et al.* (1991)) for systems with plant input saturation that use PI-controllers. A variable limit is introduced in the integral branch of the controller to ensure that the control effort does not exceed the saturation limit. The logic used in this method is as follows:

1. at each time step, calculate $u_p = k_p e$ and $u_i' = k_i \int e$. where $e = ref - y$.
2. if $e = 0$, then $u_i = u_i'$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

3. if $e > 0$,

    a. if $u_{\max} - u_p > u_i'$ (no saturation), then $u_i = u_i'$

    b. if $u_{\max} - u_p < u_i'$ (saturation), then $u_i = u_{\max} - u_p$ (max. value)

4. if $e < 0$,

    a. if $u_{\min} - u_p < u_i'$ (no saturation), then $u_i = u_i'$

    b. if $u_{\min} - u_p > u_i'$ (saturation), then $u_i = u_{\min} - u_p$ (max. value)

5. $u = u_p + u_i$

In other words, the integral term, u_i, is set to be $\max(u_i, u_{\max} - u_p)$ at every time step. The advantages of this VLPI controller are (1) it eliminates overshoot, (2) response time and system robustness can be improved by use of higher loop gains without exaggerated negative effects, (3) besides the feedback loop in the integral branch, this design is identical to the linear design.

The next four control strategies are designed for a system with plant input saturation that uses PID controllers. A detailed account of the following four anti-windup strategies can be found in (Bohn, *et al.* (1995)).

Conditional Integrator: In this method, the integration is switched on or off depending on certain conditions. These conditions can be the size of the control signal or control error. Best results are given by a method where integration is suspended when the actuator saturates and the control error and the control signal have the same sign.

Limited Integrator: In this method, a feedback signal is created from the integrator output by feeding the integrator output through a dead zone with a high gain. The dead zone is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

to reduce the integrator input, as shown in the figure below. To allow the full linear range of the actuator, the dead zone has to be the same as the linear range of the actuator. With sufficiently high dead-zone gains, the integrator output will be effectively limited to the dead zone.

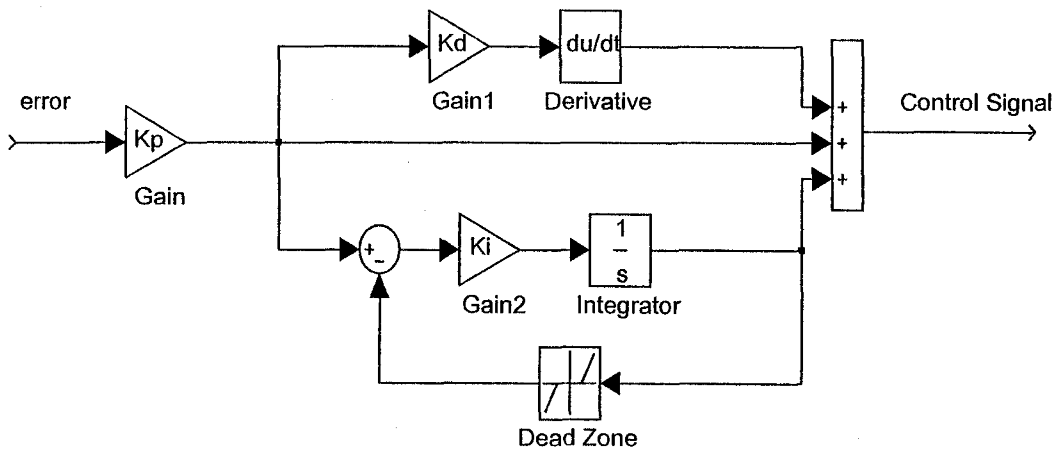

Figure 1.3. A schematic of the limited integrator design.

Tracking Anti-Windup: In this "classical" anti-windup method, once the controller output exceeds the actuator limits, a feedback signal is generated from the difference of the saturated and unsaturated control signals and used to reduce the integrator output. This saturation may either be the actual saturation in the actuator, or the model used in the controller. The figures below show two structures of the tracking anti-windup PID controller. Limiting the controller output, as shown in Figure 1.4a, may limit the speed of the actuator. To account for this effect, Figure 1.4b shows a system where the unrestricted control signal is applied to the process and dead zone is used to generate the feedback signal.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

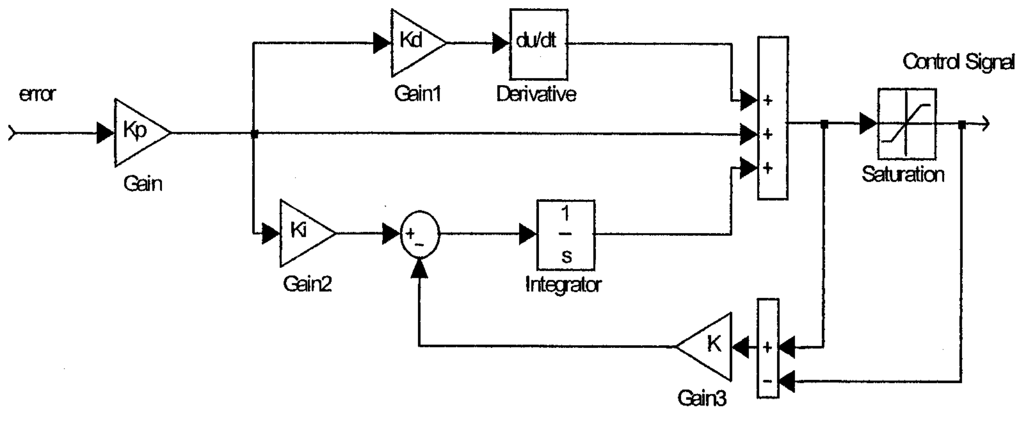

Fig. a

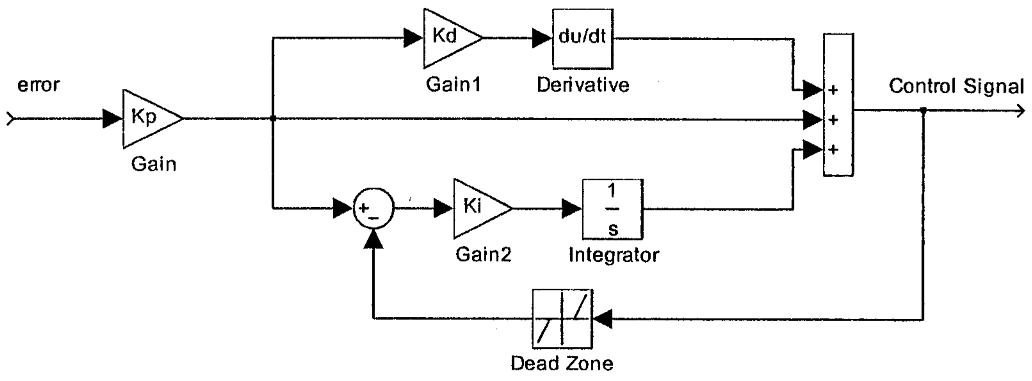

Fig. b

Figure 1.4. Tracking anti-windup control scheme

Modified Tracking Anti-Windup: In the "classical" tracking anti-windup controller design, a very high initial controller output (due to the Proportional and Derivative terms) will give a large feedback signal to the integrator. This generated feedback signal can drive the integrator to a large negative value to bring the controller back to the linear range. As time increases, the PD term will decrease, but the integrator term will not increase fast enough to compensate for this, thus resulting in slow response. The

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

modified tracking anti-windup method avoids this slow response. To do this modification, an additional limit on the proportional-derivative part of the control signal used to generate the anti-windup feedback signal has been introduced. This method can be interpreted as holding the integral action until the control signal from the proportional and derivative action returns to the linear range and then setting integrator to run. The integrator will therefore not be driven negative, and the disadvantage of the tracking method, a very slow step response for a high dead-zone gain, can be avoided. The following figure gives the structure of this controller:

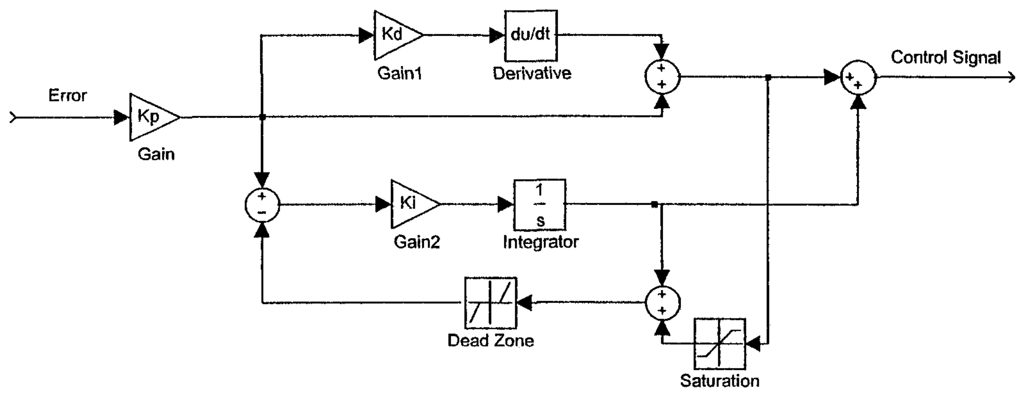

Figure 1.5. Modified tracking anti-windup method

All the above anti-windup schemes assume that the saturation is a result of the integral term, and apply corrective actions to the integral terms.

1.3 Characteristics of Underactuated System

Flexible link robot manipulators play an important role in the modern industrial and space robotic applications with the light weight structure, fast time response and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

underacturated system characteristics. Due to a high nonlinearity, a non-minimum phase and unmodeled dynamics, they present the challenge in a control systems design and modeling compared to the rigid robot. The normally used Euler-Bernoulli model of flexible link is a fourth-order partial differential equation (PDE) system that can lead to infinite number of flexible modes. Many researches approximate the PDE by a system of ordinary differential equation (ODE) through assumed modes and finite element method (Ge, Lee and Zhu (1996)), (Gutierrez, Lewis and Lowe (1998)), (Lewis, Jagannathan, and Yesildirek (1999)), (Sun *et al.* (2003)), (Talebi, Patel, and Khorasani (2001)), (Talebi, Khorasani and Patel, (2002)). A high accuracy model of the flexible link requires a large number of flexible modes.

For a rigid robot arm, tip trajectory control is equivalent to control of the actuator of the rigid mode (joint). However, for a satisfactory control of a flexible link, additional reliable control of the flexible modes should be considered to handle the unavoidable or most probably unbounded vibration of the flexible modes. The additional control issue arises from the noncollocated nature of the sensors and actuator, namely, the zero dynamics is unstable, which makes the non-minimum phase nonlinear system. At the same time, the above mentioned model truncation method yields to the unmodeled dynamics in the mathematical ODE model, and can cause the control spillover effect. As a result, the exact tracking of the flexible link robot is a challenging problem.

To tackle the control problem of flexible link manipulators, different control methods were used including feedback linearization (Lewis, Jagannathan, and Yesildirek (1999)), integral manifold approach (Moallem, Khorasani, and Patel (1997)), output redefinition method (Talebi, Patel, and Khorasani (2001)), singular perturbation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

(Siciliano and Book (1988)) and infinite dimensional approach without considering the flexible link model truncation and simplification (Luo (1993)), (Zhu, Lewis and Hunt (1994)). Moalllem *et al.* (1998) addressed an inversion based robust controller to compensate the parameter uncertainty and to achieve small tip tracking error. Talebi *et al.* (2002) developed "feedback-error-learning" NN-based controller for the tip position tracking. Output redefinition is always used to overcome the non-minimum phase characteristic of the flexible link system (Saber (2000)), in which the output of the system was redefined on the flexible link between the joint and the tip, to ensure the stable zero dynamics. Singular perturbation (Gutierrez, Lewis and Lowe (1998)), (Lewis, Jagannathan, and Yesildirek (1999)), (Sun *et al.* (2003)), (Siciliano and Book (1988)), (Siciliano, Prasad, and Calise (1992)) provides a systematic approach on modeling. It can decompose the flexible link dynamics into a slow subsystem of equivalent rigid robot model and a fast subsystem of flexible model. Based on a singular perturbation, Lewis, Jagannathan, and Yesildirek (1999) presented a modified joint angle output tracking and described a controller which includes a singular perturbation inner loop for stabilization of the fast dynamics, and a neural network inner loop for feedback linearization of the rigid dynamics. An experiment implementation (Gutierrez, Lewis and Lowe (1998)) showed the NN controller can cause the tracking error almost to zero value. Sun *et al.* (2003) developed a dynamic neuron-fuzzy adaptive controller to approximate the slow rigid dynamics of the flexible-link manipulator, and a fuzzy PD controller to stabilize the elastic dynamics.

In Chapter 6, we consider the flexible link manipulator controller design subject to the constrained control input. The imposition of the saturation constraint introduces

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

another unmodeled or unknown nonlinearity in the flexible link model. An additional difficulty is that saturation nonlinearity affects the slow and fast subsystems

Based on a singular perturbation approach and a two-time scale decomposition, this dissertation proposes a NN-based scheme for saturation control for a slow subsystem of the equivalent rigid flexible link, and a NN-based compensator for the fast subsystem of flexible modes with unknown parameters. The approach is the extension of a previous work in actuator saturation control (Gao and Selmic (2004)) and flexible link control (Lewis *et al.* (1999)).

1.4 Organization of the Dissertation

This dissertation is organized as follows. Chapter 2 provides the preliminary remarks and definitions. The four component chapters from Chapter 3 to Chapter 6 are composed of the submitted journal/conference papers. Every chapter is self-contained.

In Chapter 3, neural net (NN)-based actuator saturation compensation scheme for the nonlinear systems in Brunovsky canonical form is presented. The scheme that leads to stability, command following and disturbance rejection is rigorously proved, and verified using a nonlinear system of "pendulum type". On-line weights tuning law, the overall closed-loop performance, and the boundedness of the NN weights are derived and guaranteed based on Lyapunov approach. The actuator saturation is assumed to be unknown, and the compensator is inserted into a feedforward path. The simulation results indicate that the proposed scheme can effectively compensate for the saturation nonlinearity in the presence of system uncertainty.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Chapter 4 addresses an adaptive NN-based saturation compensator for a class of motion control systems. The designed intelligent saturation compensator can also apply to other types of control input distortion than saturation as long as the actuator output is a bounded, static function of the current control input. The controller does not require a saturation model to be known. The NN controller does not require preliminary off-line training. After some initial time, the NN controller learns saturation nonlinearity and adjusts its weights to prevent the control signal from being saturated. Rigorous stability proofs are given using Lyapunov theory. The adaptive NN weight tuning law is the same as in (Lewis *et al.* (1999)). The paper also shows that Lewis's intelligent controller can be effective for the saturation nonlinearity in the motion control systems.

Chapter 5 describes the implementation of the neural net (NN) actuator saturation compensation scheme for nonlinear systems using Simulink (Dabney and Harman (2003)) S-function. Simulink-based S-function blocks are developed, which are equivalent to the theoretical NN saturation compensation schemes. Such control system platform can be used for real-time code generation using Real-Time Workshop Toolbox. The proposed scheme consists of a NN saturation compensator and a PD controller. The NN compensator acts as an intelligent anti-windup compensation for the unknown actuator saturation nonlinearity. A customized Simulink model for the NN saturation compensator of the nonlinear systems is provided, that is suitable for real-time compensator implementation.

In Chapter 6, a robust neural network (NN) composite saturation compensation scheme is presented for the trajectory tracking and vibration suppression of a flexible link robot arm. The scheme is based on a singular-perturbation technique and can

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

accommodate the unknown disturbance and the saturation constraints. The saturation compensator is composed of a robust fast subcompensator for stabilization of the fast flexible modes, and a slow subcompensator consisting of an outer-loop PD tracking controller and robustifying term for stable tracking control of the rigid modes. The actuator saturation is assumed unknown. With respect to the slow dynamics, the NN-based compensator is inserted into the feedforward path. To compensate the saturation in the fast dynamics, the NN-based robust saturation compensator scheme is developed. No linearity in the unknown parameters is necessary, and no off-line NN learning is needed. The stability analysis is based on Lyapunov theory. Simulation results indicate that the proposed scheme can effectively compensate the saturation nonlinearity in the presence of the uncertainty for underactuated systems.

The conclusion remarks is drawn in Chapter 7, as well as the future recommended research topics in the area of intelligent control.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 2

PRELIMINARY REMARKS AND DEFINITIONS

Let $\Re$ denote real numbers, $\Re^n$ denote the real n vector, and $\Re^{m\times n}$ denote the real $m\times n$ matrices. Let S be a compact simply connected set of $\Re^n$. Let $C(S)$ defined to be With map $\{f:S\to\Re^m \mid f \text{ is continuous}\}$, The initial condition is $x_0 \equiv x(t_0)$, let the equilibrium point be x_e, and let U_{x_e} be the neighborhood of x_e (Lewis, Abdallah, and Dawson (1993)).

Definition 1 (Vector and Matrix Norms) (Lewis, Abdallah, and Dawson (1993)): By $\|\ \|$ is denoted any suitable vector norm. When it is required to be specific, we denote the P-norm by $\|\ \|_P$. For $f:S\to\Re^m$, define the supremum norm of $f(x)$, over S as

$$\|f\|_{\sup} = \sup_{x\in S}\|f(x)\|, \tag{2.1}$$

Given $A=[a_{ij}], B\in\Re^{m\times n}$ the Frobenius norm of A is defined by

$$\|A\|_F^2 = tr(A^T A) = \sum_{i,j} a_{ij}^2 \tag{2.2}$$

with $tr()$ the trace. The Frobenius norm is compatible with the 2-norm so that $\|Ax\|_2 \le \|A\|_F \|x\|_2$. The associated inner product is $<A,B>_F = tr(A^T B)$,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

and $tr(AB) = tr(BA)$. Suppose A is positive definite, then for any $B \in \Re^{mxn}$,

$$tr(BAB^T) \geq 0 \tag{2.3}$$

$$\frac{d}{dt}\{tr(A(x))\} = tr\left(\frac{dA(x)}{dt}\right) \tag{2.4}$$

Definition 2 (Uniformly Ultimate Boundedness (UUB)) (Lewis, Jagannathan, and Yesildirek: (1999)) Consider the nonlinear system:

$$\dot{x} = g(x,t) \tag{2.5}$$

with state $x(t) \in \Re^n$. The equilibrium point x_e is said to be uniformly ultimately bounded if there exists a compact set $S \subset \Re^n$ such that, for all $x_0 \in S$ there exists an $\varepsilon > 0$, and a number $T(\varepsilon, x_0)$ such that $\|x(t) - x_e\| \leq \varepsilon$ for all $t \geq t_0 + T$. That is, after a transition period T, the state $x(t)$ remains within the ball of radius ε around x_e.

Definition 3 (NN Universal Approximation Function) (Corless and Leitmann (1982)): Consider two-layer NN, consisting of two layers of tunable weights. The hidden layer has L neurons, and the output layer has m neurons,

$$y = W^T \sigma(V^T x + v_0) \tag{2.6}$$

The multilayer NN is a nonlinear mapping from input space $\Re^n$ into output space $\Re^m$, where

$$V = [V_{ji}], \quad j = 1, 2, \cdots, n; \quad i = 1, 2, \cdots, L \tag{2.7}$$

$$W = [W_{ik}], \quad i = 0, 1, 2, \cdots, L; \quad k = 1, 2, \cdots, m \tag{2.8}$$

$$x = [x_1, x_2, \cdots, x_n] \tag{2.9}$$

$$y = [y_1, y_2, \cdots, y_m] \tag{2.10}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$v_0 = [v_{01}, v_{02}, \cdots, v_{0L}]^T \quad . \tag{2.11}$$

To include the thresholds in the matrix W, the vector activation function is defined as $\sigma(W) = [1, \sigma(W_1), \sigma(W_2), \cdots \sigma(W_L)]^T$, where $W \in \Re^L$. Tuning of the weights W then also includes tuning of the thresholds.

Many well known results indicate that any sufficiently smooth function can be approximated arbitrary closely on a compact set using a two-layer NN with appropriate weights (Scanner and Slotine (1991)), (Seshagiri and Khalil (2000)), (Hovakimyan, Nardi and Calise (2001), (Corless and Leitmann (1982)), (Cybenko (1989)), (Funahashi (1989)). Function $\sigma(\cdot)$ could be any continuous sigmoidal function. NN *universal approximation property* defines that any continuous function can be approximated arbitrarily well using a linear combination of sigmoidal functions (Cybenko (1989)), namely,

$$f(x) = W^T \sigma(V^T x + v_0) + \varepsilon(x) \,, \tag{2.12}$$

where the $\varepsilon(x)$ is the NN approximation error. The reconstruction error is bounded on a compact set S by $\|\varepsilon(x)\| < \varepsilon_N$. Moreover, for any ε_N, one can find a NN such that $\|\varepsilon(x)\| < \varepsilon_N$ for all $x \in S$.

The first layer weights V are selected randomly and will not be tuned. The second layer weights W are tunable. The approximation holds (Igelnik and Pao (1995)) for such a NN, with approximation error convergence to zero of order $\mathrm{O}\left(C / \sqrt{L}\right)$, where L is the number of the hidden layer nodes (basis functions), and C is independent of L.

The approximating weights W are ideal target weights, and it is assumed that they are bounded so that $\|W\|_F \le W_M$.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Definition 4 (Feedback linearizable) (Nam (1999)): Consider a single input and single output (SISO) system

$$\dot{x} = f(x) + g(x)u, x \in \Re^n, \tag{2.13}$$

it is feedback linearizable if there exists a local diffeomorphism, namely, a coordinate transforming map $T : U_{x_e} \to T(U_{x_e})$, and a feedback $u = \beta(x)v + \alpha(x)$, and $\alpha, \beta : U_{x_e} \to \Re$ such that, in the new coordinates $z = T(x)$,

$$\dot{z} = Az + bv \tag{2.14}$$

where (A, b) is a Brunovsky controllable canonical pair, *i.e.*,

$$A = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \end{bmatrix}, b = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{2.15}$$

It is well known that equation (2.13) is feedback linearizable if and only if the vector fields $\{g, ad_f g, \cdots, ad^{n-1}{}_f g\}$ are linear independent and span $\{g, ad_f g, \cdots, ad^{n-2}{}_f g\}$ is an involutive distribution (Isidori (1989)).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 3

SATURATION COMPENSATOR FOR A CLASS OF NONLINEAR SYSTEMS

3.1 Nonlinear Systems Dynamics

Consider the single input single output (SISO) nonlinear systems with state space representation

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= f(x) + g(x)\tau \\ y &= x_1 \end{aligned} \tag{3.1}$$

with $x = [x_1, x_2, \cdots, x_n]^T$, $f : \Re^n \to \Re$, an unknown smooth function, which contains the parameter uncertainties that are not necessarily linear parameterable; $g : \Re^n \to \Re$, a known smooth function; Function $\tau \in \Re$ the control input. Equation (3.1) is said in the Brunovsky canonical form which is feedback linearizable (Nam (1999)), (Lewis *et al.* (1993)).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Assumption 3.1: Function $g(x)$ is assumed to be known, such that $|g(x)| > \varepsilon$, where $\varepsilon > 0, \varepsilon \in \Re$.

Define the desired state vector, $x_d(t)$, as

$$x_d(t) = [y_d, \dot{y}_d, \cdots, y_d^{(n-1)}]^T. \tag{3.2}$$

Assumption 3.2 (Bounded Desired Trajectory): The desired trajectory $x_d(t)$ is bounded and continuous, and $\|x_d(t)\| \le Q$ with Q known scalar bound.

3.2 Tracking Error Dynamics and Feedback Linearization

Define the state tracking error vector, $e(t) \in \Re^n$ as

$$e(t) = x(t) - x_d(t). \tag{3.3}$$

Let us define a filtered tracking error $r \in \Re$ as

$$r = K^T e, \tag{3.4}$$

where $K = [k_1, k_2, \ldots k_{n-1}, 1]^T$ is appropriately chosen coefficient vector, so that $e \to 0$ exponentially as $r \to 0$ (Slotine and Li (1991)). Then, the time derivative of the filtered tracking error can be written as

$$\dot{r} = f(x) + g(x)\tau + Y_d, \tag{3.5}$$

where $Y_d = -y_d^{(n)} + \sum_{i=1}^{n-1} k_i e_{i+1}$.

Consider the saturation nonlinearity equation (3.2). The following n-th order nonlinear system dynamics

$$x_1^{(n)} = f(x) + g(x)\tau, \tag{3.6}$$

which is equivalent to (3.1), can be described as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$x_1^{(n)} = f(x) + g(x)u + g(x)\delta, \tag{3.7}$$

in which the saturation will be treated as he system disturbance (Annaswamy *et al.* (2001)).

Similarly, in terms of the filtered tracking error above system dynamics can be described as follows

$$\dot{r} = f(x) + g(x)(u + \delta) + Y_d, \tag{3.8}$$

where Y_d is a known function of the tracking error and the desired states.

According to feedback linearization (Slotine and Li (1991)), (Lewis *et al.* (1999)), choose the tracking control law as

$$w = \frac{1}{g(x)}(-\hat{f} - Y_d + v - K_v r), \tag{3.9}$$

where $\hat{f}$ is the fixed approximation of function $f(x)$. The functional estimation error is given by

$$\tilde{f} = f - \hat{f}. \tag{3.10}$$

Approximation $\hat{f}$ is fixed in this dissertation and will not be adapted. v is a robust term chosen for the disturbance rejection. K_v is the feedback gain and normally selected as scalar. The control law u is then the tracking controller with the saturation compensator, as shown in the Figure 3.1.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

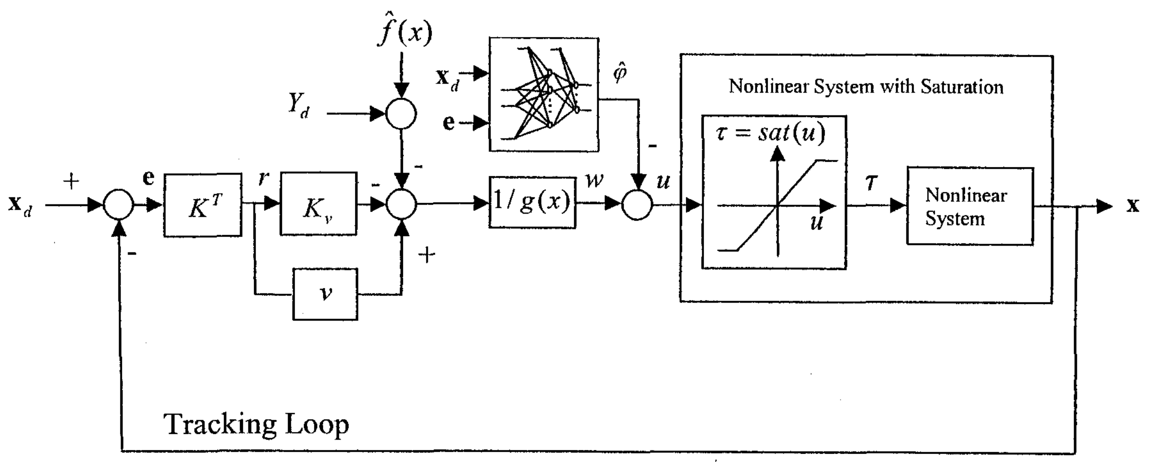

Figure 3.1. General nonlinear system and NN saturation compensator.

$$u = w - \hat{\varphi}, \tag{3.11}$$

where $\hat{\varphi}$ is the approximation of modified saturation nonlinear function

3.3 NN Saturation Compensator

Using the general NN function approximation property, there exists a NN that closely approximates the modified saturation nonlinear function $\delta(x)$.

$$\delta = W^T \sigma(V^T x_{NN}) + \varepsilon \ . \tag{3.12}$$

Implementer NN is actually an approximation of the ideal NN (3.12), and is given by

$$\hat{\varphi} = \hat{W}^T \sigma(V^T x_{NN}) \ , \tag{3.13}$$

where the NN weights approximation error is

$$\tilde{W} = W - \hat{W} \ . \tag{3.14}$$

Input to the NN saturation compensator is chosen as $x_{NN} \equiv [x_d, e]^T$.

Assumption 3.3 (Bounded Ideal NN Weights): The ideal NN weights W are bounded so that $\|W\| \le W_M$, with W_M known bounds.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Assumption 3.4 (Bounded Estimation Error): The nonlinear unknown function $f(x)$'s estimate $\hat{f}(x)$ is assumed known, so that that the functional estimation error $\tilde{f}(x)$ satisfies

$$\left\|\tilde{f}(x)\right\| \le f_M(x) \tag{3.15}$$

for some known function bounds $f_M(x)$ (Isidori (1989)), (Narendra (1991)).

This assumption is not unreasonable (Corless and Leitmann (1982)), (Gao and Selmic (2004)), (Selmic and Lewis (2000)), as in practical systems the bound f_M can be computed knowing the upper bound of variables such as payload masses, frictional effects, and so on.

Using control law (3.11) and (3.13), and substituting into (3.8), overall closed-loop error dynamics is

$$\dot{r} = \tilde{f} + g(x)\tilde{W}^T\sigma(V^T x_{NN}) + v - K_v r + g\varepsilon \ . \tag{3.16}$$

3.4 Weights Tuning Law for Guaranteed Tracking Performance

The purpose of NN saturation compensator is to design proper control laws [*i.e*,. the input u in (3.11)] and stable on-line NN weights update tuning rules, to guarantee the tracking performance of the overall closed-loop systems under the unknown saturation nonlinearity. Moreover, an NN saturation compensator, if designed properly, should reduce the deleterious effect of saturation nonlinearity on the overall system performance.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Theorem 3.1 (Tuning of NN Compensator): Given the system in (3.16) and Assumptions 3.1-3.4, choose the tracking control law (3.9), plus the saturation compensator (3.11), (3.13), and the robustifying term as

$$v(t) = -f_M(x) sign(r), \tag{3.17}$$

where the $f_M(x)$ are bounds on the functional estimation error, and *sign*(.) is standard sign function. Let the estimated NN weights be provided by the NN tuning algorithm

$$\dot{\hat{W}} = S\sigma(V^T x_{NN}) r\, g(x) - kS|r|\hat{W}, \tag{3.18}$$

where

$S = S^T > 0$: any constant matrices representing the learning rates of the NN, and k: small scalar positive design parameter. By properly selecting the control gains and the design parameters, the filtered error $r(t)$ and the NN weights $\hat{W}$ are UUB (Uniformly Ultimately Bounded).

Proof: Choose the Lyapunov function candidate as

$$L = \frac{1}{2} r^2 + \frac{1}{2} tr(\tilde{W}^T S^{-1} \tilde{W}) \tag{3.19}$$

Differentiating yields

$$\dot{L} = r\dot{r} + tr(\tilde{W}^T S^{-1} \dot{\tilde{W}}) \tag{3.20}$$

Whence substitution from (3.16) yields

$$\begin{aligned} \dot{L} &= -K_v r^2 + r\tilde{f} + rg(x)\tilde{W}^T \sigma(V^T x_{NN}) + rv + rg\varepsilon + tr(\tilde{W}^T S^{-1} \dot{\tilde{W}}) \\ &= -K_v r^2 + r(\tilde{f} + v + g\varepsilon) + tr(\tilde{W}^T (S^{-1}\dot{\tilde{W}} + \sigma(V^T x_{NN}) rg(x))) \end{aligned} \tag{3.21}$$

Applying the NN tuning rules, selected Lyapunov function is simplified to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\dot{L} = -K_v r^2 + r(\tilde{f} + v + g\varepsilon) + k|r| tr(\tilde{W}^T \hat{W}) \tag{3.22}$$

Using (3.17) one has

$$\dot{L} \leq -K_v r^2 + k|r| tr(\tilde{W}^T (W - \tilde{W})) - |r| f_M + |r|\tilde{f} + |r| g\varepsilon_N \tag{3.23}$$

Using the inequality,

$$tr\left[\tilde{X}^T (X - \tilde{X})\right] \leq \left\|\tilde{X}\right\|_F \left\|X\right\|_F - \left\|\tilde{X}\right\|_F^2, \tag{3.24}$$

the inequality (3.23) can be written as

$$\dot{L} \leq -K_v |r|^2 + k|r|(\left\|\tilde{W}\right\|_F \left\|W\right\|_F - \left\|\tilde{W}\right\|_F^2) + |r| g\varepsilon_N$$

$$\leq |r| \left\{ -K_v |r| + k(\left\|\tilde{W}\right\|_F W_M - \left\|\tilde{W}\right\|_F^2) + g\varepsilon_N \right\}.$$

$$= |r| \left\{ -K_v |r| - k(\left\|\tilde{W}\right\|_F - \frac{1}{2} W_M)^2 + \frac{1}{4} k W_M^2 + g\varepsilon_N \right\}, \tag{3.25}$$

which is guaranteed to remain negative as long as

$$K_v |r| \geq \frac{k}{4} W_M^2 + g\varepsilon_N, \tag{3.26}$$

which is equivalent to

$$|r| \geq \frac{\frac{k}{4} W_M^2 + g\varepsilon_N}{K_v}, \tag{3.27}$$

or

$$k(\left\|\tilde{W}\right\|_F^2 - \frac{1}{2} W_M)^2 \geq \frac{k}{4} W_M^2 + g\varepsilon_N, \tag{3.28}$$

which is equivalent to

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\left\|\tilde{W}\right\|_F \geq \left(\frac{\sqrt{\frac{k}{4}W_M^2 + g\varepsilon_N} + \frac{1}{2}W_M}{k} \right)^{\frac{1}{2}} . \tag{3.29}$$

The following remarks are relevant.

Modified Backpropagation Terms. The first term of (3.18) is modified version of the standard backpropagation algorithm. The k term corresponds to the e-modification (Narendra and Annaswamy (1987)), to guarantee bounded parameter estimates.

Bounds on the Tracking Error and NN Weights Estimation Errors. The right-hand side of inequality (3.27) can be taken as a practical bound on the tracking error in the sense that $r(t)$ will never stray far above it. Note that the stability radius may be decreased by any amount by increasing the PD gain K_v. It is noted that PD, PID, or any other standard controller does not posses this property when saturation nonlinearity is present in the system. Moreover, it is difficult to guarantee the stability of such a highly nonlinear system using only PD. Using the NN saturation compensation, stability of the system is proved, and the tracking error can be kept arbitrarily small by increasing the gain K_v. The NN weights errors are fundamentally bounded in terms of W_M. The tuning of parameter k offers a design tradeoff between the eventual relative magnitudes of $\left\|\tilde{W}\right\|_F$ and r.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

NN Weights Initialization. The weights V are set to random values. It is shown in (Igelnik and Pao (1995)) that for such a NN, termed random variable functional link (RVFL) NN, the approximation property holds. The weights W are initialized at zero. Then the PD loop in Figure 3.1 holds the system stable until the NN begins to learn.

Other Types of Control Input Nonlinearities. As the actuator output τ stays constant and bounded when the control input u is saturated, the NN is used to approximate the unimplemented function $\delta = \tau - u$. This convertion shows that the proposed NN intelligent compensator can be applied to other types of control input distortion and nonlinearities as long as the actuator output τ is a bounded function of the current control input u.

NN Controller in Unsaturated Range. Within the actuator unsaturated range, actuator output τ is linearly proportional to the input u. The NN and robust term are still used to deal with the system disturbance and uncertainty. Simulation results confirm that NN output has small value while the control signal is in linear range.

Linearity in the Model Parameters(LIP). The proposed NN-based saturation compensator does not require LIP. The standard adaptive control techniques for actuator nonlinearity compensation require this assumption (Tao and Kokotovic (1996)). The LIP requirement is a severe restriction for practical systems since one has to conduct some preliminary analysis to determine a regression matrix.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Intelligent Anti-Windup Saturation Compensation. The proposed method utilizes an NN controller to compensate for the saturation nonlinearity effects. Initially, the NN controller "learns" and adjusts its weights to prevent the control signal from being saturated. After the initial learning period, which will be demonstrated below in the simulation, the NN signal effectively keeps the control signal within saturation bounds. Therefore, the proposed NN control scheme presents a form of *Intelligent Anti-Windup Saturation Compensation*.

3.5 Simulation

The simulation was performed to verify the effectiveness of the proposed NN compensator. We consider a "generalized pendulum" system (Khalil (2002)) with saturation nonlinearity.

The numerical simulation program was written in visual C++ and Matlab. The integration method is a fourth-order Runge-Kutta algorithm. The integration time step/sampling interval is using 0.001. The program is running on a Dell OPTIPLEX GX260 computer with Intel Pentium CPU (2.40 GHz, 522, 232 KB RAM).

To show and focus on the functionality of adaptive NN saturation compensator, we choose the robust term $v = 0$, $\hat{f} = f$ in the following "Pendulum type" simulation case.

We consider a nonlinear system of a pendulum type given by

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -5x_1^3 - 2x_2 + \tau \\ y &= x_1 \end{aligned} \tag{3.30}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Control input τ is constrained by the saturation nonlinearity characterized by the parameters

$$\tau_{\max} = 5, \tau_{\min} = -5, m = 1 \ . \tag{3.31}$$

The size of the NN considers the stability, performance, limitation of control efforts, and possible operating conditions (Wai (2003)). The slow convergence of the tracking error is usually due to the small network size. Moreover, if the chosen size is too large, the computation burden increases. A common approach is to start with the smaller NN size, and gradually increase it until satisfactory performance is achieved.

The NN has four, ten and one neurons at the input, hidden and output layers, respectively. The first-layer weights V are selected randomly (Igelnik and Pao (1995)), (Selmic and Lewis (2000)), (Corless and Leitmann (1982)). They are uniformly randomly distributed between -1 and $+1$. These weights represent the stiffness of the sigmoid activation function. The threshold weights for the first layer v_0 are uniformly randomly distributed between -15 and +15. The threshold weights represent the bias in activation functions' positions. The second layer weights W are initialized to zero or any random number, and the effect of the inaccurate initialization number can be retrieved by the on-line weights tuning law methodology.

Tracking loop controller parameters are chosen so that $K_v = 10$, $K = [2, 1]^T$. Initial conditions are $[0, 0]^T$, and desired trajectory is given by $x_1(t) = \sin(t), x_2(t) = \cos(t)$. The position tracking errors $e_1(t)$, and $e_2(t)$ are shown in the Figure 3.2 without the saturation compensator. The control input signal $\tau(t)$ is shown in the Figure 3.3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

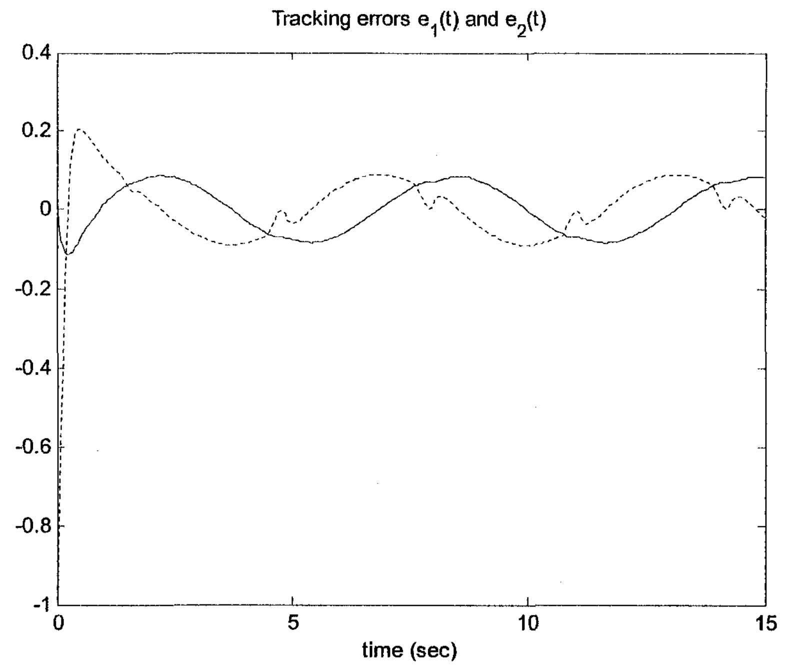

Figure 3.2. Tracking errors e_1 (solid) and e_2 (dotted) without saturation compensator.

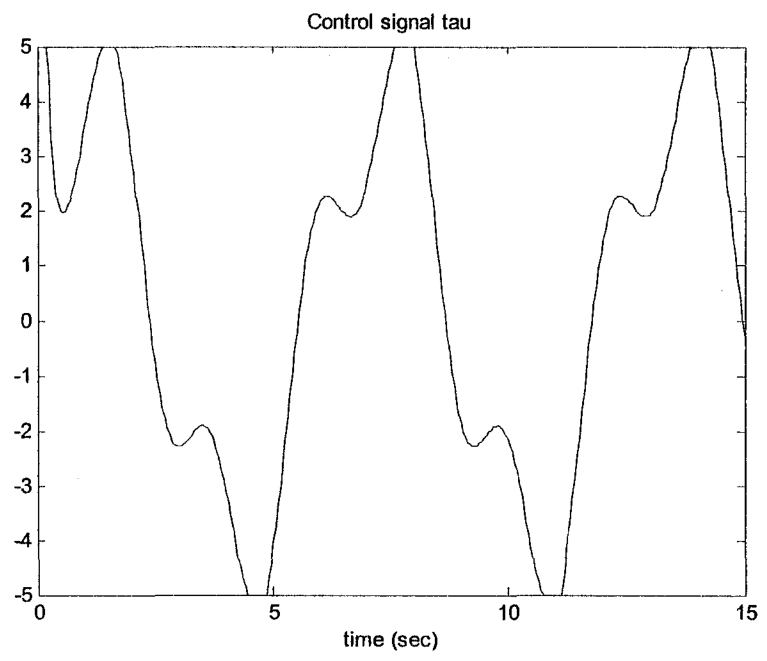

Figure 3.3. Control signal τ without saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The NN saturation compensator weights tuning parameters are chosen as

$$k = 0.0001\,, \quad S = 5\ . \tag{3.32}$$

With the NN saturation compensation included, the tracking errors are given in the Figure 3.4, control signal $\tau(t)$ in Figure 3.5, and Neural network output in Figure 3.6.

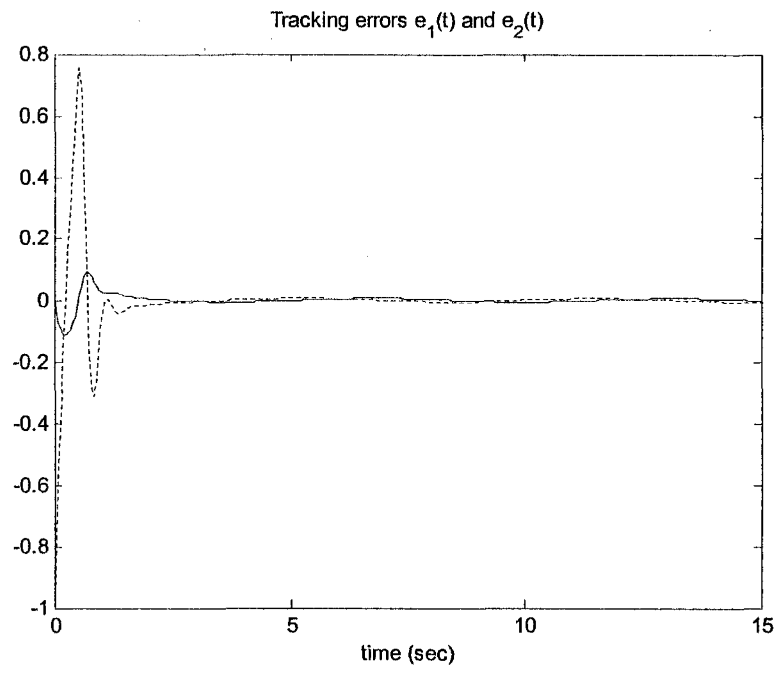

Figure 3.4. Tracking errors e_1 (solid) and e_2 (dotted) with NN saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

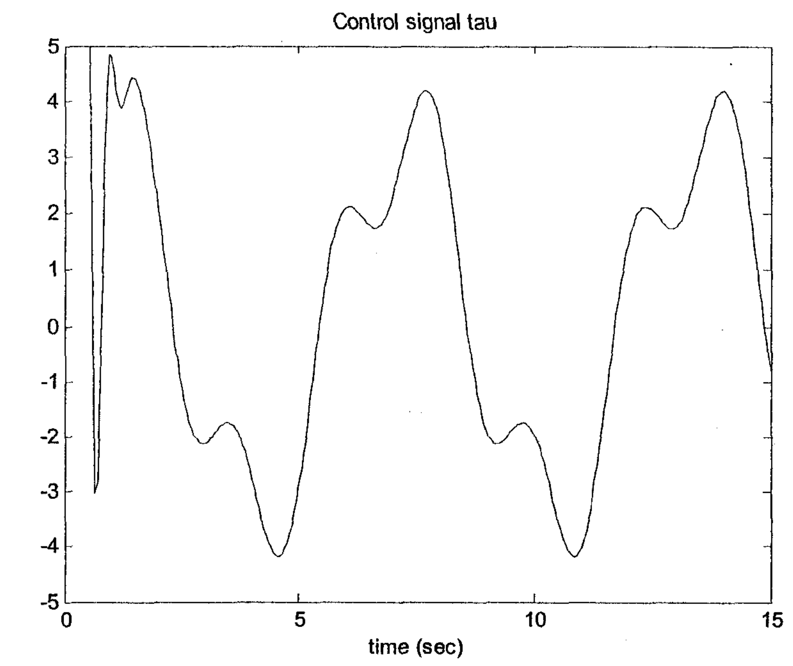

Figure 3.5. Control signal τ with NN saturation compensator.

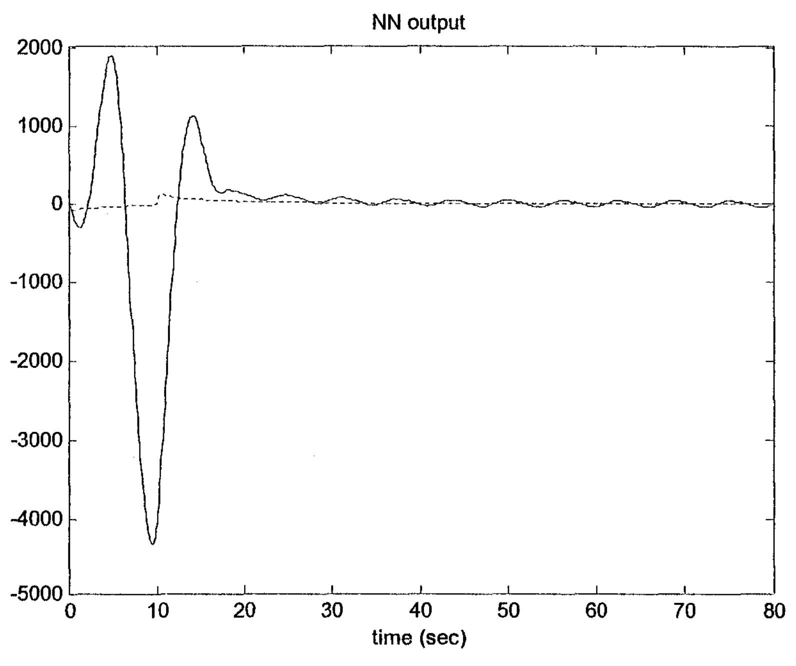

Figure 3.6. NN output with NN saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

From the above simulation, it is clear that the proposed scheme can effectively compensate the saturation nonlinearity in a class of nonlinear systems. Note also that after some initial time required for NN to learn the unknown saturation nonlinearity, the NN saturation compensator effectively prevents the control signal from reaching saturation limits. NN is trained using the filtered tracking error, trying to minimize the same. The expection of small tracking error is achieved by keeping control signal under saturation limit range. Design tradeoff is that intelligent NN saturation compensator requires extra controller complexity and extra computational power

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 4

ADAPTIVE NN SATURATION CONTROL IN MOTION CONTROL SYSTEMS

4.1 Dynamics of Mechanical Motion Tracking System

We consider here the control of mechanical systems in the presence of actuator saturation nonlinearity. Torque control actuators are subject to saturation limits, and this needs to be considered when controllers are designed for such systems.

A general dynamics of a mechanical system usually in Lagrangian form can be written as

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + G(q) + \tau_d = \tau \tag{4.1}$$

where $q(t) \in \Re^n$ is a vector describing position and orientation, $M(q)$ is the inertia matrix, τ is the vector of actuator control torques, $V_m(q,\dot{q})$ is the coriolis/centripetal matrix caused by the motion of the links, $G(q)$ is the gravity vector, and $\tau_d(t) \in \Re^n$ represents disturbances.

The system equation (4.1) satisfies some important physical properties as a consequence of the fact that they constitute a Lagrangian system. These properties (Lewis *et al.* (1999)) are important in control system design and can be described as follows:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Property 1: $M(q)$ is a positive definite symmetric matrix bounded by $m_1 I \le M(q) \le m_2 I$, where m_1 m_2 are known positive constants.

Property 2: The norm of the matrix $V_m(q,\dot{q})$ is bounded by $v_m(q)\|\dot{q}\|$ with $v_m(q)$ a known value.

Property 3: The matrix $\dot{M} - 2V_m$ is skew-symmetric. This is due to the fact that the internal forces do no work.

Property 4: The unknown disturbance satisfies $\|\tau_d\| \le \tau_M$, with τ_M a known positive constant

4.2 Tracking Error Dynamics and Adaptive Controller Design

To design a motion tracking controller that cause the mechanical system to track a desired trajectory $q_d(t)$, define tracking error as

$$e(t) = q_d(t) - q(t) \tag{4.2}$$

Assumption 4.1 (Bounded Desired Trajectory): The desired trajectory $q_d(t)$ is bounded and continuous, and $\|q_d(t)\| \le Q$ with Q known scalar bound.

Then, the filtered tracking error as

$$r = \dot{e} + \Lambda e, \tag{4.3}$$

where $\Lambda = \Lambda^T > 0$ is a design parameter matrix, usually selected as a diagonal with large positive entries. At the same time, the control goal is to guarantee the stability of the filter tracking error $r(t)$.

Differentiating r in equation (4.3) and invoking (4.1) yields the mechanical dynamics as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$M\dot{r} = -V_m r - \tau + f(x) + \tau_d , \tag{4.4}$$

where the nonlinear dynamic function is

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) \tag{4.5}$$

Actuator control torques τ are subject to saturation constraints (1.1). A robust saturation controller is given (Lewis *et al.* (1999)). In this dissertation, we use intelligent control techniques for saturation compensation.

Considering the saturation model (1.2), the mechanical dynamics are given by

$$M\dot{r} = -V_m r + f(x) + \tau_d - u - \delta \ . \tag{4.6}$$

Choose the outer tracking controller as in (Gao and Selmic (2003)).

$$w = \hat{f} - v + K_v r \ . \tag{4.7}$$

If approximation of an unknown function *f*, namely $\hat{f}$, is chosen as in (Lewis *et al.* (1999)).

$$\hat{f} = M(\ddot{q} \ + \Lambda\dot{e}) + V_m(\dot{q} + \Lambda e) \ , \tag{4.8}$$

and such tracking controller is called the PD Computed Torque Control Variant (PDCTCV) controller (Lewis *et al.* (1999)), which is restricted by that M, V_m are all known. Here the designated controller is not based on equation (4.8). The control scheme for the mechanical system under actuator saturation consists of a standard tracking controller and an NN saturation compensator, as shown in Figure 4.1.

Applying NN universal approximation property, there exists NN with some ideal weights *W*, that closely approximates the unknown modified saturation function δ

$$\delta = W^T \sigma(V^T x_{NN}) + \varepsilon \ . \tag{4.9}$$

Saturation control is given as

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$u = w - \hat{\delta} \ , \tag{4.10}$$

where $\hat{\delta}$ is the actual realization of the NN compensation function

$$\hat{\delta} = \hat{W}^T \sigma(V^T x_{NN}) \ . \tag{4.11}$$

NN input is selected as $x_{NN} \equiv [q_d, \dot{q}_d, e, \dot{e}]^T$.

Assumption 4.2 (Bounded Ideal NN Weights): The ideal NN weights W are bounded so that $\|W\| \leq W_M$, with W_M known bounds.

Assumption 4.3 (Bounded Estimation Error): The estimate $\hat{f}(x)$ of a nonlinear unknown function $f(x)$ is assumed known, so that that the functional estimation error $\tilde{f} = f - \hat{f}$ satisfies

$$\left|\tilde{f}(x)\right| \leq f_M(x) \tag{4.12}$$

for some known function bounds $f_M(x)$ (Isidori (1989)), (Narendra (1991)).

This assumption is not unreasonable (Corless and Leitmann (1982)), (Gao and Selmic (2004)), (Selmic and Lewis (2000)), as in practical systems the bound f_M can be computed knowing the upper bound of variables such as payload masses, frictional effects, and so on.

Substituting (4.10) and (4.11) into (4.6) gives the closed-loop error dynamics

$$M\dot{r} = -V_m r + \tilde{f}(x) - K_v r + v - \tilde{W}^T \sigma(V^T x_{NN}) - \varepsilon + \tau_d \tag{4.13}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

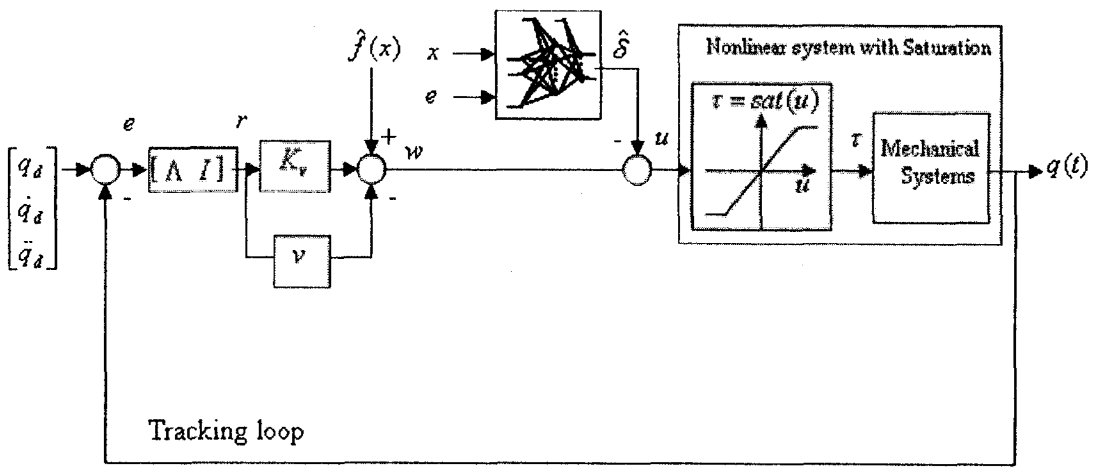

Figure 4.1. Mechanical system with saturation compensator.

4.3 NN Weights Tuning Law for Guaranteed Tracking Performance

The following theorem specifies robust and NN part of controller, such that the closed-loop system is UUB in the presence of the actuator saturation in robot manipulators.

Theorem 4.1 (Tuning of NN Compensator): Given the robot arm dynamics (4.13), Assumptions 4.1-4.3, and Properties 1-4, choose the tracking control law (4.7), and the saturation compensator (4.10), (4.11). Choose the robustifying term as

$$v(t) = -\left(f_M(x) + \tau_M\right)\frac{r}{\|r\|}, \tag{4.14}$$

where the $f_M(x)$ and τ_M are the bounds on functional estimation error and disturbance, respectively. Let the estimated NN weights be provided by the NN tuning algorithm

$$\dot{\hat{W}} = -S\sigma(V^T x_{NN})r^T - kS\|r\|\hat{W} \tag{4.15}$$

where:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$S = S^T > 0$ is any constant representing the learning rates of the NN, k is a small scalar positive design parameter.

By properly selecting the control gains and the design parameters, the filtered error $r(t)$ and the NN weights $\hat{W}$ are UUB (Uniformly Ultimately Bounded).

Proof: Choose the Lyapunov function candidate as

$$L = \frac{1}{2} r^T M r + \frac{1}{2} tr(\tilde{W}^T S^{-1} \tilde{W}) \tag{4.16}$$

Differentiating yields

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r + tr(\tilde{W}^T S^{-1} \dot{\tilde{W}}) \tag{4.17}$$

Using (4.13) yields

$$\begin{aligned} \dot{L} &= r^T(-V_m r + \tilde{f} - K_v r + v + \tau_d - \varepsilon - \tilde{W}^T \sigma(V^T x_{NN})) \\ &\quad + \frac{1}{2} r^T \dot{M} r + tr(\tilde{W}^T S^{-1} \dot{\tilde{W}}) \\ \dot{L} &= -r^T K_v r + r^T(\tilde{f} + v + \tau_d - \varepsilon) + \frac{1}{2} r^T (\dot{M} - 2V_m) r \\ &\quad + tr[\tilde{W}^T (S^{-1} \dot{\tilde{W}} - \sigma(V^T x_{NN}) r^T)] \end{aligned} \tag{4.18}$$

Applying the tuning rule (4.15), robustifying term (4.14), and Property 3, one has

$$\dot{L} = -r^T K_v r + r^T(\tilde{f} + v + \tau_d - \varepsilon) + k\|r\| tr(\tilde{W}^T \hat{W}) \tag{4.19}$$

choose $K_{v\min}$ as the smallest eigenvalues of K_v, so

$$\dot{L} \le -r^T K_{v\min} r + k\|r\| tr(\tilde{W}^T (W - \tilde{W})) - \|r\| \|f_M + \tau_M\| + \|r\| \|\tilde{f} + \tau_d\| + \|r\| \varepsilon_N$$

$$\dot{L} \le -K_{v\min} \|r\|^2 + k\|r\| \|\tilde{W}\|_F (W_M - \|\tilde{W}\|_F) + \|r\| \varepsilon_N$$

$$\le \|r\| \left\{ -K_{v\min} \|r\| + k\|\tilde{W}\|_F W_M - k\|\tilde{W}\|_F^2 + \varepsilon_N \right\}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$= \|r\| \left\{ -K_{v\min} \|r\| - k(\|\tilde{W}\|_F - \frac{1}{2} W_M)^2 + \frac{1}{4} k W_M^2 + \varepsilon_N \right\} \tag{4.20}$$

which is guaranteed negative as long as

$$K_{v\min} \|r\| \geq \frac{k}{4} W_M^2 + \varepsilon_N \tag{4.21}$$

which is equivalent to

$$\|r\| \geq \frac{\frac{k}{4} W_M^2 + \varepsilon_N}{K_{v\min}} \tag{4.22}$$

or

$$k(\|\tilde{W}\|_F^2 - \frac{1}{2} W_M)^2 \geq \frac{k}{4} W_M^2 + \varepsilon_N, \tag{4.23}$$

which is equivalent to

$$\|\tilde{W}\|_F \geq \left(\frac{\sqrt{\frac{k}{4} W_M^2 + \varepsilon_N} + \frac{1}{2} W_M}{k} \right)^{\frac{1}{2}}. \tag{4.24}$$

The remarks in last chapter apply here; also the following remarks are relevant.

NN Intelligent Controller as Saturation Compensator. Tuning law is the same as in (Lewis *et al.* (1999)). This result shows that Lewis' controller can be used as a saturation compensator, provided that the estimate $\hat{f}(x)$ of an unknown function $f(x)$ is known.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

4.4 Simulation

The simulation was performed to verify the effectiveness of the proposed NN saturation compensator. A planar two-serial robotic link arm is chosen and shown in Figure 4.2. The robotic model is provided in (Lewis *et al.* (1999)), (Selmic and Lewis (2001)). Robot dynamics is in the general form given in (4.1). In order to focus on the effects of actuator saturation nonlinearity, gravity and friction are not included in the system model. Yet, the model contains all nonlinear terms arising in the general n-link manipulators.

The numerical simulation program was written in visual C++ and Matlab. The integration method is a fourth-order Runge-Kutta algorithm. The integration time step/sampling interval is using 0.001. The program is running on a Dell OPTIPLEX GX260 computer with Intel Pentium CPU (2.40 GHz, 522, 232 KB RAM).

To show and focus on the functionality of adaptive NN saturation compensator, we choose the robust term $v = 0$ and $\hat{f} = 0$ in the following simulation.

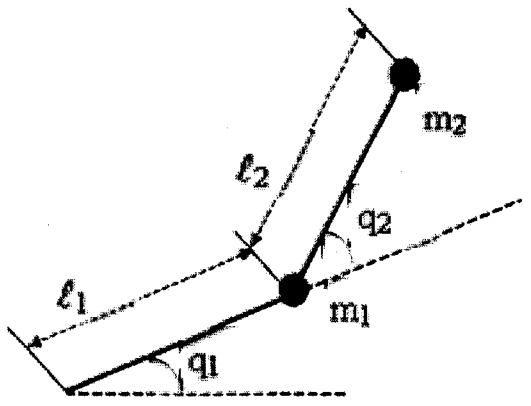

Figure 4.2. Two-link robot manipulator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where q_1, q_2 are the angles of joint 1 and joint 2; m_1, m_2 are the masses of link 1 and link 2, l_1, l_2 are the lengths of link 1 and link 2.

The system parameters are chosen as

$$m_1 = 1.8,\ m_2 = 2.0,\ l_1 = 1.0,\ l_2 = 1.0\ . \tag{4.25}$$

The parameters for the saturation nonlinearity are chosen as

$$\tau_{\max} = 10, \tau_{\min} = -10, m = 1\ . \tag{4.26}$$

In order to show the robustness of the controller, a bounded disturbance is added on the robotic system dynamics as follows:

$$\tau_d = [2\sin t, 2\cos t]^T \tag{4.27}$$

The size of the NN affects the stability, performance, limitation of control efforts, and possible operating conditions. The slow convergence of the tracking error is usually due to the small network size. Moreover, if the network chosen size is too large, the computation burden increases. Common approach is to start with the smaller NN size, and gradually increase the number of hidden layer nodes until satisfactory performance is achieved.

In this example, the NN has eight, forty and two neurons at the input, hidden and output layers, respectively. The first-layer weights V are selected randomly, they are uniformly randomly distributed between -1 and $+1$. These weights represent the stiffness of the sigmoid activation function. The threshold weights for the first layer v_0 are uniformly randomly distributed between -15 and +15. The threshold weights represent the bias in activation functions' positions. The second layer weights W are initialized to zero or any random numbers, and the effect of the inaccurate initialization number can be retrieved by the on-line weights tuning law methodology.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Within the outer loop tracking controller, parameters are chosen so that $K_v = diag\{500,500\}, \Lambda = diag\{10,10\}$. The joint variable is $q = (q, \dot{q})$. The robot arm is initially at rest on the horizontal plane $q_1 = 0, q_2 = 0$, and is commanded into a periodic sinusoidal trajectory $q_{d1} = \sin(t)$, $q_{d2} = \cos(t)$.

The tracking errors for the first and second joints are shown in Figure 4.3 without the saturation compensator considered; the actuator output control signal $\tau(t)$ for the two joints is shown in Figure 4.4.

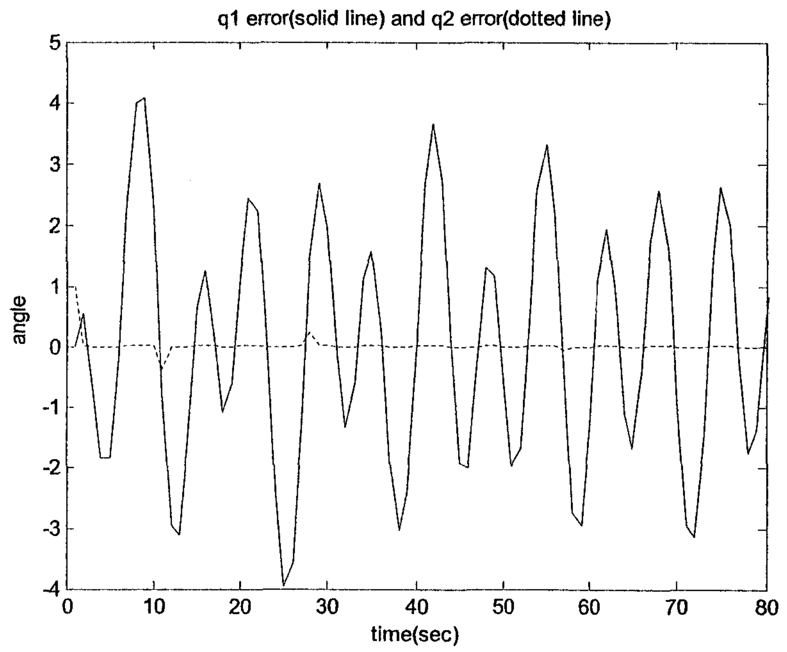

Figure 4.3. Tracking errors for q_1 (solid) and q_2 (dotted) without saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

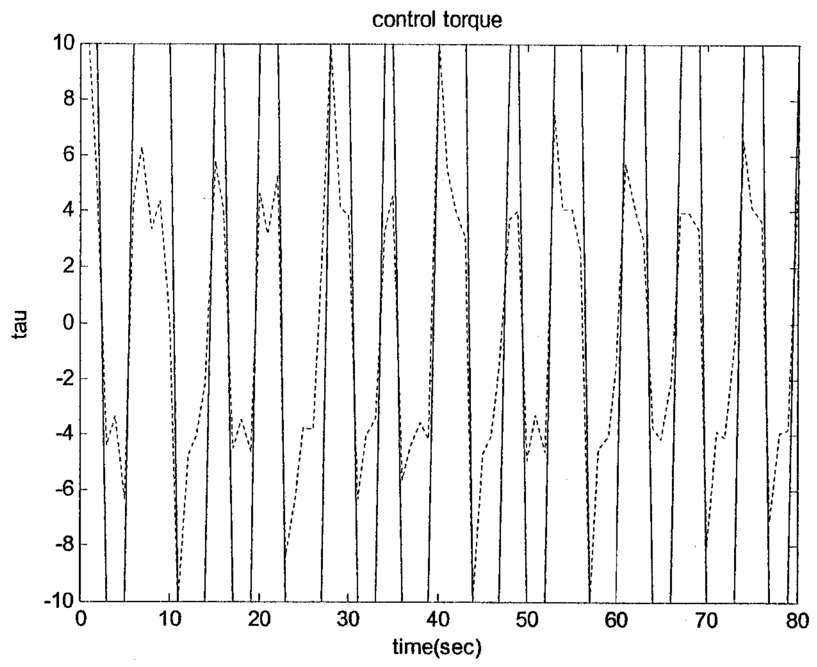

Figure 4.4. Control torque τ_1 (solid) and τ_2 (dotted) without saturation compensator.

The NN saturation compensator weights tuning parameters are chosen as

$$k = 0.002\ ,\quad S = 2\ . \tag{4.28}$$

The tracking errors for the first and second joints are shown in Figure 4.5, control signal $\tau(t)$ in Figure 4.6, and NN output is in Figure 4.7.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

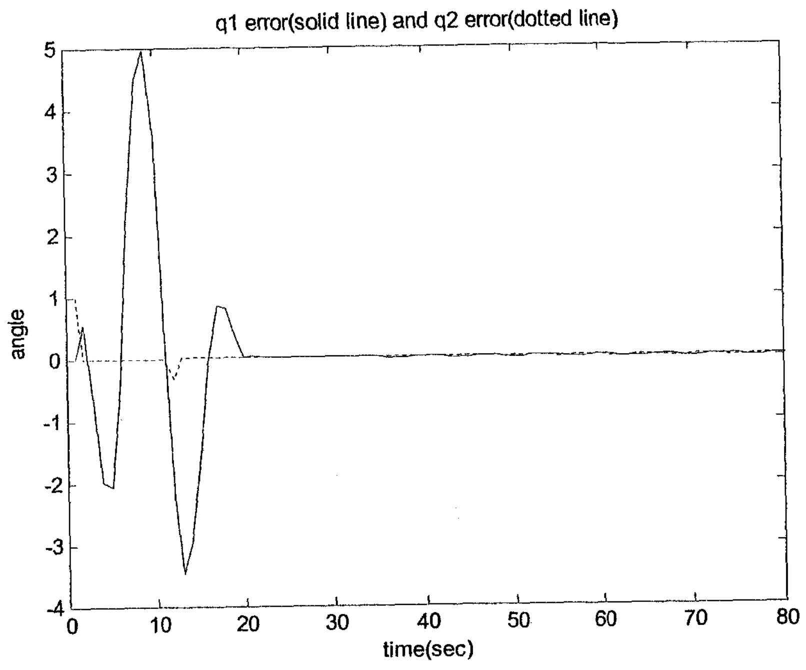

Figure 4.5. Tracking errors for q_1 (solid) and q_2 (dotted) with saturation compensator.

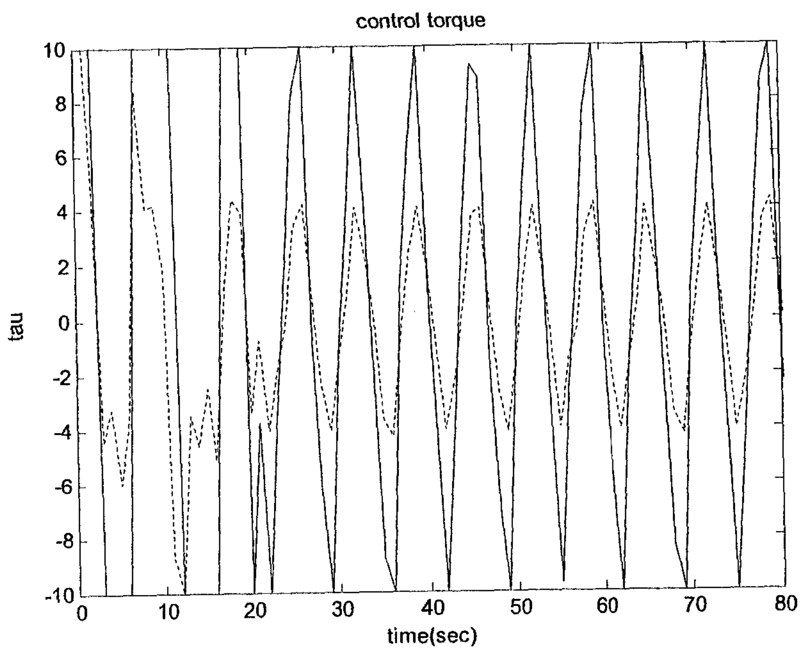

Figure 4.6. Control torque τ_1 (solid) and τ_2 (dotted) with saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

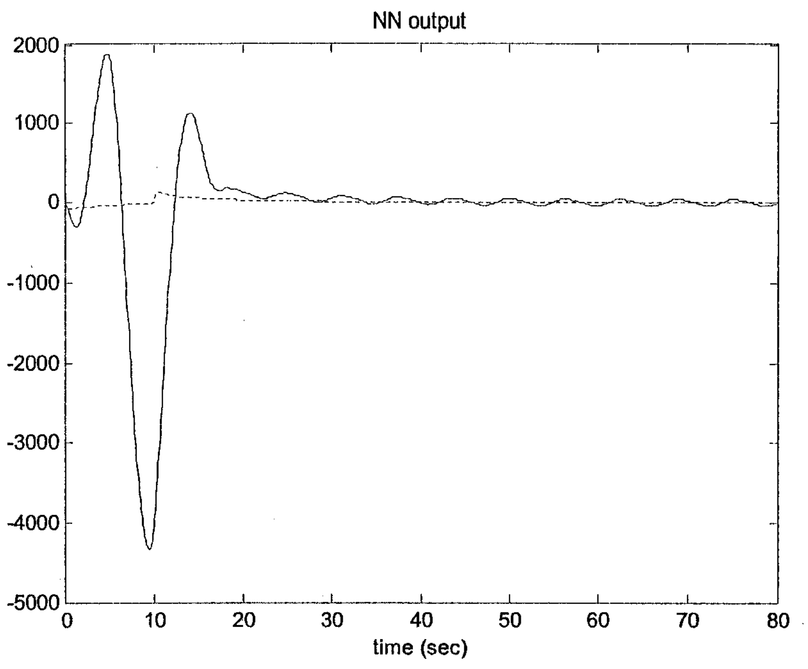

Figure 4.7. NN outputs with saturation compensator.

From the above simulation, it is clear that the proposed scheme can effectively compensate the saturation nonlinearity in a class of nonlinear systems with MIMO, in the presence of disturbance and actuator saturation nonlinearity. The NN saturation compensation scheme designed in Chapter 3 can also be applied and extended for this robotic application

To show the interconnection and the similarity of the control scheme design, it is necessary to derive a suitable state space dynamic equation. The equation (4.1) can be expressed in the form of state equation with the state vector $x = (x_1, x_2)^T = (q, \dot{q})^T$ as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f_r(x) + g_r(x)\tau \\ y &= x_1 = q \end{aligned} \tag{4.29}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

where $f_r(x) = -M^{-1}(x_1)(V_m(x)x_2 + G(x_1) + \tau_d)$ and $g_r(x) = M^{-1}(x_1)$. It is clear that equation (4.1) is feedback linearizable (Nam (1999)), (Lewis, Jagannathan, and Yesildirek (1999)), (Slotine and Li (1991)), (Chen and Khalil (1992)), (Hovakimyan, Nardi and Calise (2001) and (Isidori (1989))

Comparing equation (4.29) to (4.1), function $f_r(x) \in \Re^n$ and $g_r(x) \in \Re^n$ have the corresponding counterparts $f(x) \in \Re$ and $g(x) \in \Re$ respectively. Normally, the inertia matrix $M(x)$ is known, and it is usual to have uncertainty in the Coriolis/ centripetal matrix $V_m(q, \dot{q})$ which is difficult to compute. Thus, it is feasible to assume that function $f_r(x)$ is unknown and function $g_r(x)$ is known, which satisfies *Assumption 3.1*. Note that $f_r(x)$ and $g_r(x)$ in equation (4.29) are vector of dimension n while $f(x)$ and $g(x)$ in equation (3.1) are scalars.

Chapter 4 is self-contained and is an extension of Chapter 3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 5

NN SATURATION COMPENSATOR IMPLEMENTATION USING SIMULINK S-FUNCTIONS

5.1 Simulink S-Functions

Within the last few decades, computer simulations have become an important tool in research and education (Twigg and Johnson (2003)). Various tools provide help to engineers/designers from the research concept, prototyping, development, design, to verification and validation. The common approach in developing a computer model of a dynamic system is to start with a block diagram or flow chart, followed by the block diagram translation, to the source code of a programming language. This practice involves duplication of effort, as the system and controller have to be described twice; in the block diagram with repetition in the programming language (Dabney and Harman (2003)). The development of the faster computers, new software design tools, and the affiliated internet technology has streamlined development cycles in the control system design and implementation (Apkarian (1998)). Simulink has allowed engineers to rapidly and accurately build computer models of dynamical systems and modern control algorithms using block diagram notation.

In (Gao and Selmic (2003)), we proposed a neural network-based saturation

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

compensation is presented for a class of SISO nonlinear systems. The designed intelligent saturation compensator can also apply to other types of control input distortion than saturation as long as the actuator output is a bounded, static function of the current control input .The practical application and extension are made on a specific MIMO system--a second order robotic manipulator. The controller does not require a saturation model to be known. In this chapter, we described how to implement the customized NN saturation compensator using Simulink S-function block diagrams. S-function implementation is done using C MEX S-function (Mathworks (2002)) and MATLAB as simulation tools. The advantage of Simulink implementation is rapid development, prototyping, and potential of real-time code generation using the Real-Time Workshop Toolbox (Mathworks (2002)). This implementation is a crucial step in any real-time saturation compensation implementation. In this dissertation, we show how custom-made NN control system design can be efficiently implemented in the real-time environment on microcontroller.

5.2 Simulink Model of NN Saturation Compensator

Simulink is a software package for modeling, simulation, and analyzes of dynamical systems. It includes a wide range of predefined blocks and symbols, and various standard libraries for different applications. Simulink models consist of inputs, outputs, states, and system functions (output function, update, and derivative function) specifying the time-dependent relationships between the inputs, states and the outputs. Simulink also supports the subsystem hierarchical structure (Mathworks (2002)). It is a convenient simulation tool that simplifies complex system development and design. For

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

linear systems, Simulink is equipped with transfer function block and state-space block. But to build the system function block for equation (3.1) and (4.1), the general nonlinear system dynamics, or the application-customized NN weight dynamics for saturation compensator, no standard block can be utilized. One has to develop a customized S-function (system-function) (Mathworks (2002)), and to create the custom block for these dynamic systems

S-function provides a powerful mechanism to extend the capability of Simulink. It allows user to add new general purpose blocks in Simulink and incorporate the existing C code into it. S-function is a computer language description of Simulink block, which can be written in C, C++, MATLAB and compiled using MEX-files mex utility. S-function can also be used with Real-Time Workshop (Mathworks (2002)) to provide real-time code that can be implemented on the microcontrollers.

Given the NN saturation compensation scheme for the nonlinear systems in Figure 3.1 and Figure 4.1, the equivalent Simulink block model of the NN saturation compensator is shown in the Figure 5.1. It consists of two subsystems: reference command subsystem to specify the desired system states $x_d(t)$, namely $x_d(t)=[y_d, \dot{y}_d, \cdots, y_d^{(n-1)}]^T$, and PD controller and NN compensator subsystem. The nonlinear system model block, PD controller, and NN compensator block are written and implemented using C MEX S-function mechanism (Mathworks (2002)).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

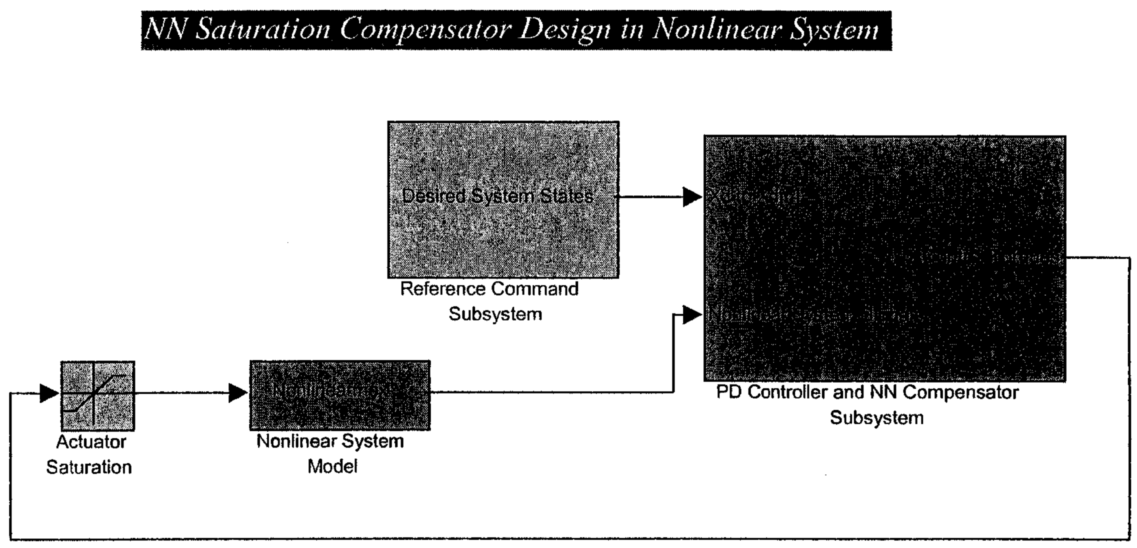
NN Saturation Compensator Design in Nonlinear System
Desired System States
Reference Command Subsystem
PD Controller and NN Compensator Subsystem
Actuator Saturation
Nonlinear System Model

Figure 5.1. NN saturation compensator hierarchical model.

A Simulink block as shown in Figure 5.2 consists of a set of inputs, states and outputs, where the output is a function of the sample time, the inputs and the block states.

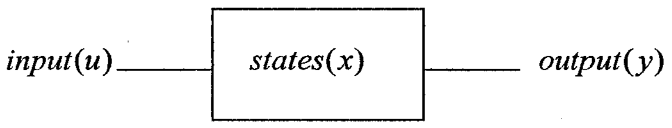

Figure 5.2. Components of Simulink block.

Considering continuous states only, the mathematical expression among the components of the Simulink block is as follows,

$$\begin{aligned} y &= f_o(t,x,u) \quad (Output) \\ \dot{x}_c &= f_c(t,x,u) \quad (Derivative) \end{aligned}, \tag{5.1}$$

and the equation (3.1) is given by,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\begin{aligned} y &= x_1 && (Output) \\ x_1^{(n)} &= f(x) + g(x)\tau && (Derivative) \end{aligned}, \qquad (5.2)$$

where the control torque τ is the input to the block, $x = [x_1, x_2, \cdots, x_n]^T$ are the states, and the output is x_1.

With respect to the NN weights tuning dynamics, the Simulink block are composed of the input to the NN compensation block (x_{NN}), the states (the weight matrix $\hat{W}$), and the output of the block (the output of the NN $\hat{\varphi} = \hat{W}^T \sigma(V^T x_{NN})$).

S-function simulates a general model and represents the mathematical relationship of the block components for equations such as weight tuning dynamics equation and equation (5.2). In Figure 5.1, top-level hierarchical block contains NN saturation compensator and PD controller sub-blocks.

Execution of Simulink S-function model proceeds in stages (Mathworks (2002)). The initialization stage determines the S-function block's characteristics (the number of inputs, outputs, states, the sample time, etc.), which follows a simulation loop, with sequential Simulink steps. During each step, Simulink invokes functions to calculate the present state, the next state, derivative and outputs for the current sample time. The following table lists the contents of the C-MEX file S-function used in simulation examples in this dissertation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Table 5.1. The contents of C-MEX S-function.

Simulation Stage	**S-Function Routine**
Initialize size	mdlInitializeSizes
Initialize sample time	mdlInitializeSampleTimes
Initialize conditions	mdlInitializeConditions
Calculate output	mdlOutputs
Calculate derivative	mdlDerivatives
End simulation	mdlTerminate

We present two Simulink simulation examples involving saturation compensation.

5.2.1 Nonlinear System of "Pendulum Type"

In the program running, the NN has four, ten, and one neurons at the input, hidden and output layers, respectively. The first-layer weights V , NN weight tuning papramenter are selected as in Chaper 3.

Two S-functions, Pendulummex which is shown and labeled pendulum system model (Figure 5.3) and NN_Pendulummex_Sat which is shown and labeled NN saturation compensator (Figure 5.5), were written in C programming language and compiled via the MATLAB MEX-file utility.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

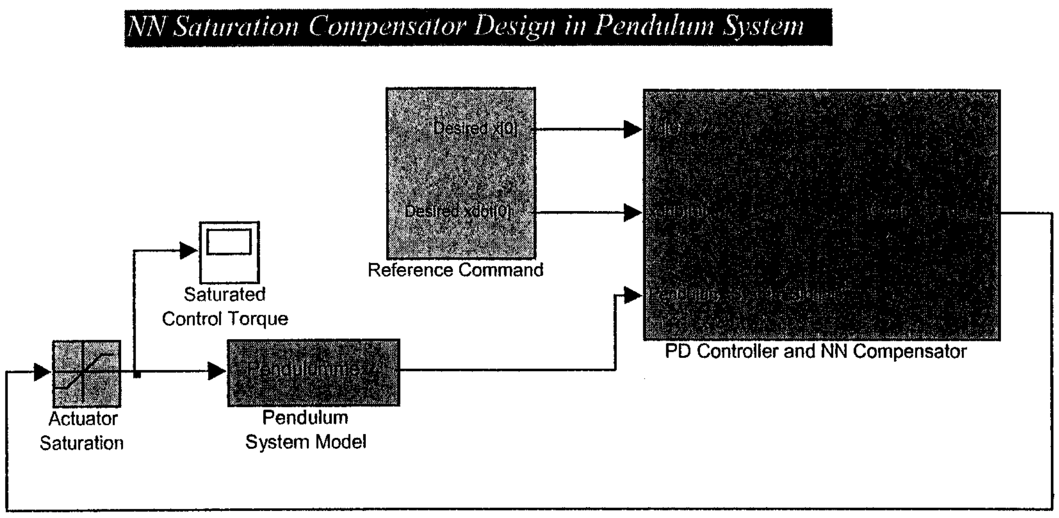

Figure 5.3. NN saturation compensator hierarchical model for pendulum system.

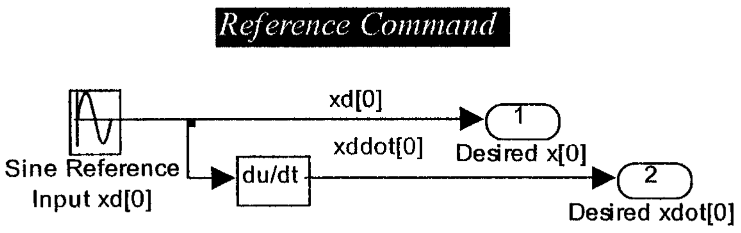

Figure 5.4. Reference command subsystem.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

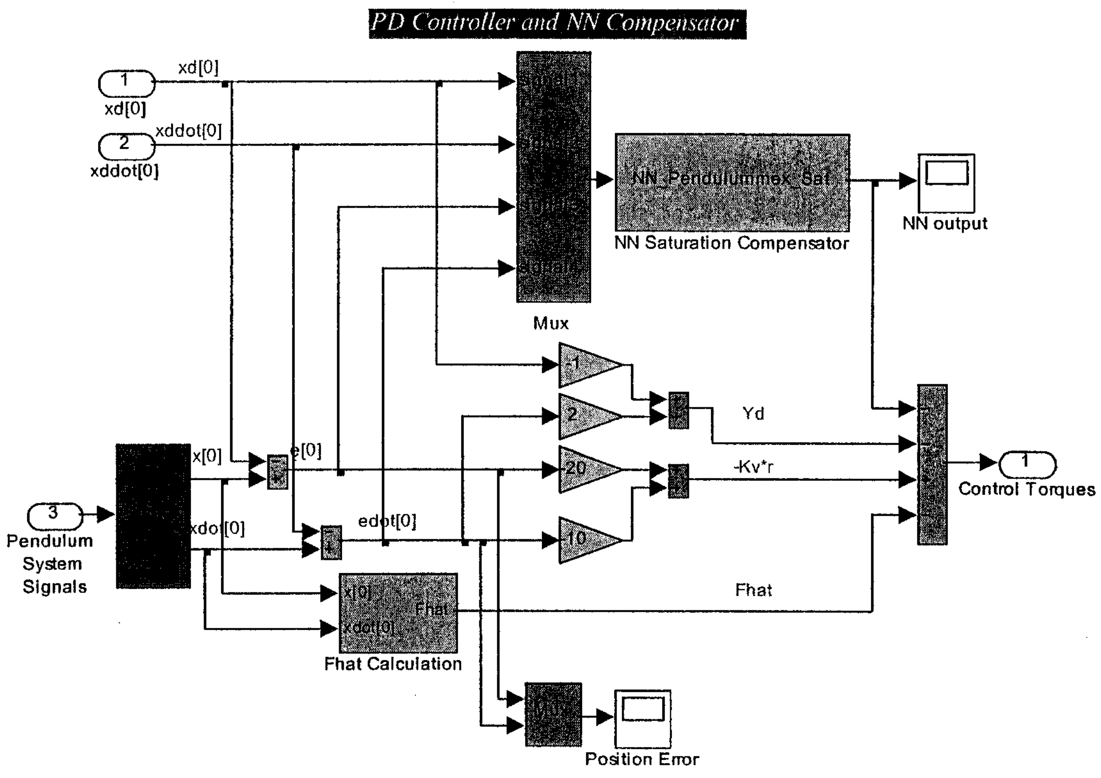

Figure 5.5. PD controller and NN compensator subsystem

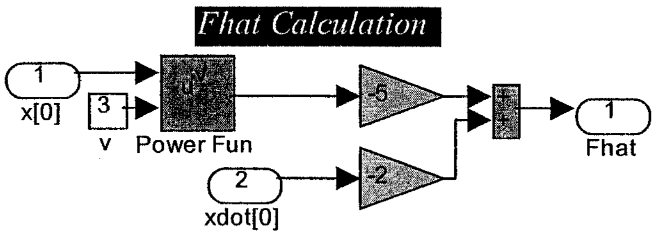

Figure 5.6. Fhat calculation sub-block

After the Simulink block running, the simulation result such as the tracking errors and control signal $\tau(t)$ are turned out as same as the Figures shown in Chapter 3.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

5.2.2 Two-Link Robot Arm

In the program running, the NN parameter such as input layer, hidden function, and output layers, the first-layer weights V, as well as NN weight tuning papramete are selected as in Chaper 4.

Two S-functions, Robotmex which is shown and labeled pendulum system model in Figure 5.7 and NN_Robotmex_Sat shown and labeled NN saturation compensator in Figure 5.9, were written in C programming language and compiled via the MATLAB MEX-file utility.

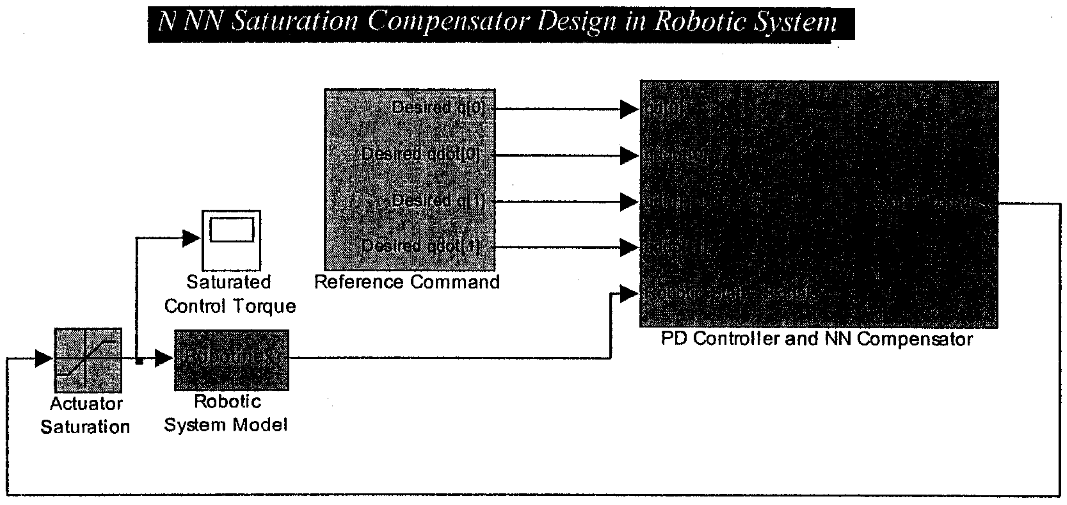

Figure 5.7. NN saturation compensator hierarchical model for robotic system.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

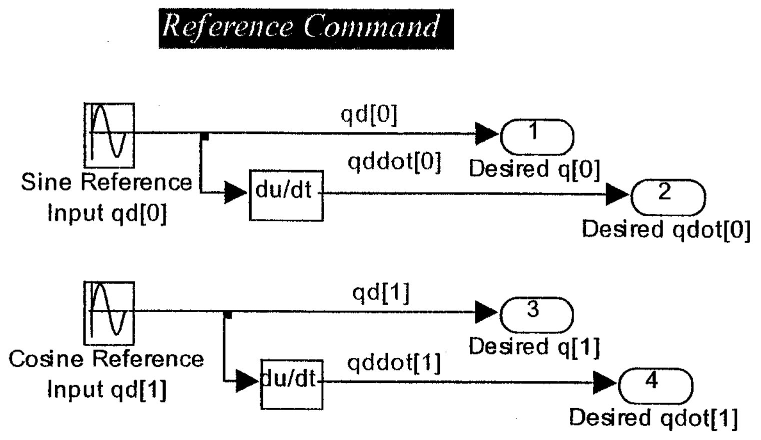

Figure 5.8. Reference command subsystem.

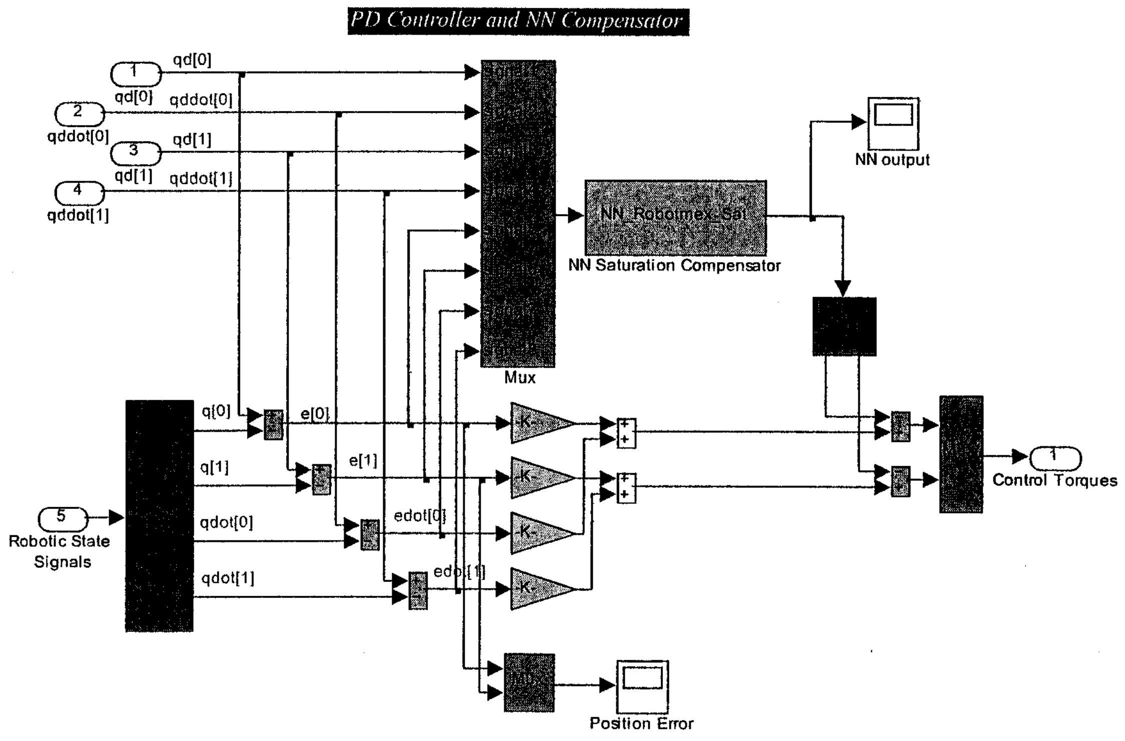

Figure 5.9. PD controller and NN compensator subsystem.

After the Simulink block running, the simulation result such as the first and second joints tracking errors and control signal $\tau(t)$ are turned out as same as the figures shown in Chapter 4.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Simulink-based S-function blocks are developed that are equivalent to theoretical NN saturation compensation schemes. Such control system platform can be used for real-time code generation using Real-Time Workshop Toolbox. This chapter shows how the custom-made NN control system design can easily be implemented in the real-time environment on microcontroller.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

the standard dynamics model of a flexible link manipulator may be defined using the

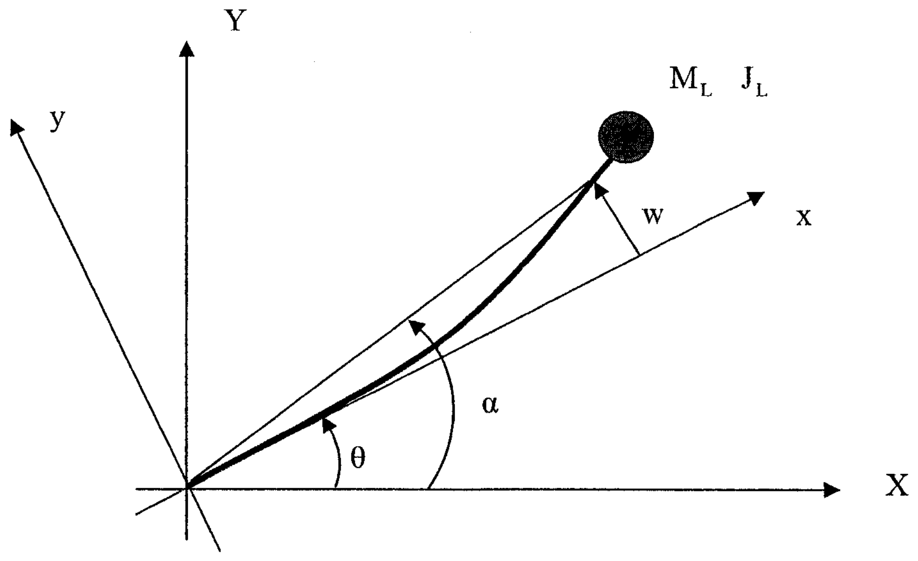

Figure 6.1. A flexible one-link robot arm.

recursive Lagrangian approach (Lewis, Jagannathan, and Yesildirek: (1999)), (Talebi *et al.* (2002)),

$$M(q)\ddot{q} + D(q,\ddot{q})\dot{q} + Kq + F(q,\dot{q}) + G(q) = B(q)\tau \,, \tag{6.4}$$

where $q = [q_{r,} q_f]^T$, $q_r \in \Re^{n_r}$ is the vector of rigid modes (generalized joint coordinates), $q_f \in \Re^{\infty}$ is the vector of infinitely many flexible modes, $M(q) = M(q_r, q_f)$ is the inertia matrix, $D(q,\dot{q}) = D(q_r, q_f, \dot{q}_r, \dot{q}_f)$ is the coriolis/centripetal matrix containing rigid and flexible modes, K is the stiffness matrix, $F(q,\dot{q}) = (q_r, q_f, \dot{q}_r, \dot{q}_f)$ is the friction matrix, $G(q) = G(q_r, q_f)$ is the gravity vector, $B(q) = B(q_r, q_f)$ is the input matrix. The input actuator control torque applied to each joint is vector τ. All of the abovementioned matrices have compatible, yet practically infinite dimensions (Zhu *et al.* (1994)). In this dissertation, we follow the finite dimensional approach that many researchers (Gutierrez,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Lewis and Lowe (1998)), (Lewis, Jagannathan, and Yesildirek (1999)), (Sun *et al.* (2003)), (Siciliano and Book (1988)), (Siciliano, Prasad, and Calise (1992)), (Moallem, Khorasani, and Patel (1997)), (Saber (2000)), (Talebi *et al.* (2002)) adopted along with the model truncation to obtain the system dynamics. Equation (6.4) is characterized as the underactuated system, which has more degrees of freedom than the dimensions of the control torque τ.

6.2 Decomposition of Flexible Link Dynamics

The singular perturbation approach consists of decomposing the system dynamics into two time scale subsystems. In this dissertation, slow dynamics and fast dynamics corresponds to the rigid modes q_r and flexible modes q_f respectively. To apply singular perturbation as in (Siciliano and Book (1988)), (Lewis, Jagannathan, and Yesildirek (1999)), equation (6.4) is converted to

$$\begin{aligned}\ddot{q}_r &= -D_{rr}^{l}\dot{q}_r - D_{rf}^{l}\dot{q}_f - K_{rf}^{l}q_f - F_r^{l} - G_r^{l} + B_r^{l}\tau \\ \ddot{q}_f &= -D_{fr}^{l}\dot{q}_r - D_{ff}^{l}\dot{q}_f - K_{ff}^{l}q_f - F_f^{l} - G_f^{l} + B_f^{l}\tau\end{aligned} \tag{6.5}$$

According to (Lewis, Jagannathan, and Yesildirek (1999)), the slow subsystem of the flexible link manipulator is given by

$$\overline{M}_{rr}\ddot{\overline{q}}_r + \overline{D}_{rr}\dot{\overline{q}}_r + \overline{F}_r + \overline{G}_r = \overline{\tau}, \tag{6.6}$$

where the bar over the variables denotes the slow part of them and it assumes that the stiffness of the flexible links is sufficiently large, $\overline{\tau}$ is the control input of the slow system, and $\overline{M}_{rr}$, $\overline{D}_{rr}$, $\overline{F}_r$, $\overline{G}$ are defined as in (Lewis, Jagannathan, and Yesildirek (1999)).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

To define the fast subsystem, let us introduce scale factor ε, where $1/\varepsilon^2$ is the smallest stiffness in K^1_{ff}, and define $\varepsilon^2\xi = q_f$, $\widetilde{K}_{ff} = \varepsilon^2 K^1_{ff}$, $\zeta_1 = \xi - \bar{\xi}$, $\zeta_2 = \dot{\xi}$, where $\bar{\xi} = \widetilde{K}^{-1}_{ff}\overline{H}^{-1}_{ff}(-\overline{D}^1_{fr}\dot{\bar{q}}_r - \overline{F}^1_f - \overline{G}^1_f + \overline{B}^1_f\bar{\tau})$ (Lewis, Jagannathan, and Yesildirek (1999)), and a fast time scale $\eta = t/\varepsilon$, resulting in

$$\frac{d}{d\eta}\begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix} = \begin{bmatrix}0 & I\\ -\overline{H}_{ff}(\bar{q}_r,0)\widetilde{K}_{ff} & 0\end{bmatrix}\begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix} + \begin{bmatrix}0\\ \overline{B}^1_f(\bar{q}_r,0)\end{bmatrix}\tau_F, \tag{6.7}$$

or

$$\frac{d\zeta}{d\eta} = A_F\zeta + B_F\tau_F, \tag{6.8}$$

with $\zeta = [\zeta_1^T \quad \zeta_2^T]^{\mathrm{T}}$.

The following remarks are relevant:

The slow subsystem equation (6.6) and fast subsystem (6.7) are reduced control effective by order of ε from original system equation (6.4), that is

$$\begin{aligned} q_r &= \bar{q}_r + O(\varepsilon)\\ q_f &= \varepsilon^2(\bar{\xi} + \zeta_1) + O(\varepsilon)\end{aligned} \tag{6.9}$$

The chosen modified joint-based tracking output control $y = \bar{q}_r$ is in contrast to the reflected-tip control $y_{ri} = q_{ri} - (w_i(x_i,t)/x_i)$ (Talebi *et al.* (2002)). Both of them have the advantage over joint-based control that can make the vibration of the flexible mode controllable.

Assumption of that (A_F, B_F) is stabilizable is reasonable, since the fast system (6.8) parameters have bounded uncertainty and perturbation because of slow system variable, which vary smoothly with time (Lewis, Jagannathan, and Yesildirek (1999)).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The model (6.6) follows the same properties of rigid link robot. Namely, $\overline{M}_{rr}(q_r)$ is a positive definite symmetric matrix and upper and lower bounded, $\overline{D}_{rr}(q_r,\dot{q}_r)$ is bounded by $d_m(q_r)\|\dot{q}_r\|$ with $d_m(q_r)$ a known value, and $\dot{\overline{M}} - 2\overline{D}_{rr}$ is skew-symmetric.

For flexible one link, q_r is a scalar, q_f is a vector, and the number of the vector depends on number of flexible modes.

For flexible one link, the composite control $\tau = \bar{\tau} + \tau_F$ is a scalar as well as $\bar{\tau}$ and τ_F.

We consider composite control τ of flexible one link robot manipulator in the presence of actuator saturation nonlinearity. Actuators control torque is subject to saturation limits and this elevates the complexity of the controller for flexible link robot arm.

6.3 Composite Control Subject to Saturation Constraint

Based on singular perturbation and due to the extension of Tikhonov's Theorem (Lewis, Jagannathan, and Yesildirek (1999)), (Siciliano and Book (1988)), the slow and fast controls are time domain separate and essentially independent. It is reasonable to design slow control $\bar{u}$ for the slow subsystem (6.6), and to design fast control u_F for the fast subsystem (6.7). Two control designs can be initiated to obtain the independent control input components and simply add together to produce the composite control u. When the saturation is imposed on the control input u, equivalently it affects the two independent control components $\bar{u}$ and u_F. To compensate the saturation of composite control u, saturation compensation should consider both the slow and fast control

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

signals, and it means to develop two independent saturation compensator components for the slow and fast subsystem respectively.

According to saturation model (1.2), the composite control subject to saturation constraint is given by

$$\tau = \bar{u} + \bar{\delta} + u_F + \delta_F , \tag{6.10}$$

where $\bar{\tau} = sat(\bar{u})$, $\delta = \bar{\tau} - \bar{u}$, $\tau_F = sat(u_F)$, $\delta_F = \tau_F - u_F$.

Based on Lyapunov theory two saturation sub-compensators are designed to accommodate $\bar{\delta}$, and δ_F. Figure 6.2 shows the overall control diagram of the flexible link system with NN saturation compensator consisting of two neural nets, namely NN$_R$ and NN$_F$ for rigid and fast subsystems.

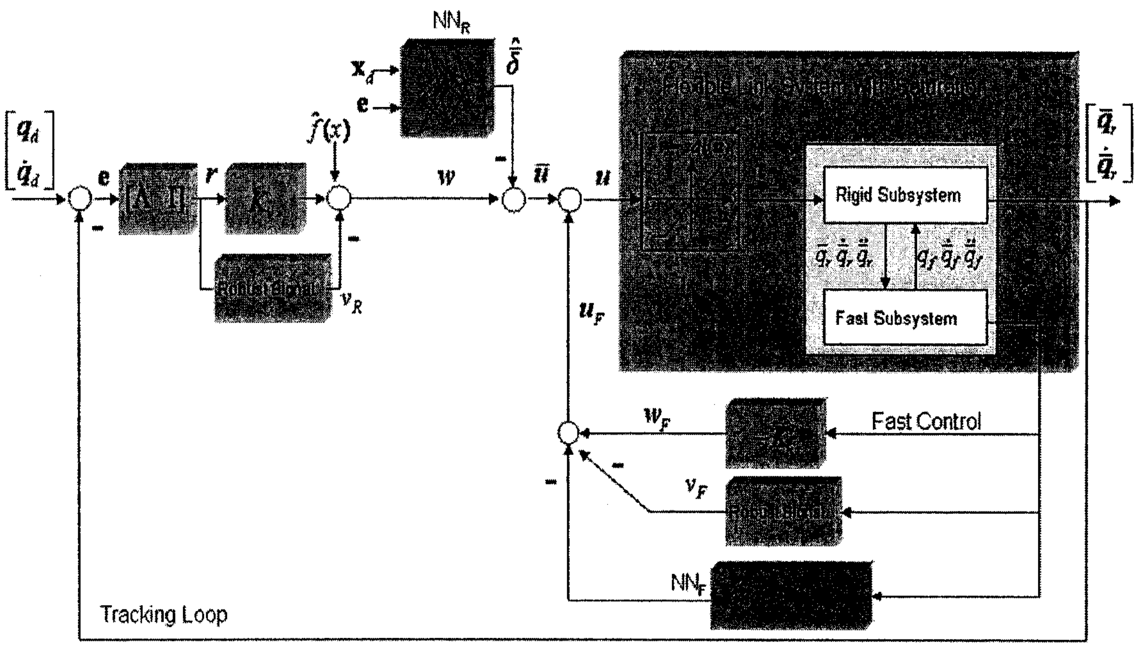
NN$_R$
x_d
e
$\hat{f}(x)$
$\hat{\bar{\delta}}$
$\begin{bmatrix} q_d \\ \dot{q}_d \end{bmatrix}$
e
r
w
$\bar{u}$
u
v_R
u_F
Rigid Subsystem
Fast Subsystem
$\bar{q}_r \dot{\bar{q}}_r \ddot{\bar{q}}_r$
$q_f \dot{q}_f \ddot{q}_f$
$\begin{bmatrix} \bar{q}_r \\ \dot{\bar{q}}_r \end{bmatrix}$
w_F
Fast Control
v_F
NN$_F$
Tracking Loop

Figure 6.2. Flexible link system with NN saturation compensator.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.4 NN-based Saturation Compensation for Rigid Dynamics

Rigid dynamics (6.6) is exactly the Lagrange form of n-link rigid robot arm equation. Thus, any existing adaptive or NN-based control and identification techniques can be applied here (Gao and Selmic (2003)), (Lewis, Jagannathan, and Yesildirek: (1999)). Subject to constraint on the slow control, equation (6.6) can be rewritten as

$$\overline{M}_{rr}\ddot{\overline{q}}_r + \overline{D}_{rr}\dot{\overline{q}}_r + \overline{F}_r + \overline{G}_r = \overline{u} + \overline{\delta} \ . \tag{6.11}$$

Given a desired arm trajectory $q_d(t)$, define tracking error as

$$e(t) = q_d(t) - \overline{q}_r(t) \tag{6.12}$$

Assumption 6.1 (Bounded Desired Trajectory): The desired trajectory $q_d(t)$ is bounded and continuous, and $\|q_d(t)\| \le Q$ with Q known scalar bound.

Then, the filtered tracking error is given as

$$r = \dot{e} + \Lambda e, \tag{6.13}$$

where $\Lambda = \Lambda^T > 0$ is a design parameter scalar. At the same time, the control goal is to guarantee the stability of the filter tracking error $r(t)$ (Slotine and Li (1991)). differentiating r in equation (6.13) and invoking (6.11) yields the rigid link dynamics as

$$\overline{M}_{rr}\dot{r} = -\overline{D}_{rr}r - \overline{u} - \overline{\delta} + f(x), \tag{6.14}$$

where the nonlinear rigid robot function is

$$f(x) = \overline{M}_{rr}(\overline{q}_r)(\ddot{q}_d + \Lambda\dot{e}) + \overline{D}_{rr}(\overline{q}_r, \dot{\overline{q}}_r)(\dot{q}_d + \Lambda e) + \overline{G}_r(\dot{\overline{q}}_r) + \overline{F}_r(\overline{q}_r)\,. \tag{6.15}$$

Vector x contains all the time signals needed to compute $f(x)$, and may be defined for instance as $x \equiv [q_d, \dot{q}_d, \ddot{q}_d, e, \dot{e}]^T$. It is noted that the function $f(x)$ contains all the potentially unknown functions in (6.11).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Choose the tracking controller as in (Gao and Selmic (2004)).

$$w = \hat{f} - v_R + K_v r, \tag{6.16}$$

where $\hat{f}$ is the fixed approximation of function $f(x)$. The functional estimation error is given by $\tilde{f} = f - \hat{f}$, and the robust term corresponding to the rigid subsystem is given by v_R.

Approximation $\hat{f}$ is fixed in this dissertation and will not be adapted. Robust term v_R is chosen for the disturbance rejection. The control scheme for robot manipulators under actuator saturation consists of a standard tracking controller and an NN saturation compensator.

NN *universal approximation property* defines that any continuous function can be approximated arbitrarily well using a linear combination of sigmoidal functions, namely,

$$f(x) = W^T \sigma(V^T x) + \varepsilon(x), \tag{6.17}$$

where the $\varepsilon(x)$ is the NN approximation error. The reconstruction error is bounded on a compact set S by $\|\varepsilon(x)\| < \varepsilon_N$. Moreover, for any ε_N one can find a NN such that $\|\varepsilon(x)\| < \varepsilon_N$ for all $x \in S$.

The first layer weights V (including thresholds) are selected randomly and will not be tuned. The second layer weights W are tunable. The approximating weights W are ideal target weights, and it is assumed that they are bounded so that $\|W\|_F \le W_M$.

Applying NN universal approximation property, there exist NN with some ideal feed-forward NN weights V_R and W_R corresponding to the rigid subsystem to closely approximate the unknown modified saturation function $\bar{\delta}$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

$$\bar{\delta} = W_R^T \sigma(V_R^T x_{NN}) + \varepsilon \ . \tag{6.18}$$

Saturation control is given by

$$\bar{u} = w - \hat{\bar{\delta}} \ , \tag{6.19}$$

where $\hat{\bar{\delta}}$ is the actual realization of the NN rigid subsystem, denoted as NN_R in Figure 6.2, compensation function given by

$$\hat{\bar{\delta}} = \hat{W}_R^T \sigma(V_R^T x_{NN}) \ , \tag{6.20}$$

where the NN weights approximation error is $\tilde{W}_R = W_R - \hat{W}_R$, and the NN input is selected as $x_{NN} \equiv [x_d, \dot{x}_d, e, \dot{e}]^T$.

Assumption 6.2 (Bounded Ideal NN Weights): The ideal NN weights W are bounded so that $\|W\| \le W_M$, with W_M known bounds.

Assumption 6.3 (Bounded Estimation Error): The estimate $\hat{f}(x)$ of a nonlinear unknown function $f(x)$ is assumed known, so that the functional estimation error $\tilde{f}(x)$ satisfies $\left|\tilde{f}(x)\right| \le f_M(x)$ for some known function bounds $f_M(x)$ (Isidori (1989)), (Narendra (1991)).

Substituting (6.18) and (6.19) into (6.14) gives the closed-loop error dynamics

$$\overline{M}_{rr}\dot{r} = -\overline{D}_{rr}r + \tilde{f}(x) - K_v r + v_R - \tilde{W}_R^T \sigma(V_R^T x_{NN}) - \varepsilon \ . \tag{6.21}$$

The following theorem specifies the robust term and the stable NN weights tuning rules of controller, such that the closed-loop system is uniformly ultimately bounded (UUB) in the presence of the actuator saturation in rigid link manipulators.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

***Theorem 6.1 (Tuning of NN Rigid Subsystem Compensator)*:** Given the rigid robot arm dynamics (6.21), Assumptions 6.1, 6.2, and 6.3, choose the tracking control law (6.16), and the saturation compensator (6.19), (6.20). Choose the robustifying term as

$$v_R(t) = -f_M(x)sign(r), \tag{6.22}$$

where the $f_M(x)$ is the bound on functional estimation error. Let the estimated NN weights be provided by the NN tuning algorithm

$$\dot{\hat{W}}_R = -S\sigma(V_R^T x_{NN})r^T - kS\|r\|\hat{W}_R \tag{6.23}$$

where $S = S^T > 0$ is any constant matrix representing the learning rates of the NN, and k is a small scalar positive design parameter. By properly selecting the control gains and the design parameters, the filtered error $r(t)$ and the NN weights $\hat{W}_R$ are UUB (Uniformly Ultimately Bounded).

Here, we skip the proof part of above theorem. Regarding the candidate of the lyapunov function, and how the robust term and weight tuning law is derived, interested reader can refer to Chapter 3. Also the detail proof of the Theorem 6.1 that uses the Lyapunov stability analysis is given in (Gao and Selmic (2003)) as well as the explicit bounds for $\|r\|$ and $\|\tilde{W}_R\|$.

During operational phase, the first layer weights V_R are selected randomly and will not be tuned. The second layer weights W_R are on-line tunable based on equation (6.23) and can be initialized to be zero. The robustifying term $v_R(t)$ is needed to for disturbance rejection and to keep the closed-loop system stable during the initial NN learning phase.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.5 Robust Saturation Compensator for Fast Dynamics

As shown in (6.7), the fast subsystem is parameterized by the slow variable $\bar{q}_r$. From equation (6.12), replacing $\bar{q}_r$ by $e+q_d$ in the fast dynamics (6.7) results in

$$\frac{d}{d\eta}\begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix}=\begin{bmatrix}0 & I\\ -\overline{H}_{ff}(e)\tilde{K}_{ff} & 0\end{bmatrix}\begin{bmatrix}\zeta_1\\ \zeta_2\end{bmatrix}+\begin{bmatrix}0\\ \overline{B}_f^1(e)\end{bmatrix}\tau_F. \tag{6.24}$$

As discussed in Section 6.3, the goal of the slow control is to make the filter error r arbitrary small, or equivalently to make the tracking error e arbitrary small. For instance, optimal control can be used to design τ_F such that the internal dynamics is stable (Lewis, Jagannathan, and Yesildirek (1999)).

Subject to constraint on the fast control, equation (6.8) can be rewritten as

$$\frac{d\zeta}{d\eta}=A_F\zeta+B_F(u_F+\delta_F), \tag{6.25}$$

with $\zeta=[\zeta_1^T \quad \zeta_2^T]^T$. Let us define the desired state vector as

$$\zeta_d=[G_d^T,\dot{G}_d^T]^T. \tag{6.26}$$

Assumption 6.4 (Bounded Desired Trajectory): The desired trajectory $\zeta_d(t)$ is continuous, available and bounded, and $\|\zeta_d(t)\|\le E$ with E known scalar bound.

Assumption 6.5: The state $\zeta(t)$ is completely measurable and available for controller design.

The output of (6.25) is choosen as

$$G=C_f\zeta, \tag{6.27}$$

where

$$C_f=[C_{f1}(\bar{q}_r),0], \tag{6.28}$$

and in (6.28) $C_{f1}(\bar{q}_r)$ is noramly selected as a constant.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In this section, a NN is used to estimate the unknown nonlinear saturation function δ_F appearing in (6.25). The NN$_F$ is a two-layer net, where $\hat{W}_F^T$ is the estimate of the second layer weights, σ is the basis activation function, $\zeta_{NN}=[\zeta_1,\zeta_2,\tilde{\zeta}_1,\tilde{\zeta}_2]^T$ is the input vector of the NN, which should include the full state information of the unknown system. There exists a NN with some ideal weights V_F^T and W_F^T that closely approximates the unknown function δ_F

$$\delta_F = W_F^T\sigma(V_F^T\zeta_{NN})+\varepsilon . \tag{6.29}$$

Choose the fast control law

$$u_F = -K_F\zeta - \hat{W}_F\sigma(V_F\zeta_{NN}) - v_F , \tag{6.30}$$

where robustifying term v_F is added to provide the robustness for the NN reconstruction error ε (Huang and Lewis (2003)) and the unmodeled dynamics, and K_F is the pole placement gain selected using Ackermann's formula. Substituting into (6.25) results in

$$\frac{d\zeta}{d\eta} = A_S\zeta + B_F\tilde{W}_F\sigma(V_F\zeta_{NN}) + B_F\varepsilon_F - B_Fv_F . \tag{6.31}$$

where A_S is a stable matrix satisfying

$$A_S = A_F - B_FK_F , \tag{6.32}$$

$$\tilde{W}_F = W_F - \hat{W}_F . \tag{6.33}$$

People normally adopt LQR or other optimal control law to reliaze the fast control torque to cause the fast subsystem stability. The control input $w_F=-K_F\zeta$ is chosen to stabilize the unstable fast subsystem (6.7) or (6.8), while the *NN$_F$* is used to compensate the fast subsystem for the saturation nonlinearity effect. The NN$_F$ prevents the control

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

signal from being saturated and ensures the system stability that would otherwise be hard to rigorously prove.

***Theorem 6.2 (Tuning of NN Fast Subsystem Estimator)*:** Given the fast subsystem dynamics (6.31), Assumptions 6.2, 6.4, and 6.5, choose the robust term as

$$v_F = sign(B_F^T \zeta)\varepsilon_M, \tag{6.34}$$

where the ε_M is the bound on NN functional estimation error. Let the estimated NN$_F$ weights be provided by the NN tuning algorithm

$$\dot{\hat{W}}_F = \Gamma\sigma(V_F^T \zeta_{NN})\zeta^T B_F - h\Gamma\|\zeta\|\hat{W}_F \tag{6.35}$$

where $\Gamma = \Gamma^T > 0$ is a constant matrix representing the learning rates of the NN, and h is a small scalar positive design parameter. By properly selecting the control gains and the design parameters, the fast system states ζ and the NN weights $\hat{W}_F$ are UUB.

Proof: Choose the Lyapunov function candidate as

$$L = \frac{1}{2}\zeta^T\zeta + \frac{1}{2}tr(\tilde{W}_F^T\Gamma^{-1}\tilde{W}_F) \tag{6.36}$$

Differentiating yields

$$\dot{L} = \zeta^T\dot{\zeta} + tr(\tilde{W}_F^T\Gamma^{-1}\dot{\tilde{W}}_F) \tag{6.37}$$

Whence substitution from dynamics equation (6.31) yields

$$\dot{L} = \zeta^T A_S \zeta + \zeta^T B_F(\varepsilon_F - v_F) + tr\left\{\tilde{W}_F^T[\Gamma^{-1}\dot{\tilde{W}}_F + \sigma(V_F^T\zeta_{NN})\zeta^T B_F]\right\} \tag{6.38}$$

Applying the NN tuning rules, selected Lyapunov function is simplified to

$$\dot{L} = \zeta^T A_S \zeta + \zeta^T B_F(\varepsilon_F - \frac{B_F^T\zeta}{\left|B_F^T\zeta\right|}\varepsilon_M) + tr\left\{\tilde{W}_F^T\hat{W}_F\right\}h\|\zeta\| \ . \tag{6.39}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Let $A_{S\min}$ be the smallest eigenvalue of A_S,

$$\dot{L} \le -A_{S\min}\|\zeta\|^2 + \zeta^T B_F \varepsilon_F - \frac{\|B_F\|^2\|\zeta\|^2}{|B_F^T\zeta|}\varepsilon_M + tr\{\tilde{W}_F^T \hat{W}_F\}h\|\zeta\| \tag{6.40}$$

Using the inequality,

$$tr\left[\tilde{X}^T(X-\tilde{X})\right] \le \|\tilde{X}\|_F\|X\|_F - \|\tilde{X}\|_F^2, \tag{6.41}$$

the inequality (6.40) can be written as

$$\dot{L} \le -A_{S\min}\|\zeta\|^2 + \zeta^T B_F \varepsilon_F - \frac{\|B_F\|^2\|\zeta\|^2}{|B_F^T\zeta|}\varepsilon_M + (\|\tilde{W}_F\|_F\|W_F\|_F - \|\tilde{W}_F\|_F^2)h\|\zeta\|$$

$$\le -\|\zeta\|\left\{A_{S\min}\|\zeta\| + h(\|\tilde{W}_F\|_F - \frac{W_M}{2})^2 - \frac{h}{4}W_M^2\right\} + \zeta^T B_F \varepsilon_F - \|B_F\|\|\zeta\|\varepsilon_M \tag{6.42}$$

which is guaranteed to remain negative as long as

$$\|\zeta\| \ge \frac{hW_M^2}{4A_{S\min}}, \tag{6.43}$$

or

$$\|\tilde{W}\|_F \ge W_M . \tag{6.44}$$

The fast subsystems NN saturation compensator is needed to ensure stability of the fast subsystem in the presence of the actuator saturation nonlinearity. Standard PD controller or optimal controller can not rigorously guarantee the closed-loop system stability in the presence of the actuator input signal limits. NN compensator not only provides efficient compensation thus improving the system performance, but also guarantees the stability of the overall flexible link system. The NN does not need any

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

previous, off-line learning. NN weights are adjusted on-line, in the real-time while keeping the control signal from being saturated.

6.6 Simulation

In this section, simulation results for the proposed neural network saturation compensator are presented. A single link flexible arm with pined-pined boundary conditions is considered in the simulation. The model with two flexible modes is as follows (Lewis, Jagannathan, and Yesildirek (1999)).

$$M=\begin{bmatrix}2.2024 & 0.0517 & 0.0410\\ 0.0517 & 0.0026 & 0.0036\\ 0.0410 & 0.0036 & 0.0080\end{bmatrix},\quad D=\begin{bmatrix}0.0200 & 0.0013 & 0.0027\\ 0.0013 & 0.0001 & 0.0002\\ 0.0027 & 0.0002 & 0.0004\end{bmatrix}$$
$$K=\begin{bmatrix}0 & 0 & 0\\ 0 & 14.0733 & \overset{0}{\underset{0}{}}\\ 0 & 0 & 225.1734\end{bmatrix},\quad F=G=0,\quad B=\begin{bmatrix}1.0000\\ 0.0668\\ 0.1337\end{bmatrix} \tag{6.45}$$

the state is $q=[q_r,\, q_{f1},\, q_{f2},\, \dot{q}_r,\, \dot{q}_{f1},\, \dot{q}_{f2}]$ and we select $y=[q_r,\, \dot{q}_r]$ for the tracking control of the joint position. Here two simulations scenarios are conducted. For each simulation case first the standard PD (proportional derivative) controller is designed under the saturation constraints, followed by the proposed NN controllers. The advantage of the NN saturation compensator can be compared over the standard PD controller.

The size of the NN considers the stability, performance, limitation of control efforts, and possible operating conditions. The slow convergence of the tracking error is usually due to the smallness of the network size. Moreover, if the chosen size is too large, the computation burden increases. Common approach is to start with the smaller NN size, and gradually increase it until satisfactory performance is achieved.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

In this dissertation, NN_R with sigmoidal function has four, ten and one neurons at the input, hidden and output layers, respectively. The first-layer weights V_R are selected randomly (Igelnik and Pao (1995)), (Selmic and Lewis (2000)), (Corless and Leitmann (1982)). They are uniformly randomly distributed between -1 and $+1$. These weights represent the stiffness of the sigmoid activation function. The threshold weights for the first layer v_0 are uniformly randomly distributed between -20 and +20. The threshold weights represent the bias in activation functions' positions. The second layer weights W_R are initialized to zero, and the effect of the inaccurate initialization number can be retrieved by the on-line weights tuning law methodology. For the out tracking PD controller will make the whole system stable subject to the saturated constraints before the NN saturation compensators start learning and control. NN_F has four, twenty and one neurons at the input, hidden and output layers, respectively. The first-layer weights V_F are selected randomly and are uniformly distributed between -1 and $+1$. The threshold weights for the first layer v_0 are uniformly randomly distributed between -15 and +16. The second layer weights W_F are initialized to zero.

The numerical simulation program was written in visual C++ and Matlab. The integration method is a fourth-order Runge-Kutta algorithm. The integration time step is using 0.001. Some common parameter for the two simulation cases are chosen as follows

$$\varepsilon = 0.26 \tag{6.46}$$

Control input τ is constrained by the saturation nonlinearity characterized by the parameters

$$\tau_{\max} = 2.5, \tau_{\min} = -2.5, m = 1 \ . \tag{6.47}$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

Following is the two scenarios of desired tracking cases we simulated.

6.6.1 Single Flexible Link with Sinusoid the Desired Tracking Path

PD out tracking loop controller parameters are chosen so that $K_v = 12$, $\Lambda = [1, 1]^T$

$K_F = [-0.01085 \quad 0.1223 \quad 0.0047 \quad 0.0474]$, Desired trajectory is given by

$x_1(t) = \sin(t), \quad x_2(t) = \cos(t)$. The simulation results with PD controller only are shown in Figure 6.3, Figure 6.4, and Figure 6.6.

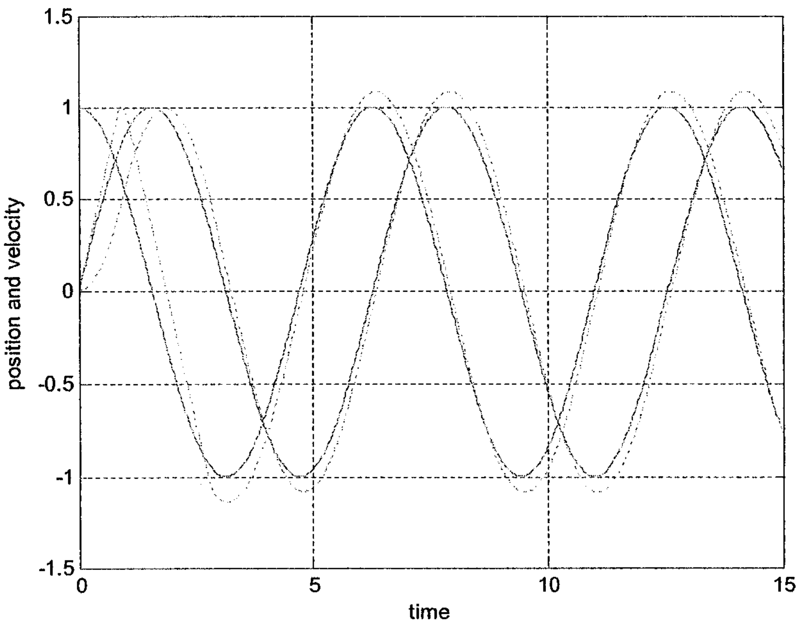

Figure 6.3. Response of flexible arm with PD controller. Actual (dashed) and desired (solid) tip position and velocity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

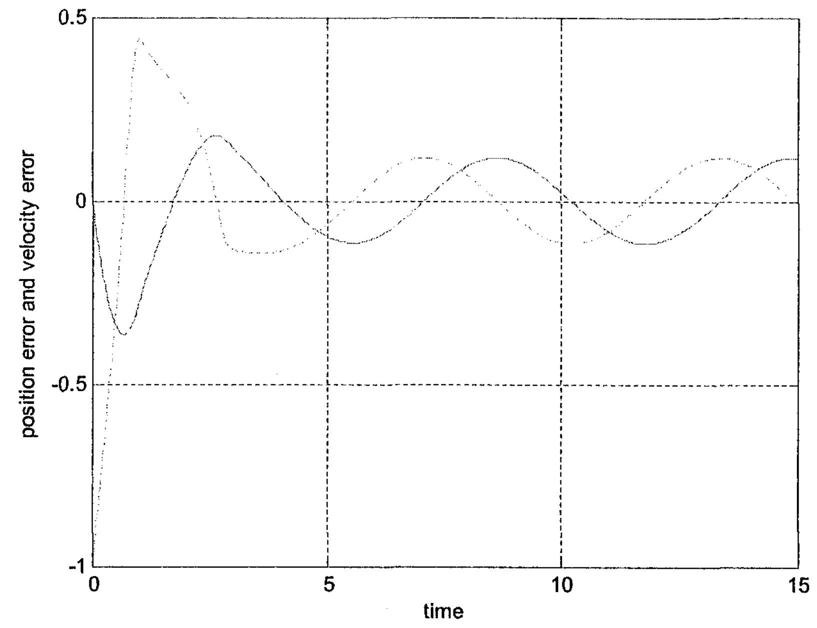

Figure 6.4. Response of flexible arm with PD controller. Position error (solid) and velocity error (dashed).

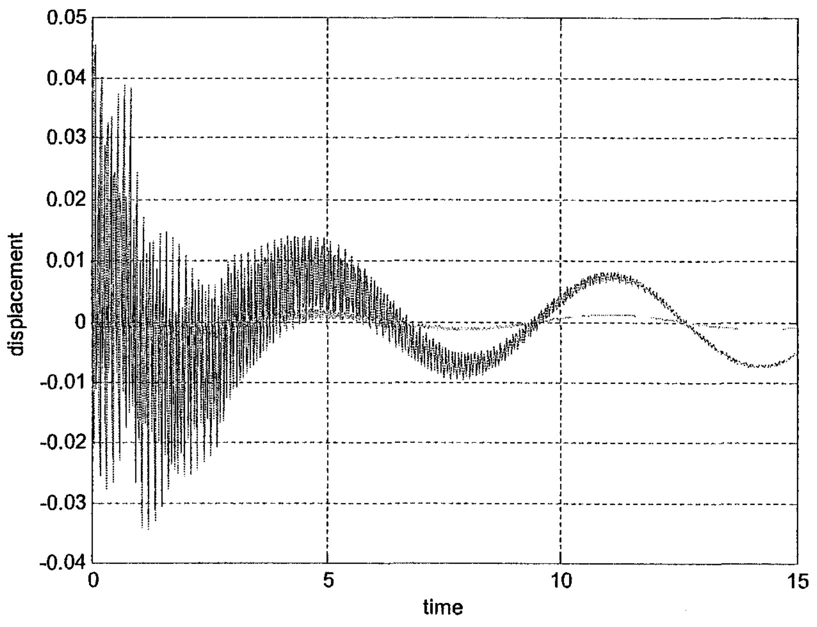

Figure 6.5. Response of flexible arm with PD controller. Flexible modes.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

The NN_R and NN_F chooses the following parameters

$$k = 0.6 \ , S = 35, f_M = 0.0 \ . \tag{6.48}$$

$$h = 0.000001 \ , \Gamma = 0.001, \varepsilon_M = 0.01 \ . \tag{6.49}$$

With the NN saturation compensation included, the simulation results are given in the Figure 6.6, Figure 6.7, and Figure 6.8.

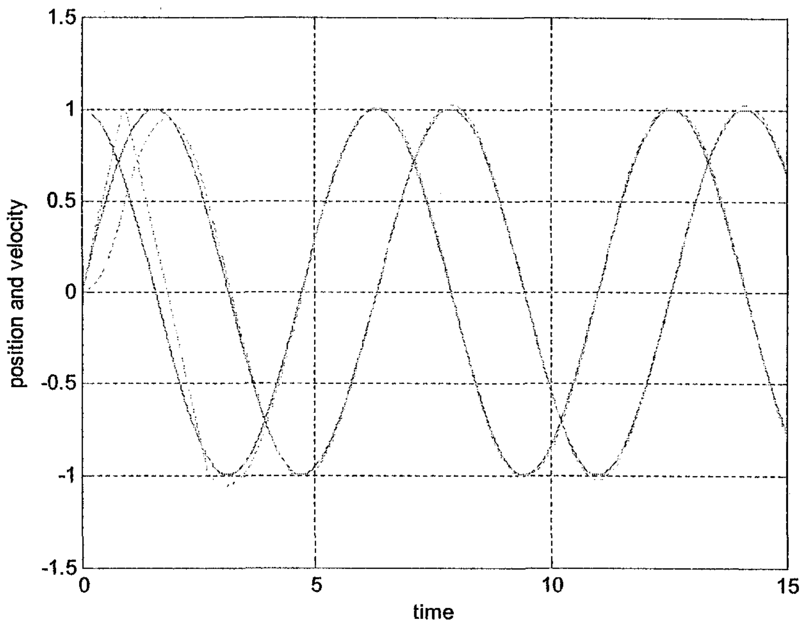

Figure 6.6. Response of flexible arm with PD controller and NN saturation compensator. Actual (dashed) and desired (solid) tip position and velocity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

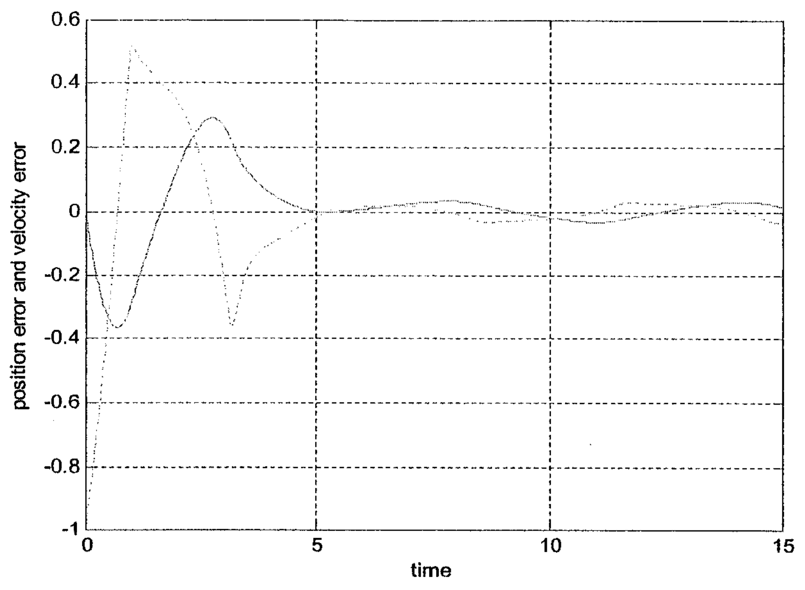

Figure 6.7. Response of flexible arm with PD controller and NN saturation compensator. Position error (solid) and velocity error (dashed).

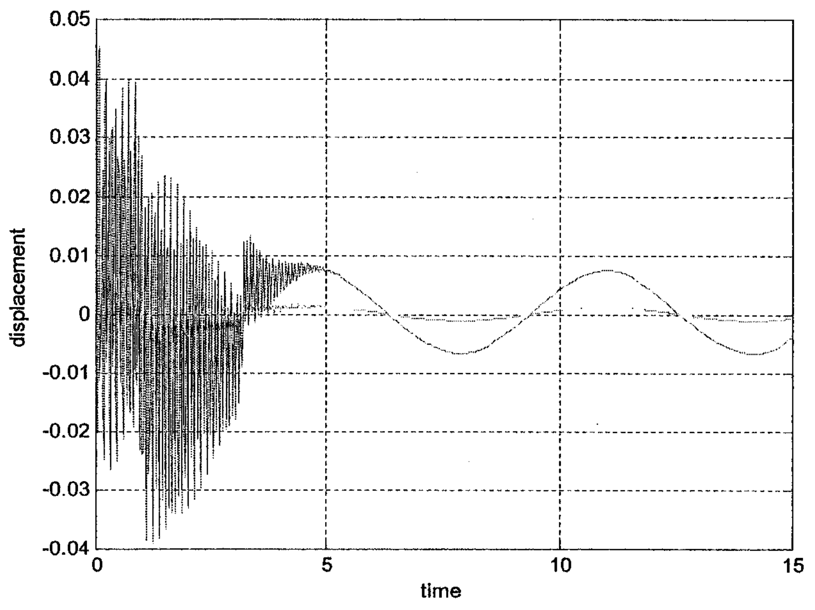

Figure 6.8. Response of flexible arm with PD controller and NN saturation compensator. Flexible modes

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

6.6.2 Single Flexible Link with Desired Acceleration/Deceleration Profile

As in (Lewis, Jagannathan, and Yesildirek (1999)), the same acceleration and/or deceleration profile is used for the open-loop testing. The open-loop position and velocity response are shown in Figure 6.9, the flexible mode in Figure 6.10.

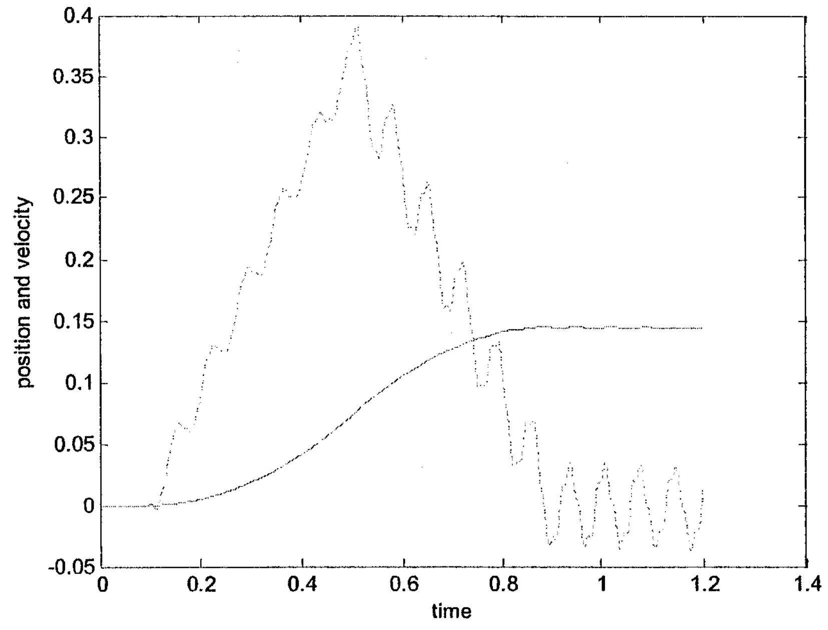

Figure 6.9. Open-loop response of flexible arm. Tip position and velocity.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

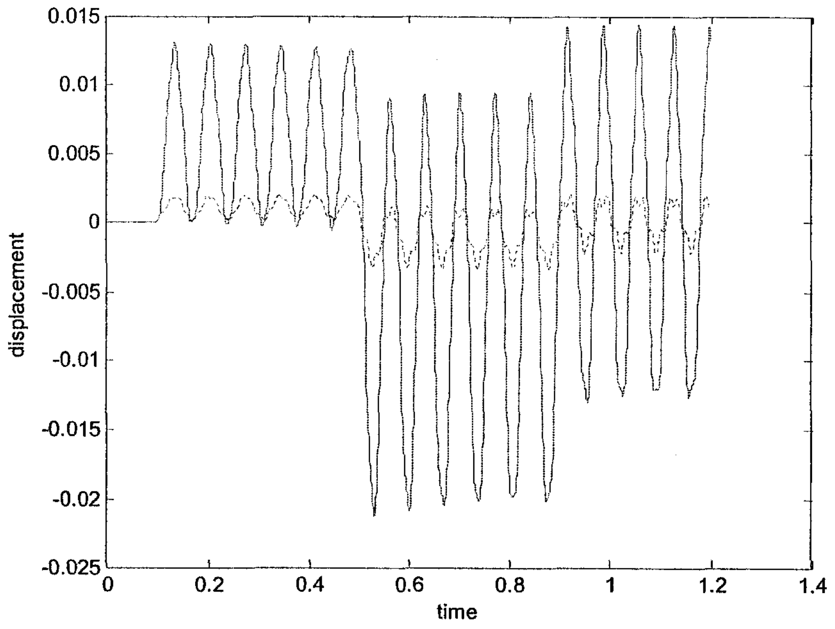

Figure 6.10. Open-loop response of flexible arm. Flexible modes.

PD out tracking loop controller parameters are chosen, so that $K_v = 15$, $\Lambda = [2, 1]^T$, $K_F = \begin{bmatrix} -0.01085 & 0.1223 & 0.0047 & 0.0474 \end{bmatrix}$. Desired trajectory is calculated based on the desired acceleration/deceleration profile and shown in the following figures. The simulation results with PD controller only are shown in Figure 6.11, Figure 6.12 and Figure 6.13.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

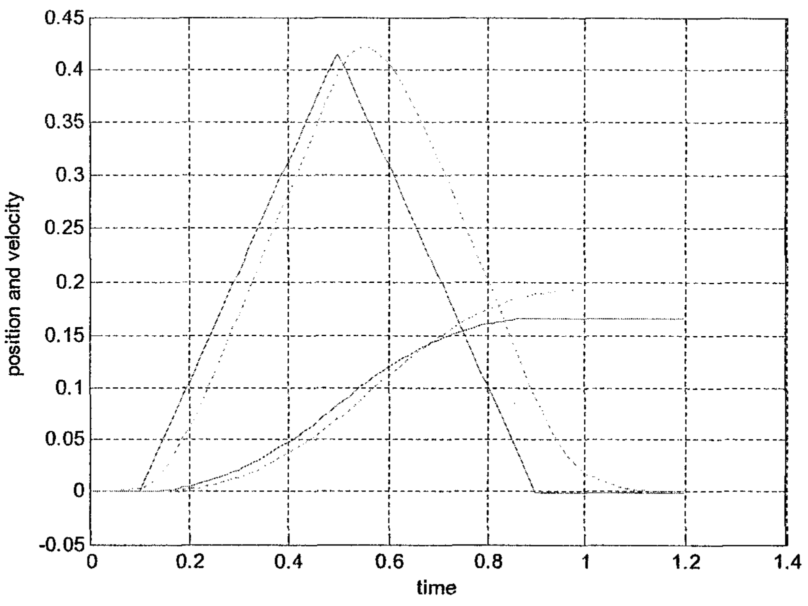

Figure 6.11. Response of flexible arm with PD controller. Actual (dashed) and desired (solid) tip position and velocity.

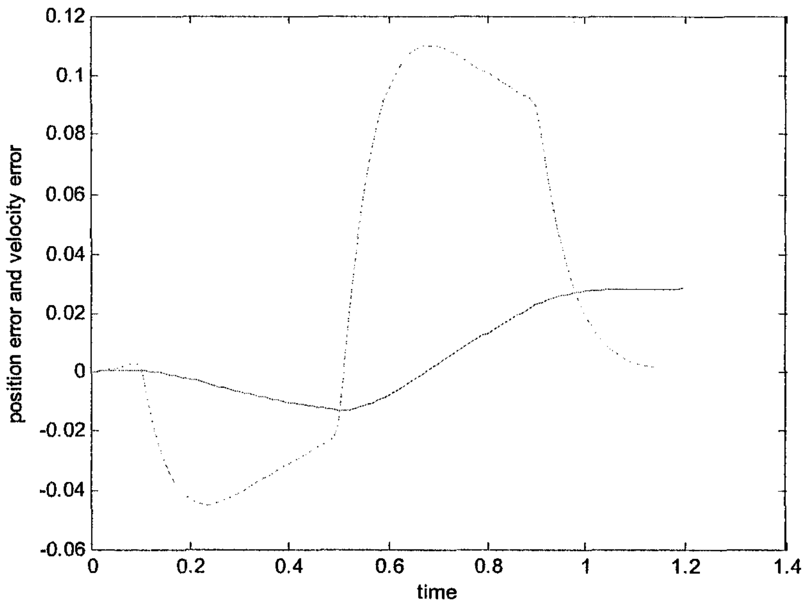

Figure 6.12. Response of flexible arm with PD controller. Position error (solid) and velocity error (dashed).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

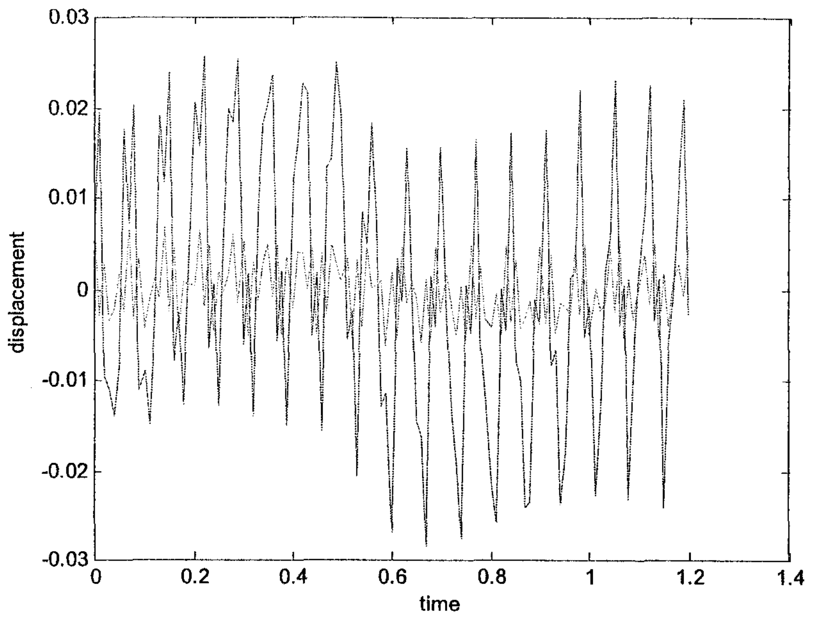

Figure 6.13. Response of flexible arm with PD controller. Flexible modes.

The NN_R and NN_F chooses the following parameters

$$k = 0.08\ , S = 20, f_M = 1.5\ . \tag{6.50}$$

$$h = 0.0001\ , \Gamma = 0.001, \varepsilon_M = 1\ . \tag{6.51}$$

With the NN saturation compensation included, the simulation results are given in the Figure 6.14, Figure 6.15 and Figure 6.16.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

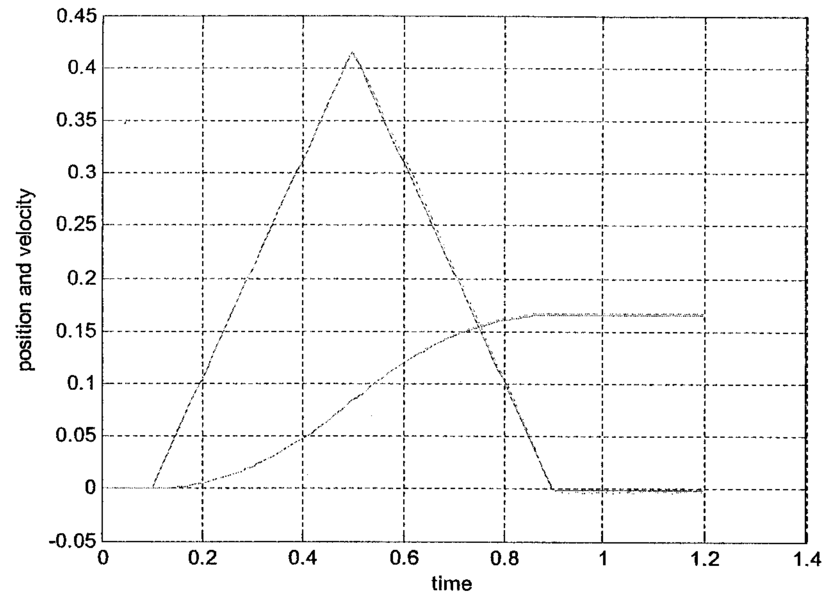

Figure 6.14. Response of flexible arm with PD controller and NN saturation compensator. Actual (dashed) and desired (solid) tip position and velocity.

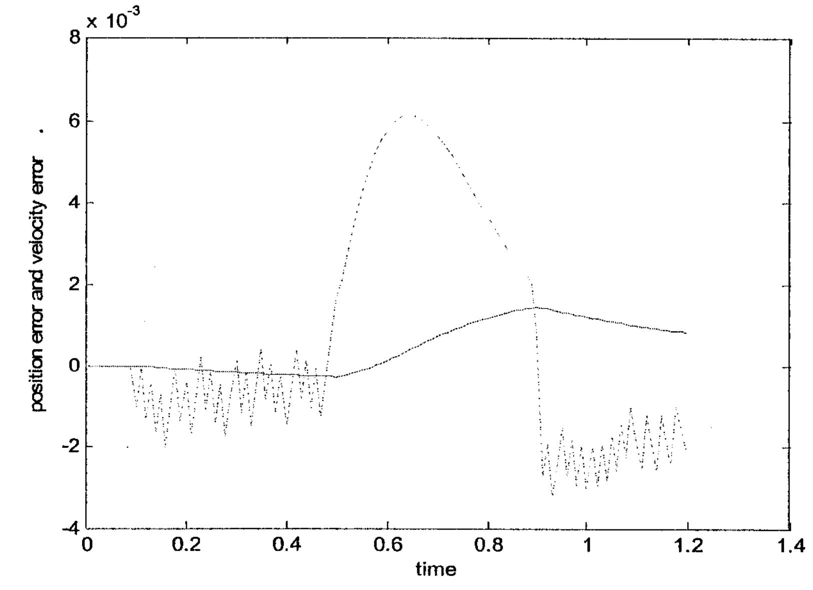

Figure 6.15. Response of flexible arm with PD controller and NN saturation compensator. Position error (solid) and velocity error (dashed).

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

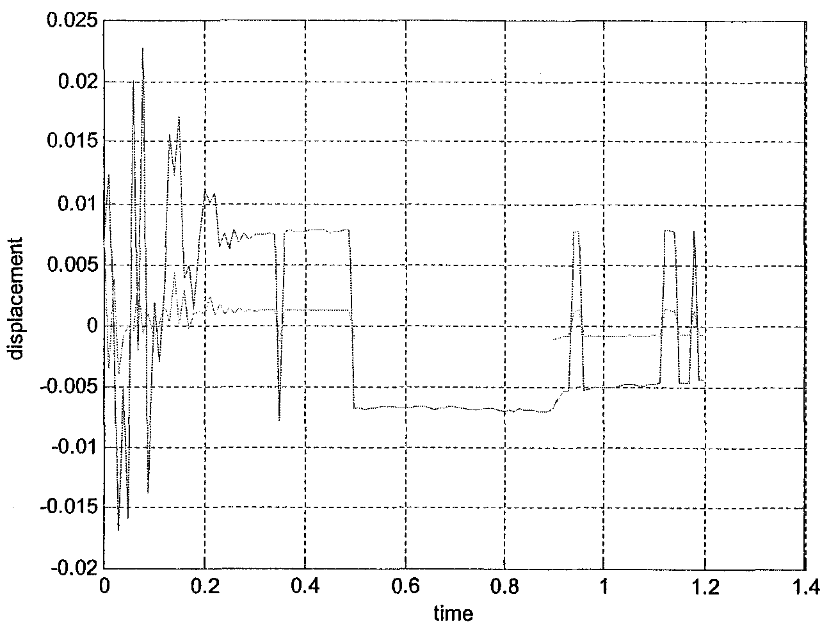

Figure 6.16. Response of flexible arm with PD controller and NN saturation compensator. Flexible modes.

From above simulation, the PD gain is selected specially to keep the flexible link tip tracking error within 10 percent subject to the saturation constraints; when the NN saturation compensator is imposed, the tracking error and flexible mode vibration are more than five times improved. It verifies the effectiveness of the proposed robust composite NN saturation compensator for single flexible link.

In this chapter, A singular-perturbation based robust composite saturation compensator consisting of an NN slow dynamics subcompensator and NN fast dynamics subcompensator is presented. It uses two neural networks for two different control components. The compensator efficiently prevents the control signal from being saturated, thus providing the intelligent anti windup saturation control scheme. Rigorous stability proof is given using Lyapunov theory. Simulation results show that the proposed

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

saturation compensation techniques can be effective for an underactuated system such as a single flexible link.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

CHAPTER 7

CONCLUSIONS AND RECOMMENDED FUTURE WORK

7.1 Conclusions

This dissertation describes intelligent actuator saturation compensation approaches for effectively controlling uncertain nonlinear systems with unknown actuator saturation. It addresses the standard mathematical actuator saturation model, filtered tracking error, designed assumptions, overall closed-loop nonlinear system dynamics, outer loop controller structure, robust NN weight tuning law, tracking performance and stability analysis, with extensive simulation results on various nonlinear system models. Through conversion of the standard actuator saturation dynamics, artificial neural network with the universal approximation property and learning capability successfully estimates the modified actuator saturation function. The developed theory in this dissertation is in a general framework readily applicable to practical NN actuator saturation compensation problems.

The proposed saturation compensation schemes apply to several common uncertain nonlinear system dynamics such as a general SISO nonlinear system with Brunovsky form and some MIMO nonlinear system dynamics (Xu and Ioannou (2003)) such as the rigid robotic dynamics and the flexible robotic dynamics. Based on filtered

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

error dynamics design and proper Lyapunov function candidate selection, the on line weight tuning law is derived to ensure the small tracking error and the bounded internal states. As indicated in the development of the compensator, no actuator output is assumed known, rigorous stability proof is given and simulation examples verify the satisfied performance.

7.2 Recommended Future Work

Since a number of compensation schemes for the common actuator nonlinearity such as saturation, deadzone, backlash, hysteresis etc. have appeared (Hu and Lin (2001)), (Gao and Selmic (2003)), (Lewis, Campos, and Selmic (2002)), (Selmic and Lewis (2001)), (Narendra and Balakrishana (1997)), (Tao and Kokotovic (1994)), (He and Jagannathan (2004)). The schemes were developed to accommodate individual actuator nonlinearity. In reality, the several kinds of actuator nonlinearity coexist with each other at the same time. If people just combine the above mentioned individual compensation scheme to tackle the combined actuator nonlinearities, the increased computation burden will be problematic. Additional research should be performed in developing the methodology of simple and composite actuator nonlinearity compensation scheme to accommodate the combined actuator nonlinearities.

In Tao's recent book on actuator failure (Tao, Chen, Tang and Joshi (2004)), adaptive law is designed to automatically to adjust the controller parameters based on the system response errors emerged from the actuator failures. He mainly focused on the fixed-value actuator failures, referred to as "lock-in-place" actuator failures, in which the actuator output may get stuck at some unknown fixed (varied, but bounded values) at

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

unknown time instants and cannot be influenced by further actuator control input. This characteristic has similarity as the saturation case addressed in this dissertation. The different point is that the NN saturation compensation control law developed in this dissertation is added in the same saturated actuator, while the adaptive actuator failure compensation control law (Tao, Chen, Tang and Joshi (2004)) is executed on some other redundant functional actuators to achieve the desired control objective. It is reasonable to extend/modify this developed saturation compensation scheme and apply to fixed-value actuator failures which have the similar characteristics as the saturation.

In the designed actuator nonlinearity compensation schemes, NNs with the universal approximation property and learning capability (Lewis, Jagannathan, and Yesildirek (1999)), (Narendra (1991)) have proved to be a powerful tool to model, identify and control the complex dynamic nonlinear systems with parameter uncertainty. NNs are used to estimate the unknown actuator failure and/or actuator nonlinear dynamics and/or function and compensate it. The objective of the use of NN is to design an adaptive on-line NN weight tuning law for updating the parameter estimates to modify the control law to accommodate the common actuator nonlinearities. Fuzzy logic has been used for its similar approximation property as NN. The classification property-that is, to discriminate information based on regions of input variables (e.g., positive or negative) of fuzzy logic (Wang (1994)) makes them natural candidate for the rejection of errors generated by the common actuator nonlinearities such as deadzone, saturation, backlash, hysteresis etc. and an ideal candidate for compensation of non analytical actuator nonlinearities (Lewis, Liu, Selmic and Wang (1997)), an extension to the research presented herein, future work should be performed in developing the actuator

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

saturation compensation using adaptive fuzzy logic, as some fore researcher has done already.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

APPENDIX A

SIMULATION SOURCE CODE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
// Saturation source code using C++

/*-----------------------------------------------------------------------------

This is control simulation of nonlinear system with saturation based on paper
"Robust composite saturation compensator for single flexible link using feedforward and
recurrent NNs"


-----------------------------------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <malloc.h>
#include <windows.h>


/* Parameters for simulation, time step, plotting time,... */
#define time_step  0.001
#define print_step  50
#define tfinal     18

#define N1        4     // Input layer
#define N2        10    // Hidden layer
#define N3         1    // Output layer

#define wtod 1/57.2958

/* Parameters of the NN (Neural Network Flexible used as compensator) */
#define NF1       9          // Input layer
#define NF2       20         // Hidden layer
#define NF3       1          // Output layer

#define ns        2        // number of slow system states
#define nsF       4          // number of fast system states
#define nsFE      4          // number of fast system estimator states

double saturationlimit = 2.50000000;             // saturation limit
double slope = 1;                              // slope of saturation

double t = 0.0;
double tF = 0.0;

double tuning_S = 20;                  // tuning parameter for NNR
double k1 = 2;                         // parameters for tuning law in NNR

double tuning_G = 18;                  // tuning parameter for NNF
double h = 0.2;                        // parameters for tuning law in NNF
/* External variables which are used by NN tuning law */
double norm_r = 0;                     // norm of the rigid states filtered tracking error;
double norm_zeta_tilda;                // norm of fast states error
double  zeta_tilda[4][1];

int N2_nn = N2;                        // NN2_nn, NN3_nn
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
int N3_nn = N3;              // are just for external variables;

int NF2_nn = NF2;              // NN2_nn, NN3_nn
int NF3_nn = NF3;              // are just for external variables;

double AS[4][4] = {0, 1, 0, 0, 0, 0, 1, 0, 0,  0, 0, 1,-1,-1,-1,-1};
double AF[4][4] = {0,      0, 1, 0, 0, 0, 0, 1, -2238.1, 13051, 0, 0, 815.7292, -6860.2, 0, 0;

double BF[4][1]= {0, 0, 3.249, 14.7183};
double p[4][4] = { 1, 0, 0, 0, 0, 1, 0,  0, 0, 0, 1, 0 , 0, 0, 0, 1};

double  NNF[1][1];           // NN output;
double wF[1][1], uF[1][1], tauF;
double   vF[4][1], norm_WF;
double epson = 0.26;

int main() {

        double  x[ns], xF[nsF], xFE[nsFE];      // states of the nonlinear systems;
        double  xd[ns];                      // desired trajectory;

        double  x_NN[N1];          // input of NNR systems, [qd, qdp,e,edot]
        double  x_NNF[NF1];       // input of NNF

        double  V_min, V_max, v0_min, v0_max;  // Initialization for NNR
        double  VF_min, VF_max, vF0_min, vF0_max;  // Initialization for NNF

         double  V[N1][N2], v0[N2];
        double  W[N2+1][N3];
        double  NN[N3][1];
        double  VF[NF1][NF2], vF0[NF2];                              // NN
        double  WF[NF2+1][NF3];

        double  RAN_MAX = 32676.0;

         double  e[2];                 // tracking errors;
         double  r;                    // filtered tracking errors;
         double  Fhat,g ;              // Approximate nonlinear robot function
         double  Yd;

        double  w, u_bar, tau;                // signals w, u, and tau;
        double  Lambda[2];                    // parameter Lambda;
         double  Kv;                          // PD gain;

        // robust term vR
        double fMx = 1.5;
        double vR, signr;

        // robust term vF
        double Kr = 100;
        double Kw = 10;

        double   KpF[1][2] = {-0.18479716427186,   1.12229191902755},
                 KdF[1][2] = {0.04679366635748,   0.47433958126960};
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
double Br_bar=1, Br_bar_inv=1;              //parameter for tao_bar and tao

double  Ff1_bar = 0, Gf1_bar = 0;
double Kff[2][2]={         14.0733, 0, 0, 225.1734};

double  Kff_tilda[2][2];
Kff_tilda[0][0]=pow(epson,2)*Kff[0][0];
Kff_tilda[0][1]=pow(epson,2)*Kff[0][1];
Kff_tilda[1][0]=pow(epson,2)*Kff[1][0];
Kff_tilda[1][1]=pow(epson,2)*Kff[1][1];

double Kff_tilda_inv[2][2] ={ 1.0511, 0, 0, 0.0657};

double Hff_bar_inv[2][2] = {0.001386, .002637, 0.002637, 0.007235};

double  Hff_bar[2][2]={2352.5, -857.4, -857.4, 450.7},
        Vfr1_bar[2][1]={-0.042, 0.337},
        Bf1_bar[2][1] = {3.2497, 14.7183};
double xi_bar[2][1];
int  i, j, k;

srand(10000);

// PARAMETERS for rigid robot PD controller
 Kv = 10.0;
Lambda[0] = 2.0;
Lambda[1] = 1.0;

// Random numbers are uniformly distributed between v_min and v_max
V_min = -1.0;     V_max = 1.0;
v0_min = -20.0;  v0_max = 20.0;

// Generate the random values for V
for (i=0; i<N1; i++)
{
for (j=0; j<N2; j++)
        {
// Generate the random number between 0 and 1 for weights V
V[i][j] = rand() / RAN_MAX;
// Transform the random number to the (V_min, V_max) interval
V[i][j] = (V[i][j] * (V_max - V_min)) + V_min;
        }
}

// Generate the random values for v0
for (i=0; i<N2; i++)
// Generate the random number between 0 and 1 for weights v0
v0[i] = rand() / RAN_MAX;
//Transform the random number to the (v0_min, v0max) interval
v0[i] = (v0[i] * (v0_max - v0_min)) + v0_min;
}


// NNF
// Random numbers are uniformly distributed between vF_min and vF_max
VF_min = -1.0;   VF_max = 1.0;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
vF0_min = -15.0; vF0_max = 15.0;
// Generate the random values for VF
for (i=0; i<NF1; i++)
{
for (j=0; j<NF2; j++)
        {
// Generate the random number between 0 and 1 for weights V
VF[i][j] = rand() / RAN_MAX;
//Transform the random number to the (V_min, V_max) interval
VF[i][j] = (VF[i][j] * (VF_max - VF_min)) + VF_min;
        }
}

// Generate the random values for vF0
for (i=0; i<NF2; i++)
// Generate the random number between 0 and 1 for weights v0
vF0[i] = rand() / RAN_MAX;
//Transform the random number to the (v0_min, v0max) interval
vF0[i] = (vF0[i] * (vF0_max - vF0_min)) + vF0_min;
}

// INITIALIZATION
for (i=0; i<ns; i++)  x[i] = 0.0;
for (i=0; i<nsF; i++) xF[i] = 0.1;
for (i=0; i<nsFE; i++) xFE[i] = -0.001;

// initialize the second layer weights to 0;
for (i=0; i<(N2+1); i++)
        for (j=0; j<N3; j++)
        W[i][j] = 0.0;


for (i=0; i<(NF2+1); i++) {
     for (j=0; j<NF3; j++)
        F[i][j] = 0.0;


// Open the files for the storage of the data
FILE  *frobot, *fweight;

frobot  = fopen("c:\\matlab6p5\\paper_flexibleN\\x_robot.dat","w");
fweight = fopen("c:\\matlab6p5\\paper_flexibleN\\x_NNweight.dat","w");

// initialization for NNF
//NNF[0][0] = 0;
uF[0][0] = 1;

/* main iteration loop */
k = 0;
do {

        // finding desired path
double amp = 1.0;
double omega = 1 ;

xd[0] = amp * sin(omega*t);
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
xd[1] = amp * omega * cos(omega*t);

// finding tracking errors
e[0] = x[0] - xd[0];
e[1] = x[1] - xd[1];

// finding filtered tracking errors
r = Lambda[0]*e[0] + Lambda[1]*e[1];

// finding the norm of r
norm_r = sqrt(r*r);

// Calculate Fhat matrix
Fhat = -0.0091*x[1];
g = 0.4541;

// signal Yd
Yd = -amp*omega*omega*sin(omega*t) + Lambda[0]*e[0];

// robust term for NNR
if(r>0) signr = 1;
 else if (r==0) signr = 0;
 else signr = -1;

vR = -fMx*signr;
vR = 0;

// rigid control signal w
w = (-Fhat - Yd - Kv * r + vR)/g;

// Calculate the NNR
// get the x_NN, input states of NN
x_NN[0] = xd[0];            x_NN[1] = xd[1];
x_NN[2] = e[0];  x_NN[3] = e[1];

// Calculate aux2 = VT * x_NN + v0;
// find the VT (transpouse);
double V_transp[N2][N1];
MatrixTransp(&V[0][0], N1, N2, &V_transp[0][0], N2, N1);

// find VT * x_NN = aux1;
double aux1[N2][1];
 MatrixMul(&V_transp[0][0], N2, N1, &x_NN[0], N1, 1,
                &aux1[0][0], N2, 1);

// find aux2 = aux1 + v0 = VT * x_NN + v0;
double aux2[N2][1];
MatrixAdd(&aux1[0][0], N2, 1, &v0[0], N2, 1,
                &aux2[0][0], N2, 1);

// find the sigmoid activation function of NN;
double sigma[N2+1][1];
for (i=1; i<(N2+1); i++)
sigma[i][0] = 1.0 / (1.0 + exp(-1*(aux2[i-1][0])));

sigma[0][0] = 1.0;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
// find NN: NN = WT * sigma;
double  WT[N3][N2+1];
MatrixTransp(&W[0][0], N2+1, N3, &WT[0][0], N3, N2+1);
MatrixMul( &WT[0][0], N3, N2+1, &sigma[0][0], N2+1, 1,
                        &NN[0][0], N3, 1);


NN[0][0] = 0;
u_bar = w - NN[0][0];

/     // SATURATION
if ((u_bar > saturationlimit/slope) || (u_bar < -saturationlimit/slope))
{

        if (u_bar >= saturationlimit/slope) tau = saturationlimit;
        if (u_bar < -saturationlimit/slope) tau = -saturationlimit;

} else {
        tau = u_bar;
       }

double KH[2][2];
MatrixMul( &Kff_tilda_inv[0][0], 2, 2,  &Hff_bar_inv[0][0], 2, 2,
              &KH[0][0], 2, 2);

double Vfr1_bar1[2][1];
Vfr1_bar1[0][0] = -Vfr1_bar[0][0]*x[1];
Vfr1_bar1[1][0] = -Vfr1_bar[1][0]*x[1];

double Bf1_bar1[2][1];
Bf1_bar1[0][0] = Bf1_bar[0][0]*tau;

Bf1_bar1[1][0] = Bf1_bar[1][0]*tau;

double VB[2][1];
MatrixAdd(&Vfr1_bar1[0][0], 2, 1, &Bf1_bar1[0][0], 2, 1,
                        &VB[0][0], 2, 1);

MatrixMul( &KH[0][0], 2, 2,  &VB[0][0], 2, 1,
              &xi_bar[0][0], 2, 1);


// get qf and qfp
double zeta12[2][1], zeta34[2][1];
double  qf[2][1],qfp[2][1];

zeta12[0][0] = xF[0];
zeta12[1][0] = xF[1];
zeta34[0][0] = xF[2];
zeta34[1][0] = xF[3];

double qf1[2][1];
MatrixAdd(&zeta12[0][0], 2, 1, &xi_bar[0][0], 2, 1,
                        qf1[0][0], 2, 1);
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
qf[0][0] = pow(epson,2)*qf1[0][0];
qf[1][0] = pow(epson,2)*qf1[1][0];
qfp[0][0] = epson*zeta34[0][0];
qfp[1][0] = epson*zeta34[1][0];


double pf[1][1], pf1[1][1] , df[1][1], df1[1][1], kx[1][1];
MatrixMul( &KpF[0][0], 1, 2,  &zeta12[0][0], 2, 1,
                &pf1[0][0], 1, 1);
MatrixMul( &KdF[0][0], 1, 2,  &zeta34[0][0], 2, 1,
                &df1[0][0], 1, 1);
pf[0][0] = -pf1[0][0];
df[0][0] = -df1[0][0];

MatrixAdd(&pf[0][0], 1, 1, &df[0][0], 1, 1,
                  &wF[0][0], 1, 1);

// robust term for NNF: vF = Kr*(zeta_tilda) + Kw*norm(WF)*(zeta_tilda)

zeta_tilda[0][0] = (xF[0] - xFE[0]);
zeta_tilda[1][0] = (xF[1] - xFE[1]);
zeta_tilda[2][0] = (xF[2] - xFE[2]);
zeta_tilda[3][0] = (xF[3] - xFE[3]);

// calculate norm_zeta_tilda
double zeta_tilda_transp[1][4], temp_ztt[1][1];
MatrixTransp(&zeta_tilda[0][0], 4, 1, &zeta_tilda_transp[0][0], 1, 4);
MatrixMul( &zeta_tilda_transp[0][0], 1,4,  &zeta_tilda[0][0], 4,1,
                &temp_ztt[0][0], 1, 1);

norm_zeta_tilda = sqrt(temp_ztt[0][0]);

// calculate norm WF
double WF_transp[NF3][NF2+1], temp_W[1][1];
MatrixTransp(&WF[0][0], NF2+1, NF3, &WF_transp[0][0], NF3, NF2+1);
MatrixMul( &WF_transp[0][0], 1, NF2+1,  &WF[0][0], NF2+1,1,
                &temp_W[0][0], 1, 1);
norm_WF = sqrt(temp_W[0][0]);

// prepair to calculate vF
double rzeta_tilda[4][1];
rzeta_tilda[0][0] = Kr*zeta_tilda[0][0];
rzeta_tilda[1][0] = Kr*zeta_tilda[1][0];
rzeta_tilda[2][0] = Kr*zeta_tilda[2][0];
rzeta_tilda[3][0] = Kr*zeta_tilda[3][0];

double wzeta_tilda[4][1];
wzeta_tilda[0][0] = Kw*norm_WF*zeta_tilda[0][0] ;
wzeta_tilda[1][0] = Kw*norm_WF*zeta_tilda[1][0];
wzeta_tilda[2][0] = Kw*norm_WF*zeta_tilda[2][0];
wzeta_tilda[3][0] = Kw*norm_WF*zeta_tilda[3][0];


MatrixAdd(&rzeta_tilda[0][0], 4, 1, &wzeta_tilda[0][0], 4, 1,
                          &vF[0][0], 4, 1);
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
for(i=0;i<4;i++) for(j=0;j<1;j++)    vF[i][j] = 0;

// calculate the NNF
x_NNF[0] = xF[0];
x_NNF[1] = xF[1];
x_NNF[2] = xF[2];
x_NNF[3] = xF[3];
x_NNF[4] = zeta_tilda[0][0];
x_NNF[5] = zeta_tilda[1][0];
x_NNF[6] = zeta_tilda[2][0];
x_NNF[7] = zeta_tilda[3][0];
x_NNF[8] = uF[0][0];

// Calculate auxF2 = VFT * x_NNF + vF0;
// find the VFT (transpouse);
double VF_transp[NF2][NF1];
MatrixTransp(&VF[0][0], NF1, NF2, &VF_transp[0][0], NF2, NF1);

// find VFT * x_NNF = auxF1;
double auxF1[NF2][1];
MatrixMul(&VF_transp[0][0], NF2, NF1, &x_NNF[0], NF1, 1,
                              &auxF1[0][0], NF2, 1);

// find auxF2 = auxF1 + vF0 = VFT * x_NNF + vF0;
double auxF2[NF2][1];
MatrixAdd(&auxF1[0][0], NF2, 1, &vF0[0], NF2, 1,
                    &auxF2[0][0], NF2, 1);

// find the sigmoid activation function of NN;
double sigmaF[NF2+1][1];
for (i=1; i<(NF2+1); i++)
          sigmaF[i][0] = 1.0 / (1.0 + exp(-1*(auxF2[i-1][0])));

sigmaF[0][0] = 1.0;

// find NN: NN = WT * sigma;
double  WFT[NF3][NF2+1];
MatrixTransp(&WF[0][0], NF2+1, NF3, &WFT[0][0], NF3, NF2+1);
MatrixMul( &WFT[0][0], NF3, NF2+1, &sigmaF[0][0], NF2+1, 1,
                    &NNF[0][0], NF3, 1);

/ get uF = wF - NNFout
NNF[0][0] = 0;
// without NNF
uF[0][0] = wF[0][0] - NNF[0][0];

// Print the chosen values to the screen and file;
if (k % print_step == 0) {
k = 0;
// print the values to the screen;
printf ("%f %f %f %f %f %f %f %f %f %f \n", t, e[0], e[1], tau,
          xF[0], xF[1], xd[0],x[0], qf[0][0], qf[1][0]);
// print the values to the file;
 fprintf(frobot, "%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f \n",t,e[0],e[1],tau,
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
xF[0], xF[1], xd[0],x[0], xd[1],x[1],qf[0][0], qf[1][0],NNF[0][0], uF[0][0],
            zeta_tilda[0][0], zeta_tilda[1][0], zeta_tilda[2][0],zeta_tilda[3][0]);
 fprintf(fweight,"%f %f\n", W[0][2], W[4][7]);
                                                                    }
k++;

// Integration for rigid robot arm dynamics
rk4(flexibleR_dyn, 2, &x[0], &tau, &x[0], t, t+time_step);

// Integration for flexible robot arm dynamics
rk4(flexibleF_dyn, 4, &xF[0], &uF[0][0], &xF[0], t, t+time_step);

// Integration for flexible robot arm estimator dynamics
rk4(flexibleFE_dyn, 4, &xFE[0], &uF[0][0], &xFE[0], t, t+time_step);

// prapare for W weights integration, the V weights are not tunable;

// NNR weight tuning

double tau_W1[N2+1][N3],tau_W[N2+1][N3];
MarixMul(&sigma[0][0], N2+1, 1, &r, 1, 1,
                              &tau_W1[0][0], N2+1, N3);
// Multiply result by S*g
for (i=0; i<(N2+1); i++) {
for (j=0; j<1; j++)
        tau_W[i][j] = tuning_S*g*tau_W1[i][j];


// Integration for NN weights W
rk4(weights_W, (N2+1)*N3, &W[0][0], &tau_W[0][0], &W[0][0], t, t+time_step);

double tau_WF1[NF2+1][NF3],tau_WF[NF2+1][NF3];
double BF_transp[1][4];
double temp_sB[NF2+1][4];
MatrixTransp(&BF[0][0], 4, 1, &BF_transp[0][0], 1, 4);
MatrixMul(&sigmaF[0][0], NF2+1, 1, &BF_transp[0][0], 1, 4,
                              &temp_sB[0][0], NF2+1, 4);
double temp_sp[NF2+1][4];
MatrixMul(&temp_sB[0][0], NF2+1, 4,&p[0][0], 4, 4,
                              &temp_sp[0][0], NF2+1, 4);
MatrixMul(&temp_sp[0][0], NF2+1, 4,&zeta_tilda[0][0], 4, 1,
                              &tau_WF1[0][0], NF2+1, 1);
// Multiply result by G
for (i=0; i<(NF2+1); i++)
for (j=0; j<1; j++)
        tau_WF[i][j] = tuning_G*tau_WF1[i][j];

// Integration for NN weights W
rk4(weights_WF, (NF2+1)*NF3, &WF[0][0], &tau_WF[0][0], &WF[0][0], t, t+time_step);

t += time_step; // increment the time;

} while (t <= tfinal);

fcloseall();
return 0;
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
} // END of main function

// Runge Kutta 4. order
void rk4 (void (*rastko)(double* x, double* tau, double* xdot),
int states_num,
double* x_in,
double* u,
double* x_out,
double t0,
double tf)

{
int i;
double h = tf-t0;
double h2 = h/2.0;
double h3 = h/3.0;
double h6 = h/6.0;
double *x_mid1, *x_mid2, *x_end, *xdot_init, *xdot_mid1;
 double *xdot_mid2, *xdot_end;

x_mid1 = (double *) calloc(states_num, sizeof(double));
x_mid2 = (double *) calloc(states_num, sizeof(double));
x_end = (double *) calloc(states_num, sizeof(double));
xdot_init = (double *) calloc(states_num, sizeof(double));
xdot_mid1 = (double *) calloc(states_num, sizeof(double));
xdot_mid2 = (double *) calloc(states_num, sizeof(double));
xdot_end = (double *) calloc(states_num, sizeof(double));

(*rastko)(x_in, u, xdot_init);     /* get xdot at initial x */

for (i=0; i<states_num; i++)
          x_mid1[i] = x_in[i] + h2*xdot_init[i];
          (*rastko)(x_mid1, u, xdot_mid1);

for (i=0; i<states_num; i++)
          x_mid2[i] = x_in[i] + h2*xdot_mid1[i];
          (*rastko)(x_mid2, u, xdot_mid2);

for (i=0; i<states_num; i++)
          x_end[i] = x_in[i] + h*xdot_mid2[i];
          (*rastko)(x_end, u, xdot_end);     /* get xdot at estimated x(t+dt) */

for (i=0; i<states_num; i++)
          x_out[i] = x_in[i] + h6*(xdot_init[i]+xdot_end[i])
          + h3*(xdot_mid1[i]+xdot_mid2[i]);

free(x_mid1); free(x_mid2); free(x_end); free(xdot_init);
free(xdot_mid1); free(xdot_mid2); free(xdot_end);


} // END

void flexibleR_dyn(double* x, double* tau, double* xdot) {

/*-----------------------------------------------------------------
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
Function for calculating the xdot, based on the dynamics of the
sustem.

INPUT VALUES:   double* x, double* tau
RETURNED VALUE: douboe* xdot
-----------------------------------------------------------------*/

// Parameters of the nonlinear system
double x0, x1, tau0;
x0 = *x;
x1 = *(x+1);

tau0 = *tau;
*xdot = x1;
*(xdot+1) = -0.0091*x1 + 0.4541*tau0;

} //Close rogid robot_dyn function

void flexibleF_dyn(double* x, double* tau, double* xdot) {

/*---------------------------------------------------------------
Function for calculating the xdot, based on the dynamics of the
sustem.

INPUT VALUES:   double* x, double* tau
RETURNED VALUE: douboe* xdot
-----------------------------------------------------------------*/

// Parameters of the nonlinear system
extern double epson, AS[4][4], AF[4][4], BF[4][1];
double x0, x1, x2, x3, tauF;

x0 = *x;
x1 = *(x+1);
x2 = *(x+2);
x3 = *(x+3);

tauF = *tau;

// State space model of the robot

*xdot = (1/epson)*x2;
*(xdot+1) = (1/epson)*x3;
*(xdot+2) = (1/epson)*(AF[2][0]*x0  + AF[2][1]*x1 + BF[2][0]*tauF);
*(xdot+3) = (1/epson)*(AF[3][0]*x0  + AF[3][1]*x1 + BF[3][0]*tauF);

} //Close flexible robot_dyn function

void flexibleFE_dyn(double* x, double* tau, double* xdot) { // here is the same tau=uF as in
above equ
extern int NF2_nn, NF3_nn;
extern double epson ,NNF[1][1], uF[1][1], BF[4][1],AS[4][4], AF[4][4], vF[4][1];

double x0, x1, x2, x3, tauF;
x0 = *x;
x1 = *(x+1);
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
x2 = *(x+2);
x3 = *(x+3);
tauF = *tau;

*xdot = (1/epson)*(x1 + vF[0][0]);
*(xdot+1) =(1/epson)*( x2 + vF[1][0]);
*(xdot+2) =(1/epson)*(x3 + BF[2][0]*(tauF+NNF[0][0]) + vF[2][0]);
*(xdot+3) =(1/epson)*( -x0  -x1 - x2 - x3 + BF[3][0]*(tauF+NNF[0][0]) + vF[3][0]);


} //Close flexible robot_dyn estimator function


// NNR tuning law dynamics
void weights_W(double* W, double* tau, double* Wdot)
{
extern double norm_r, k1, tuning_S;
extern int N2_nn, N3_nn;
int j, k;


// Tuning law: W_approxdot = tau - k1*S*norm_r*W_approx
for (j=0; j<N2_nn+1; j++) {

for (k=0; k<N3_nn; k++) {
Wdot[j*N3_nn+k] = tau[j*N3_nn+k] - k1*tuning_S*norm_r*(W[j*N3_nn+k]);
         }
}
}

// NNF tuning law dynamics
void weights_WF(double* WF, double* tau, double* WFdot)
{
extern double epson, norm_zeta_tilda, h, tuning_G;
extern int NF2_nn, NF3_nn;
int j, k;


// Tuning law: WFdot = G*sigmaF*BF'*P'*zeta_tilda - h*G*norm_zeta*WF;
for (j=0; j<NF2_nn+1; j++) {
         for (k=0; k<NF3_nn; k++) {
         WFdot[j*NF3_nn+k] = (tau[j*NF3_nn+k] -
         h*tuning_G*norm_zeta_tilda*(WF[j*NF3_nn+k]))/epson;
         }
}
}

/* ---------------------------------------------------------
Functions for matrix manipulation: transpouse, multiplication,
sumation.
-----------------------------------------------------------*/

// Sum of 2 matrices

void MatrixAdd (double *A1, int m1, int n1,
double *A2, int m2, int n2,  double *R, int p, int q) {
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
int    j, k;


// Check if matrix dimenstions are matching for this operation;
if (!(m1==m2 && m1==p && n1==n2 && n2==q)) MessageBox(NULL, "Check your matrix
dimensions !", "ERROR: Matrix dimensions not matching !", MB_ICONEXCLAMATION);

 for (j=0; j<m1; j++) {
      for (k=0; k<n1; k++) {

*(R + j*n1+k) = *(A1 + j*n1+k) + *(A2 + j*n1+k);
            }
}

} //Close MatrixAdd function

// Multiplication of the matrices

/*-----------------------------------------------------------

This function multiplis the matrix A1(m1, n1) with matrix
A2(m2, n2) . The result is the matrix R(m1, n2).

-------------------------------------------------------------*/
void MatrixMul (double *A1, int m1, int n1,
double *A2, int m2, int n2, double *R, int p, int q) {

int    j, k, l;
double  sum;


// Check if matrix dimenstions are matching for this operation;
if (!(n1==m2 && m1==p && n2==q)) MessageBox(NULL, "Check your matrix dimensions !",
"ERROR: Matrix dimensions not matching !", MB_ICONEXCLAMATION);

for (j=0; j<m1; j++) {
for (k=0; k<n2; k++) {
sum = 0;

for (l=0; l<n1; l++) {
sum = sum + (*(A1 + j*n1+l)) * (*(A2 + l*n2+k));
                        }

*(R + j*q+k) = sum;
                              }

                              }
} //Close MatrixMul function


// Transpose of matrix

/*-------

This function transpose matrix A1(m,n). The result is the matrix R(n,m)--------------*/
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

```
void MatrixTransp (double *A1, int m, int n,
double *R, int p, int q) {

int    j, k;


// Check if matrix dimenstions are matching for this operation;
if ((m != q) || (n != p)) MessageBox(NULL, "Check your matrix dimensions !", "ERROR: Matrix
dimensions not matching !", MB_ICONEXCLAMATION);

 for (j=0; j<m; j++) {
          for (k=0; k<n; k++) {
          *(R + k*m+j) = *(A1 + j*n+k);
                              }

                    }

} //Close MatrixTransp function
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

REFERENCES

[1] J. Apkarian, “Systematic Controller Design and Rapid Prototyping,” *Matlab Digest* www.mathworks.com/company/digest/dec98/systematic.shtml, Dec 1998.

[2] K. J. Astrom and B. Wittenmark, *Computer-Controlled Systems: Theory and Design,* Prentice Hall; 3rd edition, November 20, 1996.

[3] A.M. Annaswamy, S. Evesque, S. Niculescu, and A.P. Dowling, “Adaptive Control of a Class of Time-delay Systems in the Presence of Saturation,” *Adaptive Control of Nonsmooth Dynamic Systems*, Eds. G. Tao and F. Lewis, Springer-Verlag, New York, NY, 2001.

[4] B. C. Atherton, “An Analysis Package Comparing PID Anti-Windup Strategies,” *IEEE Control Systems Magazine*, Issue 2, pp. 34 – 40, April 1995.

[5] R. Barron, “Universal approximation bounds for superpositions of a sigmoidal function,” *IEEE Trans. Info. Theory*, vol. 39, no. 3, pp. 930-945, May 1993.

[6] A. G. Barto, “Reinforcement learning and adaptive critic methods,” Handbook of Intelligent Control, edited by David A. White and Donald A. Sofge, Van Nostrand Reinhold, pp. 469-492, New York. Jan. 1992.

[7] J. Campos and F. L. Lewis, “Adaptive critic neural network for feedforward compensation,” *Proc. of the American Controls Conference,* pp. 2813-2818, 1999.

[8] J. Campos, and F.L. Lewis, “Deadzone compensation in discrete-time using adaptive fuzzy logic,” *IEEE Trans. Fuzzy Systems*, vol. 7, no. 6, pp. 697-707, 1999.

[9] C.W. Chan, and K. Hui "On the existence of globally stable actuator saturation compensators," *International journal of Control*, Vol. 69, no. 6, pp. 773-788, April 1998.

[10] F.-C. Chen, and H. K. Khalil, “Adaptive control of nonlinear systems using neural networks,” *Int. J. Contr.*, vol. 55, no. 6, pp. 1299-1317, 1992.

[11] F.C. Chen and H.K. Khalil, “Adaptive control of nonlinear discrete-time systems using neural networks,” *IEEE Transactions on Automatic Control*, vol. 40, pp. 791-801, May 1995.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[12] Y. -K. Choi, M. J. Lee, S. Kim, "Design and implementation of Adaptive Neural-Network Compensator for Control Systems," *IEEE Trans. Industrial Electronics*, Vol.48, no.2, Apr 2001.

[13] S. Commuri and F. L. Lewis, "CMAC neural networks for control of nonlinear dynamical systems: structure, stability and passivity," *Proc. IEEE Int. Symp. Intell. Contr.*, Monterey, pp. 123-129, Aug. 1995.

[14] M. J. Corless and G. Leitmann, "Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems," *IEEE Trans. Automat. Contr.*, vol. 26, pp. 850-861, May, 1982.

[15] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr. Signals, Syst.*, vol. 2, no. 4, pp. 303-314, 1989.

[16] J. B. Dabney and T. L. Harman, *Mastering Simulink,* Pearson Prentice Hall, May 2003.

[17] C. A. Desoer and S. M. Shahruz, "Stability of dithered nonlinear systems with backlash or hysteresis, " *Int. J. Contr.*, vol. 43, no. 4, pp. 1045-1060, 1986.

[18] B. Friedland, *Advanced Control System design*, Prentice-Hall, New Jersey, 1996.

[19] K. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183-192, 1989.

[20] W. Gao and R. Selmic, "Neural network control of a class of nonlinear systems with actuator saturation," *Proceedings of American Control Conference*, Boston, Massachusetts, USA, pp. 2569-2574, June 2004.

[21] W. Gao and R. Selmic, "Neural network control of robot manipulator and a class of nonlinear systems with actuator saturation," *submitted to IEEE Trans. Neural networks*, July 2003.

[22] S. S. Ge, T. H. Lee, and G. Zhu, "Tip tracking control of a flexible manipulator using PD type controller," *Proceedings of IEEE Int. Conf. On Control Applications*, pp. 309-313, 1996.

[23] J. W. Gilbart and G. C. Winston, "Adaptive compensation for an optical tracking telescope,“ *Automatica*, vol. 10, pp. 125-131, 1974.

[24] L. B. Gutierrez, F. L. Lewis and J. Andy Lowe, "Implementation of a neural network tracking controller for a single flexible link: Comparing with PD and PID controller," *IEEE Trans. Industrial Electronics*, Vol. 45, no. 2, pp.307-318, April 1998.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[25] R. Hanus and Y. Peng, "Conditioning Technique for Controller with Time-Delays," *IEEE Trans. Automat. Contr.*, vol. AC-37, pp.689-692, May, 1992.

[26] P. He and S. Jagannathan, "Reinforcement learning based output feedback control of nonlinear systems with input constraints," *Proceedings of American Control Conference*, Boston, Massachusetts, USA, pp.2563-2568, June 2004.

[27] N. Hovakimyan, F. Nardi, A. Calise, "A Novel Error Observer based Adaptive Output Feedback Approach for Control of Uncertain Systems," *IEEE Transactions on Automatic Control*, vol.47, no.8, pp.1310-1314, 2002.

[28] N. Hovakimyan, F. Nardi, A. Calise, H. Lee, "Adaptive Output Feedback Control of a Class of Nonlinear Systems using Neural Networks,". *International Journal of Control,* Vol.74, no.12, pp.1161-1169, 2001.

[29] T. Hu and Z. Lin, *Control Systems with Actuator Saturation: Analysis and Design*, Birkhauser, Boston, 2001.

[30] S. Jagannathan and F. L. Lewis, "Multilayer neural network controller for a class of nonlinear systems," *Proceedings of the IEEE International Symposium on Intelligent Control,* pp.427-432, Aug.1995.

[31] Jin-quan Huang and F. L. Lewis, "Neural-network Predictive Control for Nonlinear Dynamic Systems with Time Delay," *IEEE Trans. Neural Networks*, vol. 14, no. 2, pp. 377-389, Mar. 2003.

[32] B. Igelnik and Y. H. Pao, "Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net," *IEEE Trans. Neural Networks*, vol. 6, no. 6, pp. 1320-1329, Nov. 1995.

[33] A. Isidori, *Nonlinear Control Systems*, 2nd ed. Berlin, Germany: Springer Verlag, 1989.

[34] S. Jagannathan and F. L. Lewis, "Multilayer discrete-time neural-net controller with guaranteed performance," *IEEE Trans. Neural Networks*, vol. 7, pp. 107-130, Jan. 1996.

[35] E. N. Johnson, A. J. Calise, "Neural Network Adaptive Control of Systems with Input Saturation," *In American Control Conference*, Arlington, Virginia, June 2001.

[36] H. K. Khalil, *Nonlinear Systems*, 3rd Ed., Prentice Hall, Upper Saddle River, NJ, 2002.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[37] S. P. Karason and A.M. Annaswamy, "Adaptive control in the presence of input constraints," *IEEE Transactions on Automatic Control*, Vol. 39, pp.2325-2330, 1994.

[38] Y. Kim and F. L. Lewis, *High level feedback control with neural networks,* Singapore: world scientific, 1998.

[39] J.-H. Kim, J.-H. Park, S.-W. Lee, and E. K. P. Chong, "A two-layered fuzzy logic controller for systems with deadzones, " *IEEE Trans. Industrial Electron.*, vol. 41, no. 2, pp.155-162, Apr. 1994.

[40] R. L. Kosut, "Design of linear systems with saturating linear control and bounded states," *IEEE Trans. Automat. Contr.*, 28(1), pp.121-124, 1983.

[41] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Trans. Neural Networks*, vol. 7, no. 2, pp. 1-11, Mar. 1996.

[42] F. L. Lewis, C. T. Abdallah, and D. M. Dawson, *Control of Robot Manipulators*, Macmillan, New York, 1993.

[43] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis, Philadelphia, PA, 1999.

[44] F. L. Lewis, K. Liu, R. R. Selmic, and Li-Xin Wang, "Adaptive fuzzy logic compensation of actuator deadzones," *J. Robot. Sys.*, vol. 14, no. 6, pp. 501-511, 1997.

[45] S. Lin, Goldenberg, A.A "Neural-network control of mobile manipulators," *IEEE Transactions on Autoatic Control*, Volume: 12 Issue: 5, pp.1121 – 1133, Sept. 2001.

[46] F. L. Lewis, J. Campos, and R. R. Selmic, *Neuro-Fuzzy Control of Industrial Systems With Actuator Nonlinearities,* SIAM Press, Philadelphia, PA, 2002.

[47] Z. Luo, "Direct strain feedback control of flexible robot arms: new theories and experimental result," *IEEE Trans. Automatic control*, vol. 38, no. 1, pp.1610-1612, 1993.

[48] Mathworks, Inc., *Writing S-Functions*, Version 4, Natick, Mass: 2002.

[49] Mathworks, Inc *Real-Time Workshop for use with Simulink, Version 5,* 2002.

[50] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control.* Cambridge, MA: MIT press, 1990.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[51] M. Moallem, K, Khorasani, and R. A. Patel, "Inversion-based sliding control of a flexible link manipulator," *Int. J. Control* vol, 71, no.3, pp. 477-490, Taylor and Francis Ltd, 1998.

[52] M. Moallem, K, Khorasani, and R. A. Patel, "A integral manifold approach for tip position tracking of flexible multi-link manipulators," *IEEE Transactions on Robotics and Automation,* Vol. 13, No. 6, pp. 823-837, December 1997.

[53] K. Nam, "Stabilization of Feedback Linearizable Systems Using a Radial Basis Function Network," *IEEE Trans. Automat. Contr.*, vol.44, no. 5, May 1999.

[54] K. S. Narendra, "Adaptive Control Using Neural Networks," *Neural Networks for Control*, pp. 115-142. ed. W. T. Miller, R. S. Sutton, P. J. Werbos, Cambridge: MIT Press, 1991.

[55] K. S. Narendra and A. M. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Trans. Automat. Control*, vol. 32, no. 2, pp. 134-145, Feb. 1987.

[56] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, Mar. 1990.

[57] W. Niu and M. Tomizuka, "A Robust Anti-Windup Controller Design for Asymptotic Tracking of Motion Control System Subjected to Actuator Saturation," *The 37th IEEE Conference on Decision and Control*, Tampa, December 1998.

[58] M. M. Polycarpou, "Stable adaptive neural control scheme for nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 41, no. 3, pp. 447-451, March 1996.

[59] M. M. Polycarpou and P. A. Ioannou, "Modeling, identification and stable adaptive control of continuous-time nonlinear dynamical systems using neural networks," *Proc. Amer. Contr. Conf.*, vol. 1, pp. 36-40, 1992.

[60] D. V. Pokhorov and D. C. Wunch, "Adaptive critic designs," *IEEE Transaction on Neural Networks*, vol. 8, no.5, pp. 997-1007, 1997.

[61] D. A. Recker, P. V. Kokotovic, D. Rhode, and J. Winkelman, "Adaptive nonlinear control of systems containing a dead zone," in *Proc. IEEE Conf. Decis. Contr.*, pp. 2111-2115, 1991.

[62] X. M. Ren, A. B. Rad, P. T. Chan and W. L. Lo, "Identification and control of continuous-time nonlinear systems via dynamical using neural networks," *IEEE Trans. Indus. Elec.*, vol. 50, pp. 478-486, June. 2003.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[75] J. E. Slotine and W. Li, *Applied Nonlinear Control,* J.-, Prentice Hall, 1991.

[76] J.-J. E. Slotine and W. Li, "Adaptive manipulator control: a case study," *IEEE Trans. Automat. Control*, vol. 33, no. 11, pp. 995-1003, Nov. 1988.

[77] F. C. Sun, Z. Q. Sun, L.B. Zhang and F. J. Xu, "Dynamic neuro-fuzzy adaptive control for flexible-link manipulators," *International Journal of Fuzzy Systems*, vol. 5, pp.432-437, June 2003.

[78] H. A. Talebi, R. V. Patel, and K. Khorasani, *Control of Flexible-link Manipulators using Neural Networks*, Springer 2001.

[79] H. A. Talebi, K. Khorasani and R. V. Patel, "Tracking control of a flexible-link manipulator using neural networks: experimental results," *Robotica*, vol 20, pp. 417-427. 2002.

[80] G. Tao and P. V. Kokotovic, "Adaptive Control of plants with unknown dead-zones,"*IEEE Trans. Automat. Control,* vol. 39, pp.59-68, Jan. 1994.

[81] G. Tao and P. V. Kokotovic, "Adaptive control of systems with unknown output backlash," *IEEE Trans. Automat. Control*, vol. 40, no. 2, pp. 326-330, Feb. 1995.

[82] G. Tao and P. V. Kokotovic, "Continuous-time adaptive control of systems with unknown backlash," *Trans. Automat. Control*, vol. 40, no. 6, pp. 1083-1087, June 1995.

[83] G. Tao and P. V. Kokotovic, "Discrete-time adaptive control of systems with unknown deadzones," *Int. J. Contr.*, vol. 61, no. 1, pp. 1-17, 1995.

[84] G. Tao and P. V. Kokotovic, *Adaptive Control of Systems With Actuator and Sensor Nonlinearities*, John Wiley & Sons, New York, 1996.

[85] M. Tharayil and A. Alleyne, "An Error Governor Based Saturation Compensation Scheme for PID Controllers," *University of Illinois at Urbana*-Champaign, 2001.

[86] S. Twigg and E. N. Johnson, "Use of Real Time Simulation in a Laborary Course," *Proc. of the 2003 American Society for Engineering Education Annual Conference*, 2003.

[87] K. S. Walgama and J. Sternby, "Inherent observer property in a class of anti-windup compensator," *Int. J. Control*, 52, 3, pp.705-724, 1990.

[88] R. Wai, "Tracking control based on Neural Network Strategy for Robot Manipulator," *Neurocomputing*, 51, pp.425-445, 2003.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.

[89] L. -X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice-Hall, New Jersey, 1994.

[90] P. J. Werbos, "*Neurocontrol and supervised learning: An overview and evaluation,*" Handbook of Intelligent Control, edited by David A. White and Donald A. Sofge, Van Nostrand Reinhold, pp. 65-90, New York, 1992.

[91] H. Xu and P. A. Ioannou, "Robust adaptive control for a class of MIMO nonlinear systems with guaranteed error bounds," *IEEE Trans. Automat. Contr.*, vol.48, May 2003.

[92] S. Q. Zhu, F. L. Lewis and L. R. Hunt, "Robust Stabilization of the internal Dynamics of Flexible Robots without Measuring the Velocity of the Deflection," *Proc. IEEE Conf. Decis. Contr.*, pp. 1811-1816, 1994.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.