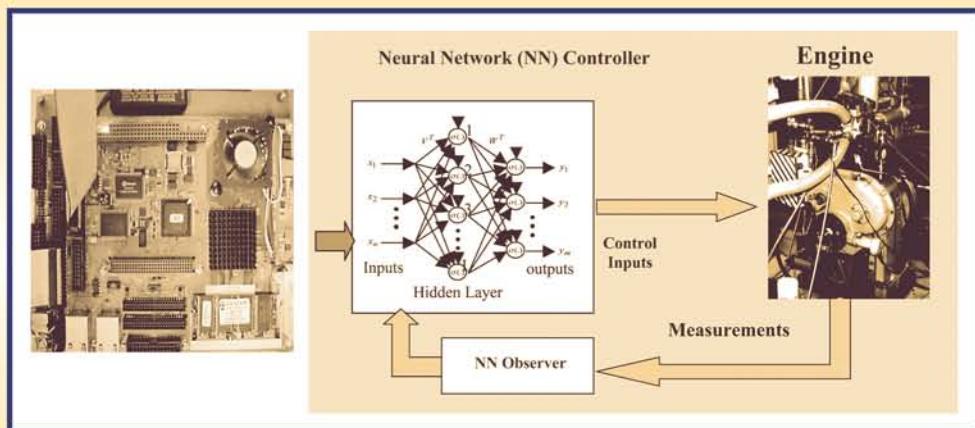


Neural Network Control of Nonlinear Discrete-Time Systems



Jagannathan Sarangapani

TLFeBOOK



Taylor & Francis
Taylor & Francis Group

**Neural Network
Control of Nonlinear
Discrete-Time
Systems**

CONTROL ENGINEERING

A Series of Reference Books and Textbooks

Editor

FRANK L. LEWIS, PH.D.

Professor

Applied Control Engineering

University of Manchester Institute of Science and Technology
Manchester, United Kingdom

1. Nonlinear Control of Electric Machinery, *Darren M. Dawson, Jun Hu, and Timothy C. Burg*
2. Computational Intelligence in Control Engineering, *Robert E. King*
3. Quantitative Feedback Theory: Fundamentals and Applications, *Constantine H. Houpis and Steven J. Rasmussen*
4. Self-Learning Control of Finite Markov Chains, *A. S. Poznyak, K. Najim, and E. Gómez-Ramírez*
5. Robust Control and Filtering for Time-Delay Systems, *Magdi S. Mahmoud*
6. Classical Feedback Control: With MATLAB®, *Boris J. Lurie and Paul J. Enright*
7. Optimal Control of Singularly Perturbed Linear Systems and Applications: High-Accuracy Techniques, *Zoran Gajic and Myo-Taeg Lim*
8. Engineering System Dynamics: A Unified Graph-Centered Approach, *Forbes T. Brown*
9. Advanced Process Identification and Control, *Enso Ikonen and Kaddour Najim*
10. Modern Control Engineering, *P. N. Paraskevopoulos*
11. Sliding Mode Control in Engineering, *edited by Wilfrid Perruquetti and Jean-Pierre Barbot*
12. Actuator Saturation Control, *edited by Vikram Kapila and Karolos M. Grigoriadis*
13. Nonlinear Control Systems, *Zoran Vukić, Ljubomir Kuljača, Dali Donlagić, and Sejid Tesnjak*
14. Linear Control System Analysis & Design: Fifth Edition, *John D'Azzo, Constantine H. Houpis and Stuart Sheldon*
15. Robot Manipulator Control: Theory & Practice, Second Edition, *Frank L. Lewis, Darren M. Dawson, and Chaouki Abdallah*
16. Robust Control System Design: Advanced State Space Techniques, Second Edition, *Chia-Chi Tsui*
17. Differentially Flat Systems, *Hebertt Sira-Ramirez and Sunil Kumar Agrawal*

18. Chaos in Automatic Control, *edited by Wilfrid Perruquetti and Jean-Pierre Barbot*
19. Fuzzy Controller Design: Theory and Applications, *Zdenko Kovacic and Stjepan Bogdan*
20. Quantitative Feedback Theory: Fundamentals and Applications, Second Edition, *Constantine H. Houpis, Steven J. Rasmussen, and Mario Garcia-Sanz*
21. Neural Network Control of Nonlinear Discrete-Time Systems, *Jagannathan Sarangapani*

Neural Network Control of Nonlinear Discrete-Time Systems

Jagannathan Sarangapani
*The University of Missouri
Rolla, Missouri*



Taylor & Francis
Taylor & Francis Group
Boca Raton London New York

CRC is an imprint of the Taylor & Francis Group,
an Informa business

Published in 2006 by

CRC Press

Taylor & Francis Group

6000 Broken Sound Parkway NW, Suite 300

Boca Raton, FL 33487-2742

© 2006 by Taylor & Francis Group, LLC

CRC Press is an imprint of Taylor & Francis Group

No claim to original U.S. Government works

Printed in the United States of America on acid-free paper

10 9 8 7 6 5 4 3 2 1

International Standard Book Number-10: 0-8247-2677-4 (Hardcover)

International Standard Book Number-13: 978-0-8247-2677-5 (Hardcover)

Library of Congress Card Number 2005036368

This book contains information obtained from authentic and highly regarded sources. Reprinted material is quoted with permission, and sources are indicated. A wide variety of references are listed. Reasonable efforts have been made to publish reliable data and information, but the author and the publisher cannot assume responsibility for the validity of all materials or for the consequences of their use.

No part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC) 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Sarangapani, Jagannathan.

Neural network control of nonlinear discrete-time systems / Jagannathan Sarangapani.

p. cm. -- (Control engineering)

Includes bibliographical references and index.

ISBN 0-8247-2677-4 (978-0-8247-2677-5)

1. Automatic control. 2. Nonlinear control theory. 3. Neural networks (Computer science) 4.

Discrete-time systems. I. Title. II. Series: Control engineering (Taylor & Francis)

TJ213.S117 2006

629.8'36--dc22

2005036368

informa

Taylor & Francis Group
is the Academic Division of Informa plc.

Visit the Taylor & Francis Web site at
<http://www.taylorandfrancis.com>

and the CRC Press Web site at
<http://www.crcpress.com>

Dedication

*This book is dedicated to my parents, my uncle, and
to my wife Sandhya, my daughter Sadhika, and
my son Anish Seshadri.*

Preface

Modern feedback control systems have been responsible for major successes in the fields of aerospace engineering, automotive technology, defense, and industrial systems. The function of a feedback controller is to alter the behavior of the system in order to meet a desired level of performance. Modern control techniques, whether linear or nonlinear, were developed using state space or frequency domain theories. These techniques were responsible for effective flight control systems, engine and emission controllers, space shuttle controllers, and for industrial systems. The complexity of today's man-made systems has placed severe constraints on existing feedback design techniques. More stringent performance requirements in both speed and accuracy in the face of system uncertainties and unknown environments have challenged the limits of modern control. Operating a complex system in different regimes requires that the controller be intelligent with adaptive and learning capabilities in the presence of unknown disturbances, unmodeled dynamics, and unstructured uncertainties. Moreover, these controllers driven by the hydraulic, electrical, pneumatic, and bio-electrical actuators have severe multiple nonlinearities in terms of friction, deadzone, backlash, and time delays.

The intelligent control systems, which are modeled after biological systems and human cognitive capabilities, possess learning, adaptation, and classification capabilities. As a result, these so-called intelligent controllers provide the hope of improved performance for today's complex systems. These intelligent controllers were being developed using artificial neural networks (NN), fuzzy logic, genetic algorithms, or a combination thereof. In this book, we explore controller design using artificial NN since NN capture the parallel processing, adaptive, and learning capabilities of biological nervous systems.

The application of NN in closed-loop feedback control systems has only recently been rigorously studied. When placed in a feedback system, even a static NN becomes a dynamical system and takes on new and unexpected behaviors. Recently, NN controllers have been developed both in continuous- and discrete-time. Controllers designed in discrete-time have the important advantage that they can be directly implemented in digital form on modern-day embedded hardware. Unfortunately, discrete-time design is far more complex than the continuous-time design when Lyapunov stability analysis is used since the first difference in Lyapunov function is quadratic in the states not linear as in the case of continuous-time.

This book for the first time presents the neurocontroller design in discrete-time. Several powerful modern control techniques in discrete-time are used in the book for the design of intelligent controllers using NN. Thorough

development, rigorous stability proofs, and simulation examples are presented in each case. Chapter 1 provides background on NN while Chapter 2 provides background information on dynamical systems, stability theory, and discrete-time adaptive controller also referred to as a self tuning regulator design. In fact, Chapter 3 lays the foundation of NN control used in the book by deriving NN controllers for a class of nonlinear systems and feedback linearizable, affine, nonlinear discrete-time systems. Both single- and multiple-layer NN controllers and NN passivity properties are covered. In Chapter 4, we introduce actuator nonlinearities and use artificial neural networks to design controllers for a class of nonlinear discrete-time systems with magnitude constraints on the input. This chapter also uses function inversion to provide NN controllers with reinforcement learning for systems with multiple nonlinearities such as dead zone and saturation. Chapter 5 confronts the additional complexity introduced by uncertainty in the control influence coefficient and presents discrete backstepping design for a class of strict feedback nonlinear discrete-time multi-input and multi-output systems. Mainly an output feedback controller is derived. Chapter 6 extends the state and output feedback controller design using NN backstepping to nonstrict feedback nonlinear systems with magnitude constraints. A practical industrial example of controlling a spark ignition engine is discussed. In Chapter 7, we discuss the system identification by developing suitable nonlinear identifier models for a broad class of nonlinear discrete-time systems using neural networks. In Chapter 8, model reference adaptive control of a class of nonlinear discrete-time systems is treated. Chapter 9 presents a novel optimal neuro controller design of a class of nonlinear discrete-time systems using Hamilton–Jacobi–Bellman formulation.

An important aspect of any control system is its implementation on an actual industrial system. Therefore, in Chapter 10 we develop the framework needed to implement intelligent control systems on actual industrial systems using embedded computer hardware. Output feedback controllers using NN were designed for lean engine operation with and without high exhaust gas recirculation (EGR) levels. Experimental results for the lean engine operation are included and EGR controller development was also included using an experimentally validated model. The appendices at the end of each chapter include analytical proofs for the controllers and computer code needed to build intelligent controllers for the above class of nonlinear systems and for real-time control applications.

This book has been written for senior undergraduate and graduate students in a college curriculum, for practicing engineers in industry, and for university researchers. Detailed derivations, stability analysis, and computer simulations show how to understand NN controllers as well as how to build them.

Acknowledgments and grateful thanks are due to my teacher, Dr. F.L. Lewis, who gave me inspiration, passion, and taught me persistence and attention to

details; Dr. Paul Werbos for introducing me to the topic of adaptive critics and guiding me along; Dr. S.N. Balakrishnan, who gave me inspiration and humor behind the control theory; Dr. J. Drallmeier who introduced me to the engine control problem. Also special thanks to all my students, in particular Pingan He, Atmika Singh, Anil Ramachandran, and Jonathan Vance who forced me to take the work seriously and become a part of it. Without monumental efforts at typing and meeting deadlines by Atmika Singh and Anil Ramachandran, this book would not be a reality.

This research work is supported by the National Science Foundation under grants ECS-9985739, ECS-0296191, and ECS-0328777.

Jagannathan Sarangapani
Rolla, Missouri

Contents

Chapter 1	Background on Neural Networks.....	1
1.1	NN Topologies and Recall.....	2
1.1.1	Neuron Mathematical Model.....	3
1.1.2	Multilayer Perceptron	8
1.1.3	Linear-in-the-Parameter NN	12
1.1.3.1	Gaussian or Radial Basis Function Networks	12
1.1.3.2	Cerebellar Model Articulation Controller Networks	13
1.1.4	Dynamic NN	15
1.1.4.1	Hopfield Network.....	15
1.1.4.2	Generalized Recurrent NN.....	19
1.2	Properties of NN	24
1.2.1	Classification and Association	25
1.2.1.1	Classification	25
1.2.1.2	Association	28
1.2.2	Function Approximation	31
1.3	NN Weight Selection and Training	35
1.3.1	Weight Computation	36
1.3.2	Training the One-Layer NN — Gradient Descent	38
1.3.2.1	Gradient Descent Tuning.....	39
1.3.2.2	Epoch vs. Batch Updating	42
1.3.3	Training the Multilayer NN — Backpropagation Tuning ..	47
1.3.3.1	Background	49
1.3.3.2	Derivation of the Backpropagation Algorithm	51
1.3.3.3	Improvements on Gradient Descent	63
1.3.4	Hebbian Tuning.....	67
1.4	NN Learning and Control Architectures	69
1.4.1	Unsupervised and Reinforcement Learning	69
1.4.2	Comparison of the Two NN Control Architectures	70
References	71	
Problems	73	

Chapter 2	Background and Discrete-Time Adaptive Control	75
2.1	Dynamical Systems	75
2.1.1	Discrete-Time Systems	75
2.1.2	Brunovsky Canonical Form	76
2.1.3	Linear Systems	77
2.2	Mathematical Background.....	79
2.2.1	Vector and Matrix Norms	79
2.2.2	Continuity and Function Norms	82
2.3	Properties of Dynamical Systems	83
2.3.1	Stability	83
2.3.2	Passivity	86
2.3.3	Interconnections of Passive Systems	87
2.4	Nonlinear Stability Analysis and Controls Design	88
2.4.1	Lyapunov Analysis for Autonomous Systems.....	88
2.4.2	Controller Design Using Lyapunov Techniques	92
2.4.3	Lyapunov Analysis for Nonautonomous Systems	97
2.4.4	Extensions of Lyapunov Techniques and Bounded Stability	99
2.5	Robust Implicit STR	102
2.5.1	Background	104
2.5.1.1	Adaptive Control Formulation	105
2.5.1.2	Stability of Dynamical Systems	106
2.5.2	STR Design	111
2.5.2.1	Structure of the STR and Error System Dynamics	111
2.5.2.2	STR Parameter Updates	112
2.5.3	Projection Algorithm	116
2.5.4	Ideal Case: No Disturbances and No STR Reconstruction Errors	117
2.5.5	Parameter-Tuning Modification for Relaxation of PE Condition	119
2.5.6	Passivity Properties of the STR	123
2.5.7	Conclusions	127
References	127	
Problems	129	
Appendix 2.A	131	
Chapter 3	Neural Network Control of Nonlinear Systems and Feedback Linearization	139
3.1	NN Control with Discrete-Time Tuning	142

3.1.1	Dynamics of the m th Order Multi-Input and Multi-Output Discrete-Time Nonlinear System.....	143
3.1.2	One-Layer NN Controller Design	145
3.1.2.1	NN Controller Design	146
3.1.2.2	Structure of the NN and Error System Dynamics	147
3.1.2.3	Weight Updates of the NN for Guaranteed Tracking Performance	148
3.1.2.4	Projection Algorithm	155
3.1.2.5	Ideal Case: No Disturbances and No NN Reconstruction Errors	156
3.1.2.6	Parameter Tuning Modification for Relaxation of PE Condition.....	160
3.1.3	Multilayer NN Controller Design.....	167
3.1.3.1	Error Dynamics and NN Controller Structure	170
3.1.3.2	Multilayer NN Weight Updates	172
3.1.3.3	Projection Algorithm	179
3.1.3.4	Multilayer NN Weight-Tuning Modification for Relaxation of PE Condition	185
3.1.4	Passivity of the NN	191
3.1.4.1	Passivity Properties of the Tracking Error System	191
3.1.4.2	Passivity Properties of One-Layer NN	192
3.1.4.3	Passivity of the Closed-Loop System.....	195
3.1.4.4	Passivity of the Multilayer NN.....	196
3.2	Feedback Linearization	197
3.2.1	Input–Output Feedback Linearization Controllers	197
3.2.1.1	Error Dynamics	198
3.2.2	Controller Design	199
3.3	NN Feedback Linearization	200
3.3.1	System Dynamics and Tracking Problem	201
3.3.2	NN Controller Design for Feedback Linearization	204
3.3.2.1	NN Approximation of Unknown Functions	204
3.3.2.2	Error System Dynamics	206
3.3.2.3	Well-Defined Control Problem	209
3.3.2.4	Controller Design.....	210
3.3.3	One-Layer NN for Feedback Linearization	211
3.3.3.1	Weight Updates Requiring PE	211
3.3.3.2	Projection Algorithm	222
3.3.3.3	Weight Updates not Requiring PE	223
3.4	Multilayer NN for Feedback Linearization	233
3.4.1	Weight Updates Requiring PE	234

3.4.2	Weight Updates Not Requiring PE.....	236
3.5	Passivity Properties of the NN	254
3.5.1	Passivity Properties of the Tracking Error System.....	255
3.5.2	Passivity Properties of One-Layer NN Controllers	256
3.5.3	Passivity Properties of Multilayer NN Controllers.....	256
3.6	Conclusions	259
	References	259
	Problems	262
Chapter 4	Neural Network Control of Uncertain Nonlinear Discrete-Time Systems with Actuator Nonlinearities	265
4.1	Background on Actuator Nonlinearities	266
4.1.1	Friction	266
4.1.1.1	Static Friction Models	267
4.1.1.2	Dynamic Friction Models	268
4.1.2	Deadzone	269
4.1.3	Backlash.....	272
4.1.4	Saturation	273
4.2	Reinforcement NN Learning Control with Saturation	274
4.2.1	Nonlinear System Description	276
4.2.2	Controller Design Based on the Filtered Tracking Error ...	277
4.2.3	One-Layer NN Controller Design	279
4.2.3.1	The Strategic Utility Function	279
4.2.3.2	Critic NN.....	280
4.2.3.3	Action NN	281
4.2.4	NN Controller without Saturation Nonlinearity	283
4.2.5	Adaptive NN Controller Design with Saturation Nonlinearity	287
4.2.5.1	Auxiliary System Design.....	287
4.2.5.2	Adaptive NN Controller Structure with Saturation	288
4.2.5.3	Closed-Loop System Stability Analysis.....	288
4.2.6	Comparison of Tracking Error and Reinforcement Learning-Based Controls Design	296
4.3	Uncertain Nonlinear System with Unknown Deadzone and Saturation Nonlinearities	297
4.3.1	Nonlinear System Description and Error Dynamics	300
4.3.2	Deadzone Compensation with Magnitude Constraints	300
4.3.2.1	Deadzone Nonlinearity	300
4.3.2.2	Compensation of Deadzone Nonlinearity	301
4.3.2.3	Saturation Nonlinearities	303

4.3.3	Reinforcement Learning NN Controller Design	304
4.3.3.1	Error Dynamics.....	304
4.3.3.2	Critic NN Design	305
4.3.3.3	Main Result	306
4.4	Adaptive NN Control of Nonlinear System with Unknown Backlash	309
4.4.1	Nonlinear System Description	310
4.4.2	Controller Design Using Filtered Tracking Error without Backlash Nonlinearity	311
4.4.3	Backlash Compensation Using Dynamic Inversion	312
4.5	Conclusions	319
References	320
Problems	323
Appendix 4.A	325
Appendix 4.B	329
Appendix 4.C	330
Appendix 4.D	338

Chapter 5 Output Feedback Control of Strict Feedback Nonlinear MIMO Discrete-Time Systems 343

5.1	Class of Nonlinear Discrete-Time Systems.....	345
5.2	Output Feedback Controller Design	345
5.2.1	Observer Design	346
5.2.2	NN Controller Design	347
5.2.2.1	Auxiliary Controller Design	348
5.2.2.2	Controller Design with Magnitude Constraints ...	349
5.3	Weight Updates for Guaranteed Performance	350
5.3.1	Weights Updating Rule for the Observer NN	350
5.3.2	Strategic Utility Function	351
5.3.3	Critic NN Design	351
5.3.4	Weight-Updating Rule for the Action NN.....	353
5.4	Conclusions	361
References	362
Problems	363
Appendix 5.A	364
Appendix 5.B	366

Chapter 6 Neural Network Control of Nonstrict Feedback Nonlinear Systems 371

6.1	Introduction	371
-----	--------------------	-----

6.1.1	Nonlinear Discrete-Time Systems in Nonstrict Feedback Form	371
6.1.2	Backstepping Design	373
6.2	Adaptive NN Control Design Using State Measurements	374
6.2.1	Tracking Error-Based Adaptive NN Controller Design	375
6.2.1.1	Adaptive NN Backstepping Controller Design ...	375
6.2.1.2	Weight Updates	378
6.2.2	Adaptive Critic-Based NN Controller Design.....	381
6.2.2.1	Critic NN Design	382
6.2.2.2	Weight-Tuning Algorithms.....	383
6.3	Output Feedback NN Controller Design.....	392
6.3.1	NN Observer Design	394
6.3.2	Adaptive NN Controller Design	396
6.3.3	Weight Updates for the Output Feedback Controller	400
6.4	Conclusions	406
	References	407
	Problems	409
	Appendix 6.A	411
	Appendix 6.B	419

Chapter 7	System Identification Using Discrete-Time Neural Networks	423
7.1	Identification of Nonlinear Dynamical Systems.....	425
7.2	Identifier Dynamics for MIMO Systems.....	426
7.3	NN Identifier Design.....	429
7.3.1	Structure of the NN Identifier and Error System Dynamics.....	430
7.3.2	Multilayer NN Weight Updates	432
7.4	Passivity Properties of the NN	439
7.5	Conclusions	443
	References	444
	Problems	444

Chapter 8	Discrete-Time Model Reference Adaptive Control.....	447
8.1	Dynamics of an m nth-Order Multi-Input and Multi-Output System	448
8.2	NN Controller Design	451
8.2.1	NN Controller Structure and Error System Dynamics	451
8.2.2	Weight Updates for Guaranteed Tracking Performance	454
8.3	Projection Algorithm	460

8.4	Conclusions	468
References	469	
Problems	470	

Chapter 9 Neural Network Control in Discrete-Time Using Hamilton–Jacobi–Bellman Formulation 473

9.1	Optimal Control and Generalized HJB Equation in Discrete-Time	475
9.2	NN Least-Squares Approach	486
9.3	Numerical Examples.....	490
9.4	Conclusions	508
References	508	
Problems	509	

Chapter 10 Neural Network Output Feedback Controller Design and Embedded Hardware Implementation..... 511

10.1	Embedded Hardware-PC Real-Time Digital Control System	512
10.1.1	Hardware Description	512
10.1.2	Software Description	514
10.2	SI Engine Test Bed.....	514
10.2.1	Engine-PC Interface Hardware Operation.....	516
10.2.2	PC Operation.....	518
10.2.3	Timing Specifications for Controller	520
10.2.4	Software Implementation.....	521
10.3	Lean Engine Controller Design and Implementation	523
10.3.1	Engine Dynamics	526
10.3.2	NN Observer Design	528
10.3.3	Adaptive NN Output Feedback Controller Design.....	530
10.3.3.1	Adaptive NN Backstepping Design.....	531
10.3.3.2	Weight Updates for Guaranteed Performance.....	535
10.3.4	Simulation of NN Controller C Implementation	537
10.3.5	Experimental Results	539
10.4	EGR Engine Controller Design and Implementation	547
10.4.1	Engine Dynamics with EGR	549
10.4.2	NN Observer Design	551
10.4.3	Adaptive Output Feedback EGR Controller Design	553
10.4.3.1	Error Dynamics	554
10.4.3.2	Weight Updates for Guaranteed Performance.....	557
10.4.4	Numerical Simulation	559

10.5 Conclusions	563
References	564
Problems	565
Appendix 10.A	566
Appendix 10.B.....	570
Index	595

1 Background on Neural Networks

In this chapter, a brief background on neural networks (NN) will be included covering mainly the topics that will be important in a discussion of NN applications in closed-loop control of discrete-time dynamical systems. Included are the NN topologies and recall, properties, training techniques, and control architectures. Applications are given in classification, function approximation, with examples provided using the MATLAB® NN toolbox (MATLAB 2004, MATLAB NN Toolbox 1995).

Surveys of NN are given, for instance, by Lippmann (1987), Simpson (1992), and Hush and Horne (1993); many books are also available, as exemplified by Haykin (1994), Kosko (1992), Kung (1993), Levine (1991), Peretto (1992), and other books too many to mention. It is not necessary to have an exhaustive knowledge of NN pattern recognition applications for feedback control purposes. Only a few network topologies, tuning techniques, and properties are important, especially the NN function approximation property (Lewis et al. 1999). These are the topics of this chapter and for more details on background on NN, refer Lewis et al.(1999), Haykin (1994), and so on.

Applications of NN in closed-loop digital control are dramatically distinct from those in open-loop applications, which are mainly in digital signal processing (DSP). The latter include classification, pattern recognition, and approximation of nondynamic functions (e.g., with time delays). In DSP applications, NN usage is developed over the years that show how to choose network topologies and select weights to yield guaranteed performance. The issues associated with weight-training algorithms are well understood. By contrast, in closed-loop control of dynamical systems, most applications have been ad hoc, with open-loop techniques (e.g., backpropagation weight tuning) employed in a naïve yet hopeful manner to solve problems associated with dynamic NN evolution within a feedback loop, where the NN must provide stabilizing controls for the system as well as ensure that all its weights remain bounded. Most published papers have consisted of only limited discussion followed by simulation examples. Very limited work has been done in applying and demonstrating these concepts on hardware.

By now, several researchers have begun to provide rigorous mathematical analyses of NN in closed-loop control applications (see Chapter 3). The background for these efforts was provided by Narendra and coworkers in several seminal works (see References) in the early 1990s followed by Lewis and coworkers (see References) in early to mid-1990s. It has been discovered that standard open-loop weight-tuning algorithms such as backpropagation or Hebbian tuning must be modified to provide guaranteed stability and tracking in feedback control systems (Lewis et al. 1999).

1.1 NN TOPOLOGIES AND RECALL

Artificial NN are modeled on biological processes for information processing, including specifically the nervous system and its basic unit, the neuron. Signals are propagated in the form of potential differences between inside and outside of cells. The components of a neuronal cell are shown in Figure 1.1. Dendrites bring signals from other neurons into the cell body or soma, possibly multiplying each incoming signal by a transfer weighting coefficient. In the soma, cell capacitance integrates the signals which collect in the axon hillock. Once the combined signal exceeds a certain cell threshold, a signal, the action potential, is transmitted through the axon. Cell nonlinearities make the composite action potential a nonlinear function of the combination of arriving signals. The axon connects through the synapses with the dendrites of subsequent neurons. The synapses operate through the discharge of neurotransmitter chemicals across intercellular gaps, and can be either excitatory (tending to fire the next neuron) or inhibitory (tending to prevent the firing of the next neuron).

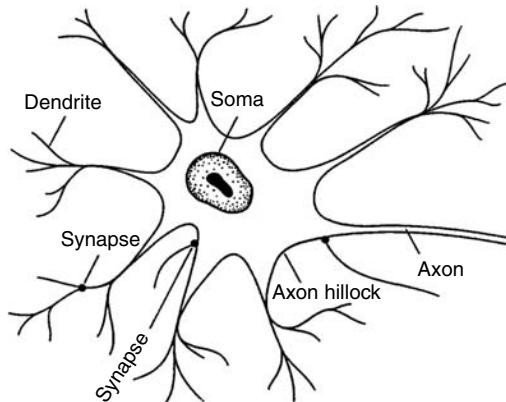


FIGURE 1.1 Neuron anatomy. (Reprinted from B. Kosko, *Neural Networks and Fuzzy Systems*, Prentice Hall, NJ, 1992. With permission.)

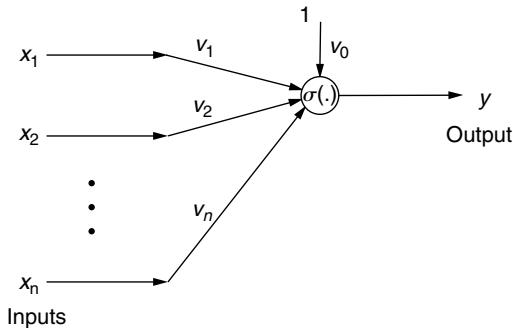


FIGURE 1.2 Mathematical model of a neuron.

1.1.1 NEURON MATHEMATICAL MODEL

A mathematical model of the neuron is depicted in Figure 1.2, which shows the dendrite weights v_j , the firing threshold v_0 (also called the bias), the summation of weighted incoming signals, and the nonlinear function $\sigma(\cdot)$. The cell inputs are the n signals at the time instant $kx_1(k), x_2(k), x_3(k), \dots, x_n(k)$ and the output is the scalar $y(k)$, which can be expressed as

$$y(k) = \sigma \left(\sum_{j=1}^n v_j x_j(k) + v_0 \right) \quad (1.1)$$

Positive weights v_j correspond to excitatory synapses and negative weights to inhibitory synapses. This network was called the perceptron by Rosenblatt in 1959 (Haykin 1994).

The nonlinear cell function is known as the activation function. The activation functions are selected specific to the applications though some common choices are illustrated in Figure 1.3. The intent of the activation function is to model the nonlinear behavior of the cell where there is no output below a certain value of the argument. Sigmoid functions are a general class of monotonically nondecreasing functions taking on bounded values between $-\infty$ and $+\infty$. It is noted that, as the threshold or bias v_0 changes, the activation functions shift left or right. For many NN training algorithms (including backpropagation), the derivative of $\sigma(\cdot)$ is needed so that the activation function selected must be differentiable.

The expression for the neuron output $y(k)$ at the time instant k ($y(t)$ in the case of continuous-time) can be streamlined by defining the column vector of

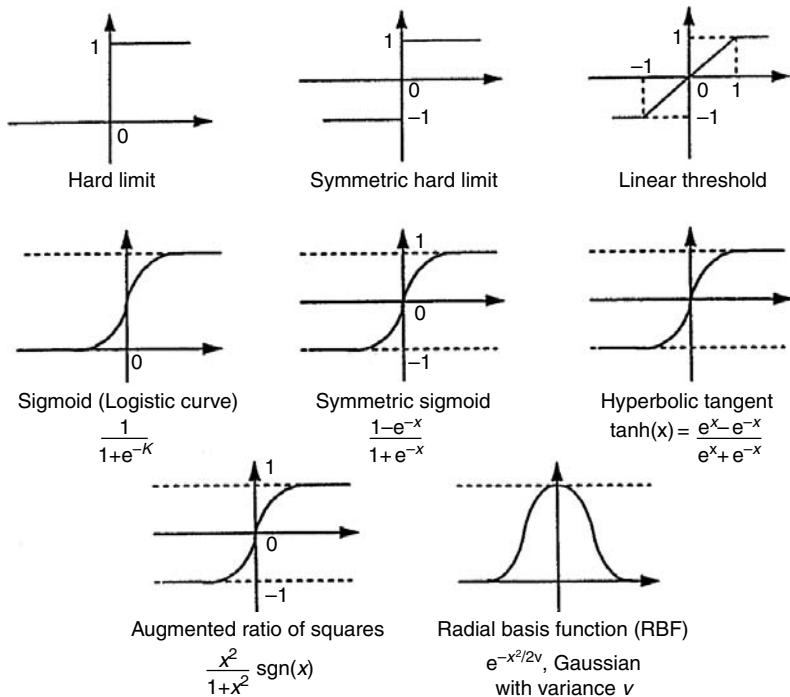


FIGURE 1.3 Common choices for the activation functions.

NN weights $\bar{v}(k) \in \Re^n$ as

$$\bar{x}(k) = [x_1 \ x_2 \ \dots \ x_n]^T \quad \bar{v}(k) = [v_1 \ v_2 \ \dots \ v_n]^T, \quad (1.2)$$

Then, it is possible to write in matrix notation

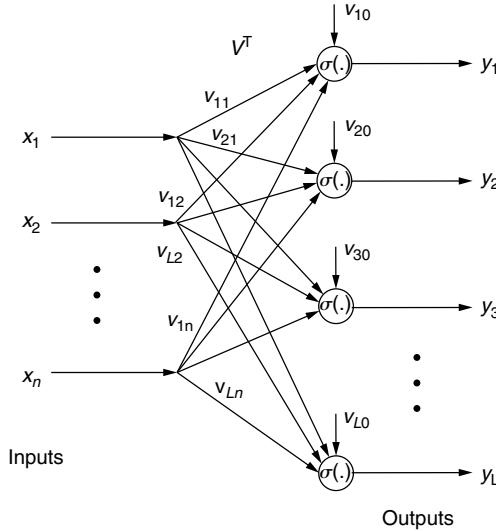
$$y = \sigma(\bar{v}^T \bar{x}) + v_0 \quad (1.3)$$

Defining the augmented input column vector $x(k) \in \Re^{n+1}$ and NN weight column vector $v(k) \in \Re^{n+1}$ as

$$\begin{aligned} x(k) &= [1 \ \bar{x}^T]^T = [1 \ x_1 \ x_2 \ \dots \ x_n] \\ v(k) &= [v_0 \ \bar{v}^T]^T = [v_0 \ v_1 \ v_2 \ \dots \ v_n]^T \end{aligned} \quad (1.4)$$

one may write

$$y = \sigma(v^T x). \quad (1.5)$$

**FIGURE 1.4** One-layer NN.

Though the input vector $\bar{x}(k) \in \Re^n$ and the weight vector $\bar{v}(k) \in \Re^n$ have been augmented by 1 and v_0 , respectively, to include the threshold, we may at times loosely say that $x(k)$ and v are elements of \Re^n .

The neuron output expression vector $y(k)$ is referred to as the cell recall mechanism. They describe how the output is reconstructed from the input signals and the values of the cell parameters.

Figure 1.4 shows an NN consisting of L cells, all fed by the same input signals $x_j(k)$ and producing one output $y(k)$ per neuron. We call this a one-layer NN. The recall equation for this network is given by

$$y_l(k) = \sigma \left(\sum_{j=1}^n v_{lj} x_j(k) + v_{l0} \right); \quad l = 1, 2, \dots, L \quad (1.6)$$

It is convenient to write the weights and the thresholds in a matrix and vector forms, respectively. By defining the matrix of weights and the vector of thresholds as

$$\bar{V}^T \equiv \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{L1} & v_{L2} & \cdots & v_{Ln} \end{bmatrix}, \quad b_v = \begin{bmatrix} v_{10} \\ v_{20} \\ \vdots \\ v_{L0} \end{bmatrix}, \quad (1.7)$$

One may write the output vector $y(t) = [y_0 \ y_1 \ y_2 \ \cdots \ y_L]^T$ as

$$y = \bar{\sigma}(\bar{V}^T \bar{x} + b_v) \quad (1.8)$$

The vector activation function is defined for a vector $w \equiv [w_1 \ w_2 \ \cdots \ w_L]^T$ as

$$\bar{\sigma}(w) \equiv [\bar{\sigma}(w)_1 \ \bar{\sigma}(w)_2 \ \cdots \ \bar{\sigma}(w)_L]^T \quad (1.9)$$

A further refinement may be achieved by inserting the threshold vector as the first column of the augmented matrix of weights as

$$V^T \equiv \begin{bmatrix} v_{10} & v_{11} & \cdots & v_{1n} \\ v_{20} & v_{21} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{L0} & v_{L1} & \cdots & v_{Ln} \end{bmatrix} \quad (1.10)$$

Then, the NN outputs may be expressed in terms of the augmented input vector $x(k)$ as

$$y = \bar{\sigma}(V^T x) \quad (1.11)$$

In other works (e.g., the MATLAB NN Toolbox) the matrix of weights may be defined as the transpose of our version; our definition conforms more closely to the usage in control system literature.

Example 1.1.1 (Output Surface for One-Layer NN): A perceptron with two inputs and one output is given by the equation $y = \sigma(-4.79x_1 + 5.90x_2 - 0.93) \equiv \sigma(vx + b)$, where $v \equiv \bar{V}^T$ (Lewis et al. 1999). Plots of the NN output surface y as a function of the inputs x_1, x_2 over the grid $[-2, 2] \times [-2, 2]$ are given in Figure 1.5. Output surfaces corresponding to the specific activation functions used are shown. To make this plot, the MATLAB NN Toolbox 4.0 was used. The following sequence of commands was used:

```
% Example 1.1.1: Output surface of one-layer NN

% Set up plotting grid for sampling x
[x1,x2] = meshgrid(-2:0.1:2);

% Compute NN input vectors p and simulate NN using sigmoid
p1 = x1(:);
p2 = x2(:);
p = [p1';p2'];
```

```
%Setup NN weights and bias
net = newff(minmax(p), [1], {'hardlim'});
net.IW{1,1}=[-4.79 5.9]; net.b{1}=[-0.93];

%Simulate NN
a = sim(net,p);

%Format results for using 'mesh' or 'surfl' plot
routines:
a1 = eye(41);
a1(:) = a';
mesh(x1,x2,a1);
xlabel('x1');
ylabel('x2');
title('NN output surface using hardlimit');
```

If the reader is unfamiliar with MATLAB programming, it is important to read the MATLAB *User's Guide* to understand the use of the colon in matrix formatting. The prime on vectors or matrices (e.g., $p1'$) means matrix transpose. The semicolon at the end of a command suppresses printing of the result in the command window. The symbol % means that the rest of the statement is a comment. It is important to note that MATLAB defines NN weight matrices as the transposes of our weight matrices; therefore, in all examples the MATLAB convention is followed (we use lowercase letters here to help make the distinction). There are routines that compute the outputs of various NN given the inputs; for instance NEWFF() is used in this example to create the network. The functions of three-dimensional (3D) plotting routines MESH and SURFL should be studied.

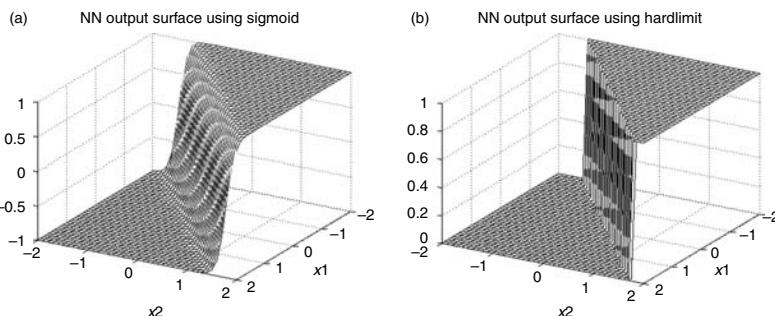


FIGURE 1.5 Output surface of a one-layer NN. (a) Using sigmoidal activation function. (b) Using hard limit function.

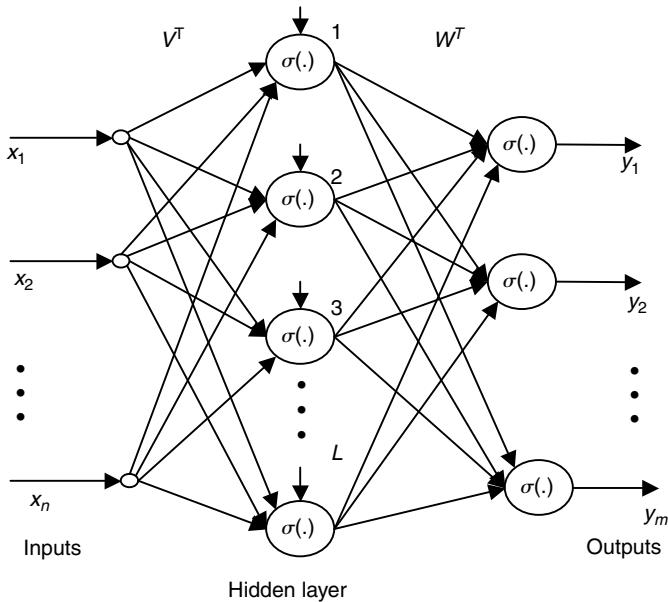


FIGURE 1.6 Two-layer neural network.

1.1.2 MULTILAYER PERCEPTRON

A two-layer NN, which has two layers of neurons, with one layer having \$L\$ neurons feeding a second layer having \$m\$ neurons, is depicted in Figure 1.6. The first layer is known as the hidden layer, with \$L\$ being the number of hidden-layer neurons; the second layer is known as the output layer. An NN with multiple layers is called a multilayer perceptron; its computing power is significantly enhanced over the one-layer NN. With a one-layer NN it is possible to implement digital operations such as AND, OR, and COMPLEMENT (see the problems section). However, research in NN was stopped many years ago when it was shown that the one-layer NN is incapable of performing the EXCLUSIVE OR operation, which is a basic problem in digital logic design. It was later demonstrated that the two-layer NN can implement the EXCLUSIVE OR (X-OR) and this again accelerated the NN research in the early 1980s. Several researchers (Hush and Horne 1993) presented solutions to the X-OR operation by using sigmoid activation functions.

The output of the two-layer NN is given by the recall equation

$$y_i = \sigma \left(\sum_{l=1}^L w_{il} \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right) + w_{i0} \right); \quad i = 1, 2, \dots, m \quad (1.12)$$

Defining the hidden-layer outputs z_l allows one to write

$$\begin{aligned} z_l &= \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right); \quad l = 1, 2, \dots, L \\ y_i &= \sigma \left(\sum_{l=1}^L w_{il} z_l + w_{i0} \right); \quad i = 1, 2, \dots, m \end{aligned} \quad (1.13)$$

Defining first-layer weight matrices \bar{V} and V as in the previous subsection, and second-layer weight matrices as

$$\bar{W}^T \equiv \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1L} \\ w_{21} & w_{22} & \cdots & w_{2L} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mL} \end{bmatrix}, \quad b_w = \begin{bmatrix} w_{10} \\ w_{20} \\ \vdots \\ w_{m0} \end{bmatrix} \quad (1.14)$$

$$W^T \equiv \begin{bmatrix} w_{10} & w_{11} & w_{12} & \cdots & w_{1L} \\ w_{20} & w_{21} & w_{22} & \cdots & w_{2L} \\ \vdots & \vdots & & & \vdots \\ w_{m0} & w_{m1} & w_{m2} & \cdots & w_{mL} \end{bmatrix} \quad (1.15)$$

one may write the NN output as,

$$y = \bar{\sigma}(\bar{W}^T \bar{\sigma}(\bar{V}^T \bar{x} + b_v) + b_w), \quad (1.16)$$

or, in streamlined form as

$$y = \bar{\sigma}(W^T \sigma(V^T x)). \quad (1.17)$$

In these equations, the notation $\bar{\sigma}$ means the vector is defined in accordance with (1.9). In (1.17) it is necessary to use the augmented vector

$$\sigma(w) \equiv [1 \quad \bar{\sigma}(w)^T]^T = [1 \quad \sigma(w_1) \quad \sigma(w_2) \quad \cdots \quad \sigma(w_L)]^T, \quad (1.18)$$

where a 1 is placed as the first entry to allow the incorporation of the thresholds w_{i0} as the first column of W^T . In terms of the hidden-layer output vector $z \in \Re^L$

one may write

$$\bar{z} = \sigma(V^T x), \quad (1.19)$$

$$y = \sigma(W^T z). \quad (1.20)$$

where $z \equiv [1 \ \bar{z}^T]^T$.

In the remainder of this book we shall not show the overbar on vectors — the reader will be able to determine by the context whether the leading 1 is required. We shall generally be concerned in later chapters with two-layer NN with linear activation functions in the output layer, so that

$$y(k) = W^T \sigma(V^T x(k)) \quad (1.21)$$

It is important to mention that the input-to-hidden-layer weights will be selected randomly and held fixed whereas the hidden-to-output-layer weights will be tuned. This will minimize the computational complexity associated with using NN in feedback control applications while ensuring that one can use NN in control.

Example 1.1.2 (Output Surface for Two-Layer NN): A two-layer NN with two inputs and one output (Lewis et al. 1999) is given by the equation $y = \bar{W}^T \bar{\sigma}(\bar{V}^T \bar{x} + b_v) + b_w \equiv w\sigma(vx + b_v) + b_w$, with weight matrices and thresholds given by

$$v = \bar{V}^T = \begin{bmatrix} -2.69 & -2.80 \\ -3.39 & -4.56 \end{bmatrix}, \quad b_v = \begin{bmatrix} -2.21 \\ 4.76 \end{bmatrix}$$

$$w = \bar{W}^T = \begin{bmatrix} -4.91 & 4.95 \end{bmatrix}, \quad b_w = [-2.28]$$

Plots of the NN output surface y as a function of the inputs x_1, x_2 over the grid $[-2, 2] \times [-2, 2]$ can be generated. Different outputs can be illustrated corresponding to the use of different activation functions. To make the plot in Figure 1.7 the MATLAB NN Toolbox 4.0 was used with the sequence of commands given in Example 1.1.2.

```
% Example 1.1.2: Output surface of two-layer NN

% Set up NN weights
v = [-2.69 -2.80; -3.39 -4.56];
bv = [-2.21; 4.76];
w = [-4.91 4.95];
bw = [-2.28];
```

```
% Set up plotting grid for sampling x
[x1,x2] = meshgrid(-2:0.1:2);

% Compute NN input vectors p and simulate NN using
% sigmoid
p1 = x1(:);
p2 = x2(:);
p = [p1'; p2'];

net = nnt2ff(minmax(p), {v,w}, {bv,bw}, {'hardlim',
    'purelin'});
a = sim(net,p);

% Format results for using 'mesh' or 'surf' plot
routines:
a1 = eye(41);
a1(:) = a';
mesh(x1,x2,a1);
AZ = 60, EL = 30;
view(AZ,EL);
xlabel('x1');
ylabel('x2');
%title('NN output surface using sigmoid');
title('NN output surface using hardlimit');
```

Plotting the NN output surface over a region of values for x reveals graphically the decision boundaries of the network and aids in visualization.

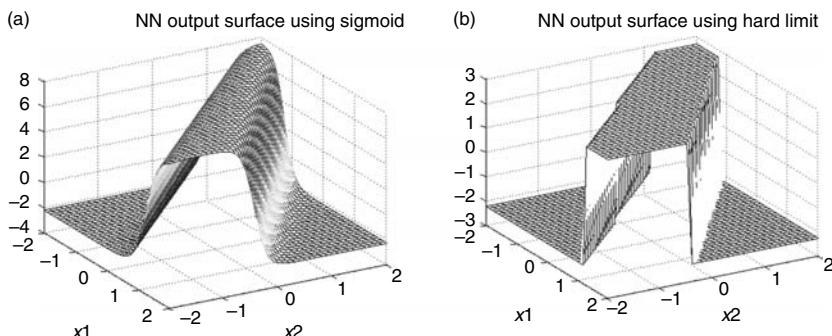


FIGURE 1.7 Output surface of a two-layer NN. (a) Using sigmoid activation function. (b) Using hard limit activation function.

1.1.3 LINEAR-IN-THE-PARAMETER NN

If the first-layer weights and the thresholds V in (1.21) are predetermined by some a priori method, then only the second-layer weights and thresholds W are considered to define the NN, so that the NN has only one layer of weights. One may then define the fixed function $\phi(x) = \sigma(V^T x)$ so that such a one-layer NN has the recall equation

$$y = W^T \phi(x), \quad (1.22)$$

where $x \in \Re^n$ (recall that technically x is augmented by 1), $y \in \Re^m$, $\phi(\cdot) : \Re \rightarrow \Re^L$, and L is the number of hidden-layer neurons. This NN is *linear* in the NN parameters W . This will make it easier for us to deal with such networks in subsequent chapters. Specifically, it is easier to train the NN by tuning the weights. This one-layer having only output-layer weights W should be contrasted with the one-layer NN discussed in (1.11), which had only input-layer weights V .

More generality is gained if $\sigma(\cdot)$ is not diagonal, for example, as defined in (1.9), but $\phi(\cdot)$ is allowed to be a general function from \Re^n to \Re^L . This is called a functional link neural net (FLNN) (Sadegh 1993). Some special FLNNs are now discussed. We often use $\sigma(\cdot)$ in place of $\phi(\cdot)$, with the understanding that, for linear in the parameter nets, this activation function vector is not diagonal, but is a general function from \Re^n to \Re^L .

1.1.3.1 Gaussian or Radial Basis Function Networks

The selection of a suitable set of activation functions is considerably simplified in various sorts of structured nonlinear networks, including radial basis functions (RBFs) and cerebellar model articulation controller (CMAC). It will be shown here that the key to the design of such structured nonlinear networks lies in a more general set of NN thresholds than allowed in the standard equation (1.12), and in the Gaussian or RBF (Sanner and Slotine 1991) given when x is a scalar

$$\sigma(x) = e^{-(x-\mu)^2/2p}, \quad (1.23)$$

where μ is the mean and p the variance. RBF NN can be written as (1.21), but have an advantage over the usual sigmoid NN in that the n -dimensional Gaussian function is well understood from probability theory, Kalman filtering, and elsewhere, making the n -dimensional RBF easier to conceptualize.

The j th activation function can be written as

$$\sigma_j(x) = e^{-1/2[(x-\mu_j)^T P_j^{-1}(x-\mu_j)]} \quad (1.24)$$

with $x, \mu_j \in \Re^n$. Define the vector of activation functions as $\sigma_j(x) \equiv [\sigma_1(x) \ \sigma_2(x) \ \dots \ \sigma_L(x)]^T$. If the covariance matrix is diagonal so that $P_j = \text{diag}\{p_{jk}\}$, then (1.24) becomes separable and may be decomposed into components as

$$\sigma_j(x) = e^{-1/2 \sum_{k=1}^n (x_k - \mu_{jk})^2 / P_{jk}} = \prod_{k=1}^n e^{-1/2[(x_k - \mu_{jk})^2 P_{jk}]}, \quad (1.25)$$

where x_j, μ_{jk} are the k th components of x, μ_j . Thus, the n -dimensional activation functions are the product of n scalar functions. Note that this equation is of the form of the activation functions in (1.12), but with more general thresholds, as a threshold is required for each different component of x at each hidden-layer neuron j ; that is, the threshold at each hidden-layer neuron in Figure 1.6 is a vector. The RBF variances p_{jk} are identical, and the offsets μ_{jk} are usually selected in designing the RBF NN and left fixed; only the output-layer weights W^T are generally tuned. Therefore, the RBF NN is a special sort of FLNN (1.22) (where $\phi(x) = \sigma(x)$).

Figure 1.8 shows separable Gaussians for the case $x \in \Re^n$. In this figure, all the variances p_{jk} are identical, and the mean values μ_{jk} are chosen in a special way that spaces the activation functions at the node points of a two-dimensional (2D) grid. To form an RBF NN that approximates functions over the region $\{-1 < x_1 \leq 1, -1 < x_2 \leq 1\}$, one has here selected $L = 5 \times 5 = 25$ hidden-layer neurons, corresponding to 5 cells along x_1 and 5 along x_2 . Nine of these neurons have 2D Gaussian activation functions, while those along the boundary require the illustrated one-sided activation functions.

The Gaussian means and variances can also be chosen randomly as an alternative to choosing them manually. In 2D, for instance (cf. Figure 1.8), this produces a set of L Gaussians scattered at random over the (x_1, x_2) plane with different variances.

The importance of RBF NN is that they show how to select the activation functions and the number of hidden-layer neurons for specific NN applications (e.g., function approximation — see below).

1.1.3.2 Cerebellar Model Articulation Controller Networks

A CMAC NN (Albus 1975) has separable activation functions generally composed of splines. The activation functions of a 2D CMAC composed of second-order splines (e.g., triangle functions) are shown in Figure 1.9, where $L = 5 \times 5 = 25$. The activation functions of a CMAC NN are called receptive field functions in analogy with the optical receptor fields of the eye.

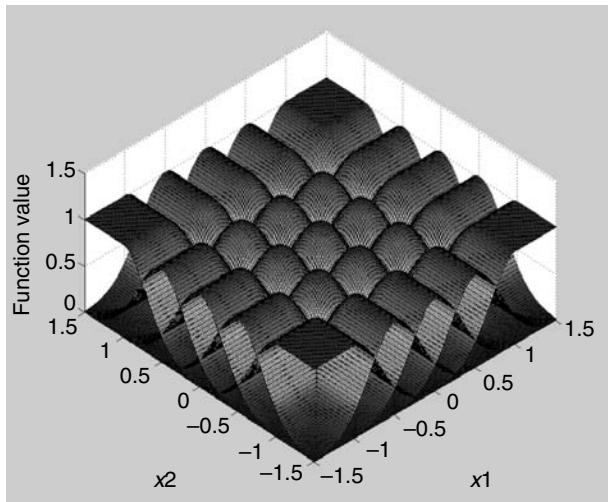


FIGURE 1.8 2D separable Gaussian functions for an RBF NN.

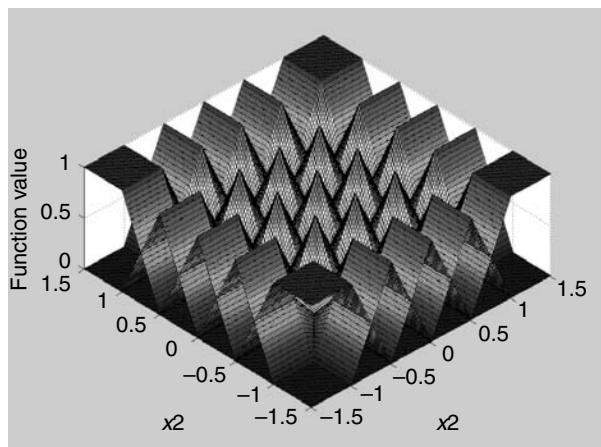


FIGURE 1.9 Receptive field functions for a 2D CMAC NN with second-order splines.

An advantage of CMAC NN is that the receptive field functions based on the splines have finite support so that they may be efficiently evaluated. An additional computational advantage is provided by the fact that higher-order splines may be computed recursively from lower-order splines.

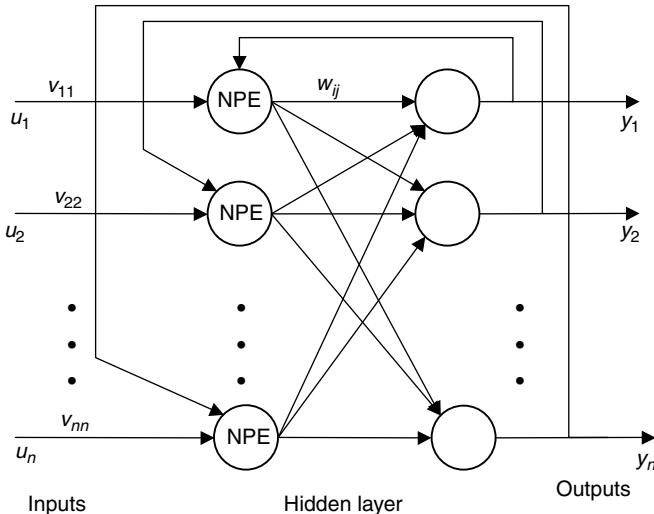


FIGURE 1.10 Hopfield dynamical neural net.

1.1.4 DYNAMIC NN

The NN that have been discussed so far contain no time-delay elements or integrators. Such NN are called *nondynamic* as they do not have any memory. There are many different dynamic NN, or recurrent NN, where some signals in the NN are either integrated or delayed and fed back into the network. The seminal work of Narendra and coworkers (see References) should be explored for more details.

1.1.4.1 Hopfield Network

Perhaps the most familiar dynamic NN is the Hopfield net, shown in Figure 1.10, a special form of two-layer NN where the output y_i is fed back into the hidden-layer neurons (Haykin 1994). In the Hopfield net, the first-layer weight matrix V is the identity matrix I , the second-layer weight matrix W is square, and the output-layer activation function is linear. Moreover, the hidden-layer neurons have increased processing power in the form of a memory. We may call such neurons with internal signal processing as neuronal processing elements (NPEs) (cf. Simpson 1992).

In the continuous-time case the internal dynamics of each hidden-layer NPE contains an integrator $1/s$ and a time constant τ_i in addition to the usual nonlinear activation function $\sigma(\cdot)$. The internal state of the NPE is described by the signal $x_i(t)$. The continuous-time Hopfield net is described by the ordinary

differential equation

$$\tau_i \dot{x}_i = -x_i + \sum_{j=1}^n w_{ij} \sigma_j(x_j) + u_i \quad (1.26)$$

with output equation

$$y_i = \sum_{j=1}^n w_{ij} \sigma_j(x_j) \quad (1.27)$$

This is a dynamical system of special form that contains the weights w_{ij} as adjustable parameters and positive time constants τ_i . The activation function has a subscript to allow, for instance, for scaling terms g_j as in $\sigma_j(x_j) \equiv \sigma(g_j x_j)$, which can significantly improve the performance of the Hopfield net. In the traditional Hopfield net the threshold offsets u_i are constant bias terms. It can be seen that (1.26) has the form of a state equation in control system theory, where the internal state is labeled as $x(t)$. It is for this reason that we have named the offsets as u_i . The biases play the role of the control input term, which is labeled as $u(t)$. In traditional Hopfield NN, the term “input pattern” refers to the initial state components $x_i(0)$.

In the discrete-time case, the internal dynamics of each hidden-layer NPE contains a time delay instead of an integrator, as shown in Figure 1.11. The NN is now described by the difference equation

$$x_i(k+1) = p_i x_i(k) + \sum_{j=1}^n w_{ij} \sigma_j(x_j(k)) + u_i(k) \quad (1.28)$$

with $|p_i| < 1$. This is a discrete-time dynamical system with time index k .

Defining the NN weight matrix W^T , vectors $x \equiv [x_1 \ x_2 \ x_3 \ \cdots \ x_n]^T$ and $u \equiv [u_1 \ u_2 \ u_3 \ \cdots \ u_n]^T$, and the matrices $\Gamma \equiv \text{diag}\{\frac{1}{\tau_1} \ \frac{1}{\tau_2} \ \cdots \ \frac{1}{\tau_n}\}^T$,

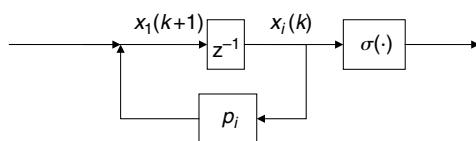


FIGURE 1.11 Discrete-time Hopfield hidden-layer processing neuron dynamics.

$P \equiv \text{diag}\{p_1, p_2, \dots, p_n\}^T$ with each $|p_i| < 1$; $i = 1, \dots, n$, one may write the discrete-time Hopfield network dynamics as

$$x(k+1) = P\Gamma x(k) + \Gamma W^T \sigma(x(k)) + \Gamma u(k) \quad (1.29)$$

(Note that technically some of these variables should have overbars. We shall generally drop the overbars henceforth.) A system theoretic block diagram of this dynamics is given in Figure 1.12.

Example 1.1.3 (Dynamics and Lyapunov Surface of Hopfield Network): Select $x = [x_1 \ x_2]^T \in \mathbb{R}^2$ and choose parameters so that the Hopfield net is

$$x(k+1) = -\frac{1}{2}x(k) + \frac{1}{2}W^T \sigma(x) + \frac{1}{2}u(k)$$

with weight matrix

$$W = W^T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Select the symmetric activation function in Figure 1.3 so that

$$\xi_i = \sigma_i(x_i) \equiv \sigma(g_i x_i) = \frac{1 - e^{-g_i x_i}}{1 + e^{-g_i x_i}}$$

Then,

$$x_i = \sigma_i^{-1}(\xi_i) = -\frac{1}{g_i} \ln \left(\frac{1 - \xi_i}{1 + \xi_i} \right)$$

Using sigmoid decay constants $g_1 = g_2 = 100$, these functions are plotted in Figure 1.13.

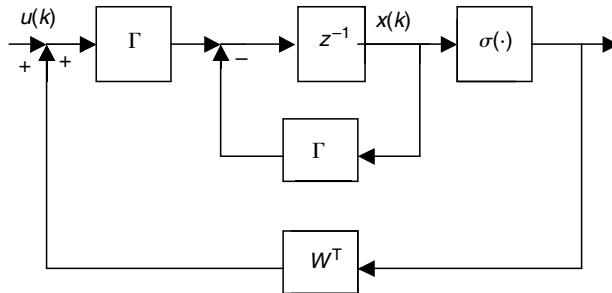


FIGURE 1.12 Discrete-time Hopfield network in block diagram form.

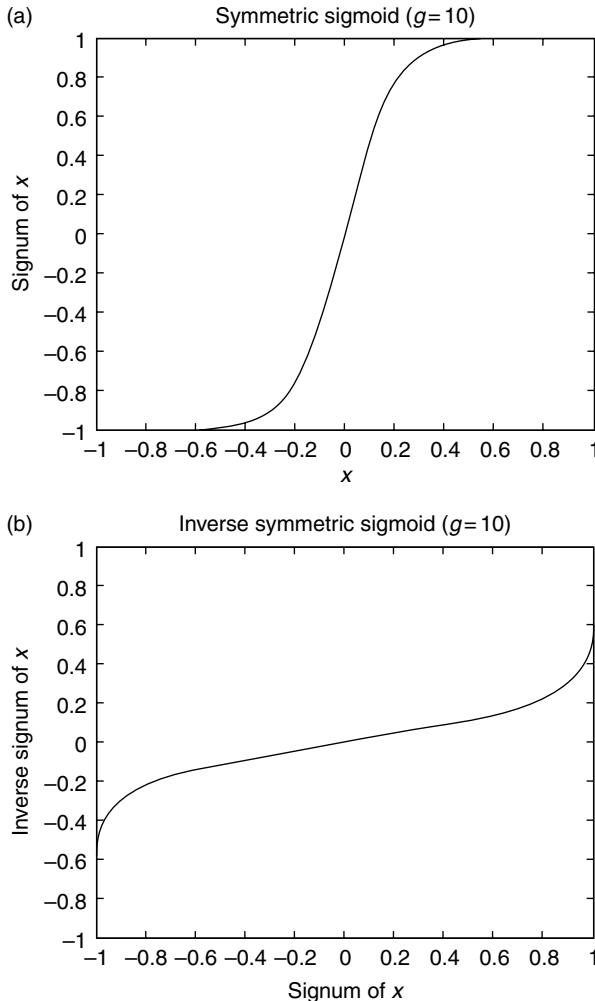


FIGURE 1.13 Hopfield net function. (a) Sigmoidal activation function. (b) Inverse of symmetric sigmoidal activation function.

State trajectory phase-plane plots: State trajectory phase-plane plots for various initial condition vectors $x(0)$ and $u = 0$ are shown in Figure 1.14, which plots $x_2(k)$ vs. $x_i(k)$. All initial conditions converge to the vicinity of either point $(-1, -1)$ or point $(1, 1)$. As seen in Section 1.3.1, these are the exemplar patterns stored in the weight matrix W . Techniques for selecting the weights for the desired performance are given in Section 1.3.1.

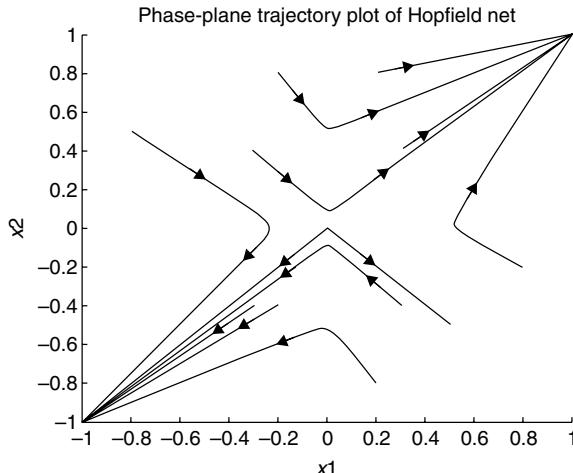


FIGURE 1.14 Hopfield net phase-plane plots $x_1(k)$ vs. $x_2(k)$.

The state trajectories are plotted with MATLAB, which requires the following M file to describe the system dynamics:

```
% hopfield.m : Matlab M file for Hopfield net dynamics
function xdot = hopfield(t,x)
g=100;
tau=2;
u=[0 ; 0];
w = [0 1; 1 0];
xi = (1-exp(-g*x))./(1+exp(-g*x));
xk+1 = (-x+w*v+u)/tau;
```

In MATLAB an operator preceded by a period denotes the element-by-element matrix operation; thus $./$ denotes element-by-element vector division.

1.1.4.2 Generalized Recurrent NN

A generalized dynamical system is shown in Figure 1.15 (cf. work of Narendra, see References). In this figure, $H(z) = C(zI - A)^{-1}B$ represents the transfer function of linear dynamical system or plant given by

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y &= Cx \end{aligned} \tag{1.30}$$

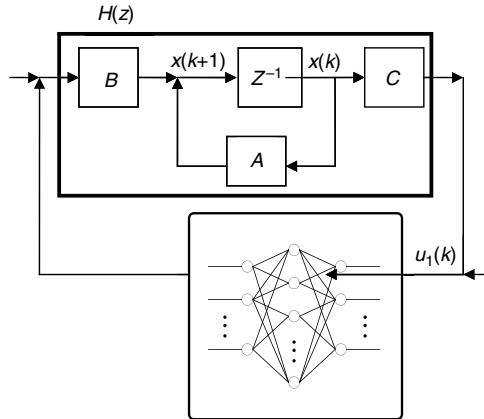


FIGURE 1.15 Generalized discrete-time dynamical NN.

with internal state $x(k) \in \Re^n$, control input $u(k)$, and output $y(k)$. The NN can be a two-layer net described by (1.16) and (1.17). This dynamic NN is described by the equation

$$x(k+1) = Ax(k) + B[\sigma(W^T\sigma(V^T(Cx + u_1)))] + Bu_2 \quad (1.31)$$

From examination of (1.28) it is plain to see that the Hopfield net is a special case of this equation, which is also true of many other dynamical NN in the literature. A similar version holds for the continuous-time case. If the system matrices A , B , and C are diagonal, then the dynamics can be interpreted as residing with the neurons, and one can speak of NPEs with increased computing power and internal memory. Otherwise, there are additional dynamical interconnections around the NN as a whole.

Example 1.1.4 (Chaotic Behavior of NN): This example is taken from Lewis et al. (1999) as an outcome of a discussion between Professor Abdallah (1995) and Lewis in Becker and Dörfler (1988). Even in simple NN it is possible to observe some very interesting behavior, including limit cycles and chaos. Consider for instance the discrete Hopfield NN with two inputs, two states, and two outputs is given by $x(k+1) = Ax(k) + W^T\sigma(V^Tx(k)) + u(k)$, which is the discrete-time form of (1.31).

a. *Starfish attractor — changing the NN weights:* Select the system matrices as

$$A = \begin{bmatrix} -0.1 & 1 \\ -1 & 0.1 \end{bmatrix} \quad w = W^T = \begin{bmatrix} \pi & 1 \\ 1 & -1 \end{bmatrix} \quad v = V^T = \begin{bmatrix} 1.23456 & 2.23456 \\ 1.23456 & 2.23456 \end{bmatrix}$$

and the input as $u_k = [1 \quad 1]^T$.

It is straightforward to simulate the time performance Hopfield system, using the following MATLAB code:

```
% Matlab function file for
simulation of discrete
Hopfield NN function [x,y]=starfish(n)
x1(1)=-rand;
x2(1)=rand;
a11=-0.1;a12=1;
a21=-1;a22=0.1;
w11 = pi;w12 = 1;
w21 = 1; w22 = -1;
u1 = 1;
u2 = -1;
v11 = 1.23456; v12 = 2.23456;
v21 = 1.23456; v22 = 2.23456;
for k=1:N
x1(k+1)=a11*x1(k)+a12*x2(k)+w11*tanh(v11*x1(k))
+w12*tanh(v12*x2(k))+u1;
x2(k+1)=a21*x1(k)+a22*x2(k)+w21*tanh(v11*x1(k))
+w22*tanh(v22*x2(k))+u1;
end
end
```

where the argument N is the number of times the iterations have to be performed. The system is initialized at a random initial state x_0 . The tanh activation function is used.

The result of the simulation is plotted using the MATLAB function plot (x_1 , x_2 , "."); it is shown for $N = 2000$ points in Figure 1.16. The time history is attracted into the shape of a starfish after an initial transient. The dimensions of the attractor can be determined by using Lyapunov's exponent analysis. If the attractor has a noninteger dimension, it is called a strange attractor and the NN exhibits chaos.

Changing the NN weight matrices results in a different behavior. Setting

$$v = V^T = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$$

yields the plot shown in Figure 1.17. It is very easy to destroy the chaotic behavior. For instance, setting

$$v = V^T = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

yields the plot shown in Figure 1.18, where the attractor is a stable limit-cycle.

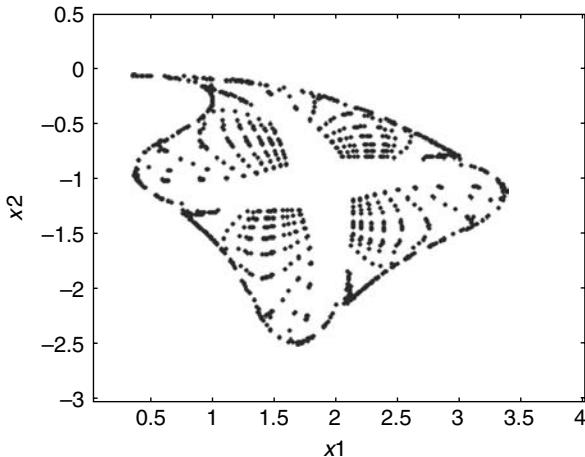


FIGURE 1.16 Phase-plane plot of discrete-time NN showing attractor.

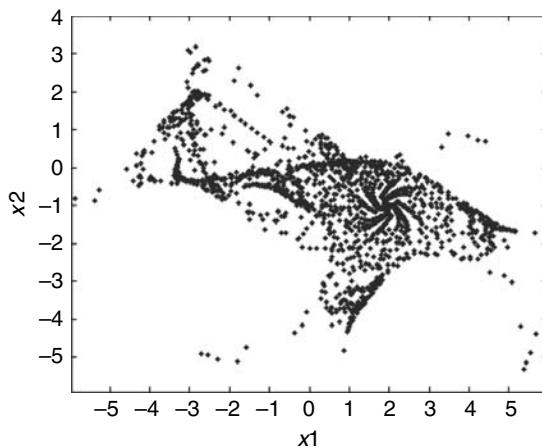


FIGURE 1.17 Phase-plane plot of discrete-time NN with modified weight matrix V .

b. *Anemone attractor — changing the plant A matrix:* Changes in the plant matrices (A, B, C) also influence the characteristics of the attractor. Setting

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 0.1 \end{bmatrix}$$

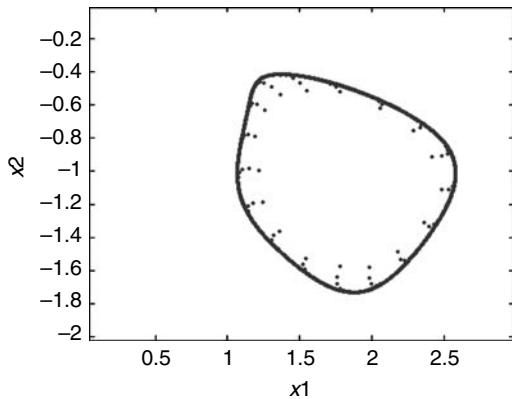


FIGURE 1.18 Phase-plane plot of discrete-time NN with modified weight matrix V showing limit-cycle attractor.

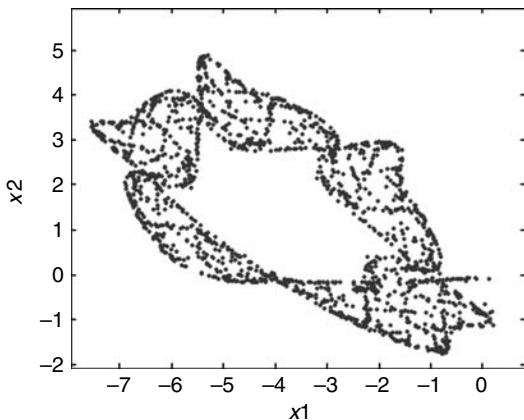


FIGURE 1.19 Phase-plane plot of discrete-time NN with modified A matrix.

yields the phase-plane plot shown in Figure 1.19. Also changing the NN first-layer weight matrix to

$$v = V^T = \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix}$$

yields the behavior shown in Figure 1.20.

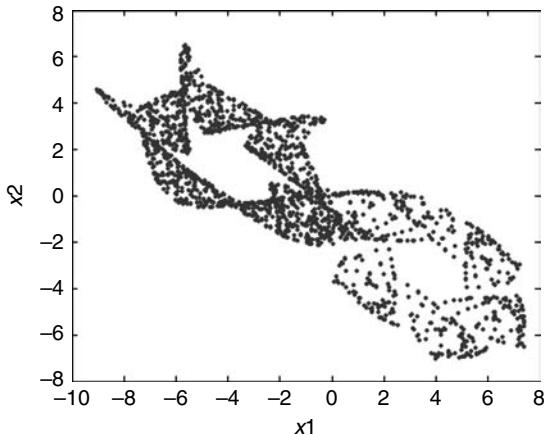


FIGURE 1.20 Phase-plane plot of discrete-time NN with modified A and V matrices.

1.2 PROPERTIES OF NN

Neural networks are complex nonlinear distributed systems, and as a result they have a broad range of applications. Many of the remarkable properties of NN are a result of their origins from biological information processing cells. In this section we discuss two properties: classification (for pattern recognition see other books in references), and function approximation. These are both open-loop applications, in that the NN is not required to control a dynamical system, in a feedback loop. However, we shall see in subsequent chapters that for closed-loop feedback control purposes the function approximation property in particular is a key requirement capability.

There are two issues that should be understood clearly. On one hand, NN are complex with some important properties and capabilities. However, to function as desired, suitable weights of the NN must be determined. Thus, on the other hand, there are effective algorithms to compute or tune the weights by training the NN in such a manner that, when training is complete, it exhibits the desired properties as originally planned for. Thus, in Section 1.3 we discuss techniques of weight selection and tuning so that the NN performs as a classifier and a function approximator.

It is important to note that, though it is possible to construct NN with multiple hidden layers, the computational burden increases with the number of hidden layers. An NN with two hidden layers (three-layer network) can form the most complex decision regions for classification. However, in many practical situations it is usually found that the two-layer NN (e.g., with one hidden layer) is sufficient. Specifically, since two-layer NN are the simplest to

have the function approximation capability, they are sufficient for all the control applications discussed in this book.

1.2.1 CLASSIFICATION AND ASSOCIATION

In DSP, NN have been extensively used as pattern recognizers, classifiers, and contrast enhancers (Lippmann 1987). In all these applications the fundamental issue is distinguishing between different inputs presented to the NN; usually the input is a constant time-invariant vector, often binary (consisting of 1s and 0s) or bipolar (having entries of, e.g., ± 1). The NN in such uses is known as a content-addressable associative memory, which associates various input patterns with the closest of a set of exemplar patterns (e.g., identifying noisy letters of the alphabet).

1.2.1.1 Classification

Recall a one-layer NN with two inputs x_1, x_2 and one output is given by

$$y = \sigma(v_0 + v_1x_1 + v_2x_2) \quad (1.32)$$

where in this application $\sigma(\cdot)$ is the symmetric hard limiter in Figure 1.3. The output can take on values of ± 1 . When y is zero, there holds the relation

$$\begin{aligned} 0 &= v_0 + v_1x_1 + v_2x_2 \\ x_2 &= -\frac{v_1}{v_2}x_1 - \frac{v_0}{v_2} \end{aligned} \quad (1.33)$$

As illustrated in Figure 1.21, this is a line partitioning \Re^2 into two decision regions, with y taking a value of $+1$ in one region and -1 in the other. Therefore, if the input vectors $x = [x_1 \ x_2]^T$ take on values as shown by the *As* and *Bs*, they can be partitioned into the two classes *A* and *B* by examining the values of y resulting when the values of x are presented to the NN.

Given the two regions into which the values of x should be classified, it is necessary to know how to select weights and thresholds to draw a line between the two regions. Weight selection and NN training are discussed in Section 1.3.

In the general case of n inputs x_j and L outputs y_l , the one-layer NN output (shown in Figure 1.4) values of x do not fall into regions that are separable using hyperplanes; this implies that they cannot be classified using a one-layer NN (see Lippmann 1987).

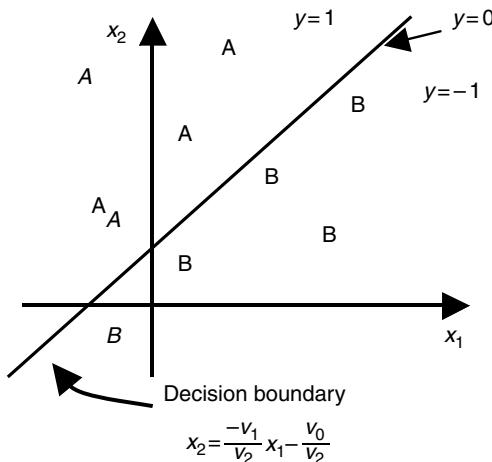
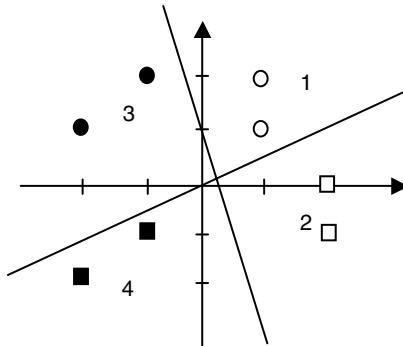


FIGURE 1.21 Decision region of a simple one-layer NN.

The two-layer NN with n inputs, L hidden-layer neurons, and m outputs (Figure 1.6) can implement more complex decisions than the one-layer NN. Specifically, the first layer forms L hyperplanes (each dimension $n - 1$), and the second layer combines them into m decision regions by taking various intersections of the regions defined by the hyperplanes, depending on the output-layer weights. Thus, the two-layer NN can form open or closed convex decisions regions (see Lippmann 1987). The X-OR problem can be solved by using a two-layer NN. The three-layer NN can form arbitrary decision regions, not necessarily convex, and suffices for the most complex classification problems. Smooth decision boundaries can be obtained by using smooth activation functions. This discussion has assumed hard limit activation function. With smooth activation functions, moreover, the backpropagation training algorithm given in the next section or those developed in this book and elsewhere can be used to determine the weights needed to solve any specific classification problem.

The NN structure should be complex enough for the decision problem at hand; too complex a network makes for additional computation time that is not necessary. The number of nodes in the hidden layer should typically be sufficient to provide three or more edges for each decision region generated by the output-layer nodes. Arbitrarily increasing the number of nodes and layers does not always lead to an improvement in the results as it can cause a NN to memorize the mapping instead of generalizing it which is not considered satisfactory.

**FIGURE 1.22** Probable decision boundaries.

Example 1.2.1 (Simple Four Class Decision Problem): A simple perceptron has to be trained to classify an input vector into four classes. The four classes are

$$\text{class 1: } \left\{ p_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, p_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\} \quad \text{class 2: } \left\{ p_3 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, p_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$$

$$\text{class 3: } \left\{ p_5 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, p_6 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\} \quad \text{class 4: } \left\{ p_7 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, p_8 = \begin{bmatrix} -2 \\ -2 \end{bmatrix} \right\}$$

A perceptron with s neurons can categorize 2^s classes. Thus to solve this problem, a perceptron with at least two neurons is needed. A two-neuron perceptron creates two decision boundaries. Therefore, to divide the input space into the four categories, we need to have one decision boundary divide the four classes into two sets of two. The remaining boundary must then isolate each class. Two such boundaries are illustrated in Figure 1.22 showing that our patterns are in fact linearly separable.

The target vectors for each of the classes are chosen as

$$\text{class 1: } \left\{ t_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\} \quad \text{class 2: } \left\{ t_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

$$\text{class 3: } \left\{ t_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad \text{class 4: } \left\{ t_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$$

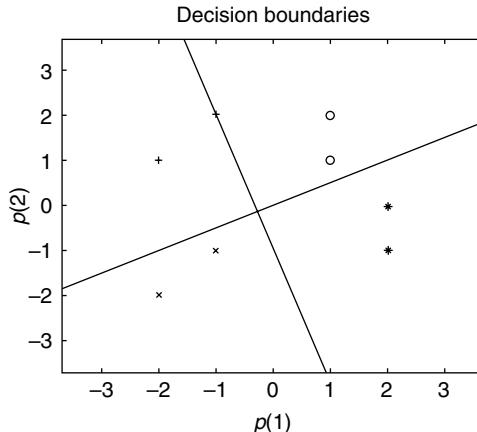


FIGURE 1.23 Final decision boundaries.

We can create this perceptron using the NEWP function in the MATLAB toolbox. The following sequence of commands is used to generate the classification boundaries:

```
p=[1 1 2 2 -1 -2 -1 -2;1 2 -1 0 2 1 -1 -2];
t=[0 0 0 0 1 1 1 1;0 0 1 1 0 0 1 1];
net=newp([-2 2;-2 2],2);
net=train(net,p,t);
v = net.iw{1,1},
b = net.b{1}
```

There can be several possible answers to this question. Figure 1.23 shows the final decision boundaries obtained using the toolbox. As we can see the generated boundaries are not as good as the probable decision boundaries in Figure 1.22. This is due to random initialization of the weights and biases. A similar but more complex classification problem will be later taken up in Section 1.3.

1.2.1.2 Association

In the association problem there are prescribed input vectors $X^P \in \Re^m$, each of which is to be associated with its corresponding X^P . In practical situations there might be multiple input vectors prescribed by the user, each with an associated desired output vector. Thus, there might be prescribed P , known as *exemplar* input/output pairs $(X^1, Y^1), (X^2, Y^2), \dots, (X^P, Y^P)$ for the NN.

Pattern recognition is often a special case of association. In illustration, $X^P, p = 1, \dots, 26$ could be the letters of the alphabet drawn on a 7×5 grid of 1s and 0s (e.g., 0 means light, 1 means dark), and Y^1 could be A , Y^2 could be B , and so on. For presentation to the NN as vectors, X^P might be encoded as the columns of the 7×5 grid stacked on top of one another to produce a 35-vector of 1s and 0s, while Y^P might be the p th column of the 26×26 identity matrix, so that A would be encoded as $[1 \ 0 \ 0 \ \dots]^T$ and so on. Then, the NN should associate pattern X^P with target output Y^P to classify the letters.

Selection of correct weights and thresholds for the NN is very important for solving pattern recognition and association problems for a given set of input/output pairs (X^P, Y^P) . This is illustrated in the next example.

Example 1.2.2 (NN Weights and Biases for Pattern Association): It is desired to design a one-layer NN with one input x and one output y that associates the input $X^1 = -3$ with the target output $Y^1 = 0.4$ and input $X^2 = 2$ with the target output $Y^2 = 0.8$. Thus the desired input/output pairs to be associated are $(-3, 0.4)$, $(2, 0.8)$. The NN has only one weight and one bias and the recall equation is

$$y = \sigma(vx + b)$$

Denote the actual NN outputs when the input exemplar patterns are presented as

$$y^1 = \sigma(vX^1 + b) \quad y^2 = \sigma(vX^2 + b)$$

when the NN is performing as prescribed, one should have $y^1 = Y^1, y^2 = Y^2$. To measure the performance of the NN, define the least-squares output error as

$$E = \frac{1}{2}[(Y^1 - y^1)^2 + (Y^2 - y^2)^2]$$

when E is small, the NN is performing well.

Using MATLAB NN Toolbox 4.0, it is straightforward to plot the least-squares output error E as a function of the weights w and the bias b . The result is shown in Figure 1.24 for the sigmoid and the hard limit activation functions.

To design the NN, it is necessary to select w and b to minimize the error E . It is seen that for the hard limit, E is minimized for a range of weight/bias values. On the other hand, for the sigmoid function the error is minimized for (w, b) in the vicinity of $(0.3, 0.6)$. Moreover, the sigmoid allows a smaller minimum value of E than does the hard limit. Since the error surface plot using the sigmoid is smooth, conventional gradient-based techniques can be used to determine the optimal weight and bias. This topic is discussed in Section 1.3.2 for the one-layer NN and Section 1.3.3 for the multilayer NN.

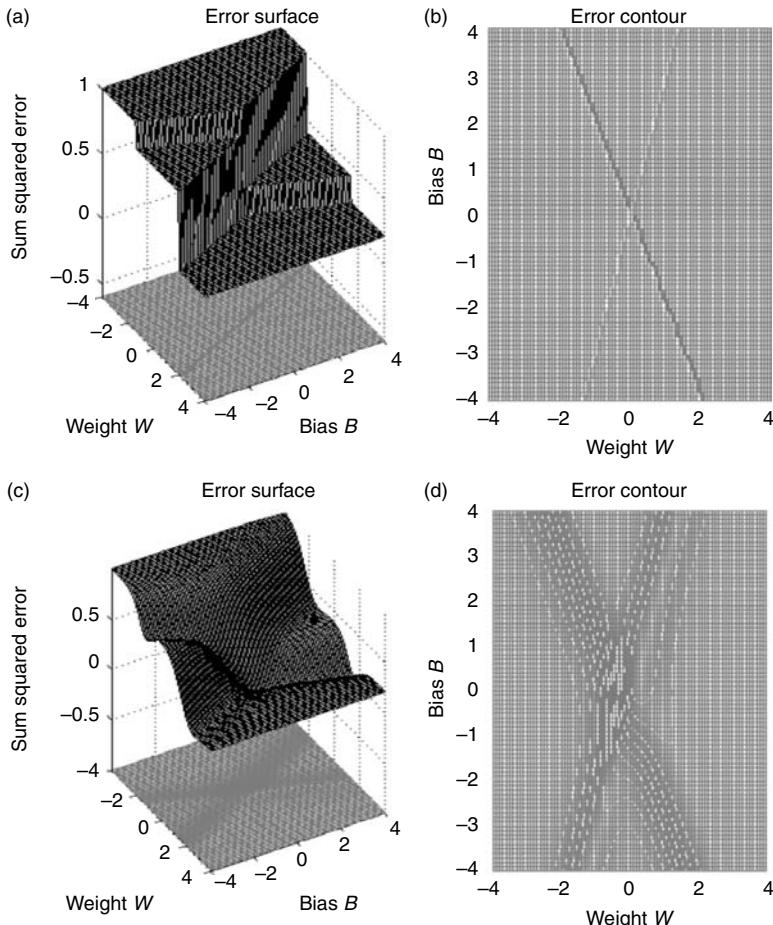


FIGURE 1.24 Output error plots vs. weights for a neuron. (a) Error surface using hard limit activation function. (b) Error contour plot using hard limit activation function. (c) Error surface using sigmoid activation function. (d) Error contour plot using sigmoid activation function.

To make, for instance, Figure 1.24a, the following MATLAB commands were used:

```
% set up input patterns, target outputs, and weight/bias ranges:  
p = [-3 2];  
t = [0.4 0.8];
```

```
wv = -4:0.1:4;
bv = -4:0.1:4;
% compute the output error surface:
es = errsurf(p,t,wv,bv,'logsig');
% plot and label error surface
mesh(wv, bv, es)
view(60,30)
set (gca,' xlabel, text(0,0,'weight'))
set (gca,' ylabel, text(0,0,'bias'))
title ('Error surface plot using sigmoid')
```

Note that the input patterns are stored in a vector p and the target outputs are stored in a vector t with corresponding entries.

1.2.2 FUNCTION APPROXIMATION

Of fundamental importance in NN closed-loop control applications is the universal function approximation property of NN having at least two layers. (One-layer NNs do not generally have a universal approximation capability.) The approximation capabilities of NN have been studied by many researchers including Cybenko (1989), Hornik et al. (1989), and Sandberg and coworkers (e.g., Park and Sandberg 1991).

The basic universal approximation result says that any smooth function $f(x)$ can be approximated arbitrarily closely on a compact set using a two-layer NN with appropriate weights. This result has been shown using sigmoid activations, hardlimit activations, and others. Specifically, let $f(x) : \Re^n \rightarrow \Re^m$ be a smooth function. Then given a compact set $S \in \Re^n$ and a positive number ε_N , there exists a two-layer NN (1.21) such that

$$f(x) = W^T \sigma(V^T x) + \varepsilon \quad (1.34)$$

with $\|\varepsilon\| = \varepsilon_N$ for all $x \in S$, for some sufficiently large value L of hidden-layer neurons. The value ε (generally a function of x) is called the NN function approximation error, and it decreases as the hidden-layer size L increases. We say that, on the compact set S , as S becomes larger, the required L generally increases correspondingly. Approximation results have also been shown for smooth functions with a finite number of discontinuities.

Even though the results says that there exists an NN that approximates $f(x)$, it should be noted that it does not show how to determine the required weights. It is in fact not an easy task to determine the weights so that an NN does indeed approximate a given function $f(x)$ closely enough. In the next section we shall show how to accomplish this using backpropagation tuning.

If the function approximation is to be carried out in the context of a dynamic closed-loop feedback control scheme, the issue is thornier and is solved in subsequent chapters. An illustration of NN function approximation is given in Example 1.3.3.

Functional-Link NN: In Section 1.1.3 was discussed a special class of one-layer of NN known as FLNN written as

$$y = W^T \phi(x) \quad (1.35)$$

with W the NN output weights (including the thresholds) and $\phi(\cdot)$ a general function from \Re^n to \Re^L . In subsequent chapters, these NN have a great advantage in that they are easier to train than general two-layer NN since they are LIPs. Unfortunately, for LIP NN, the functional approximation property does not generally hold. However, a FLNN can still approximate functions as long as the activation functions $\phi(\cdot)$ are selected as a basis, which must satisfy the following two requirements on a compact simply connected set S of \Re^n (Sadegh 1993):

1. A constant function on S can be expressed as (1.35) for a finite number L of hidden-layer functions
2. The functional range of (1.35) is dense in the space of continuous functions from S to \Re^m for a countable L

If $\phi(\cdot)$ provides a basis, then a smooth function $f(x) : \Re^n \rightarrow \Re^m$ can be approximated on a compact set S of \Re^n , by

$$f(x) = W^T \phi(x) + \varepsilon \quad (1.36)$$

for some ideal weights and thresholds W and some number of hidden-layer neurons L . In fact, for any choice of a positive number ε_N , one can find a feedforward NN such that $\|\varepsilon\| < \varepsilon_N$ for all x in S .

Barron (1993) has shown that for all LIP approximators there is a fundamental lower bound, so that ε is bounded below by terms on the order of $1/L^{2/n}$. Thus, as the number of NN inputs “ n ” increases, increasing L to improve the approximation accuracy becomes less effective. This lower bound problem does not occur in the multilayer nonlinear-in-the-parameters network.

Approximation property of n -layer NN: For a suitable approximation of unknown nonlinear functions, several NN architectures are currently available. In Cybenko (1989), it is shown that a continuous function $f(x(k)) \in C(S)$,

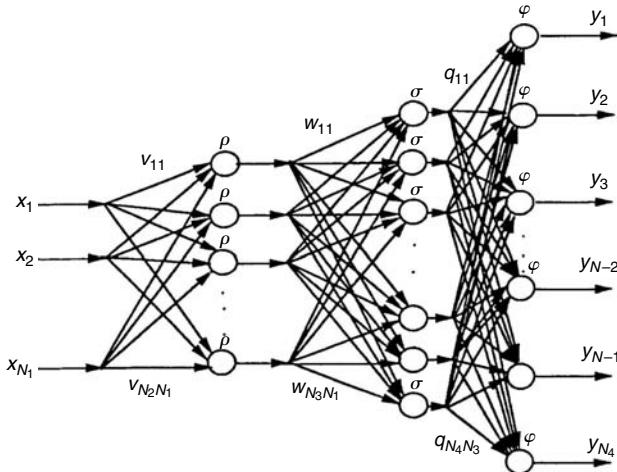


FIGURE 1.25 A multilayer neural network.

within a compact subset S of \Re^n , can be approximated using a n -layer feedforward NN, shown in Figure 1.25 as

$$f(x(k)) = W_n^T \phi [W_{n-1}^T \phi(\dots \phi(x(k)))] + \varepsilon(x(k)) \quad (1.37)$$

where $W_n, W_{n-1}, \dots, W_2, W_1$ are target weights of the hidden-to-output- and input-to-hidden-layers, respectively, $\phi(\cdot)$ denotes the vector of activation functions (usually, they are chosen as sigmoidal functions) at the instant k , $x(k)$ is the input vector, and $\varepsilon(x(k))$ is the NN functional reconstruction error vector. The actual NN output is defined as

$$\hat{f}(x(k)) = \hat{W}_n^T(k) \hat{\phi}_n(k) \quad (1.38)$$

where $\hat{W}_n(k)$ is the actual output-layer weight matrix. For simplicity, $\phi(\hat{W}_{n-1}^T \phi_{n-1}(\cdot))$ is denoted as $\hat{\phi}_n(k)$. Here ‘ N ’ stands for number of nodes at a given layer.

If there exists N_2 and constant ideal weights $W_n, W_{n-1}, \dots, W_2, W_1$ such that $\varepsilon(x(k)) = 0$ for all $x \in S$, then $f(x)$ is said to be in the functional range of the NN. In general given a real number, $\varepsilon_N \geq 0$, $f(x)$ is within ε_N of the NN range if there exists N_2 and constant weights so that for all x of \Re^n , (1.37) holds with $\|\varepsilon(x(k))\| \leq \varepsilon_N$ where $\|\cdot\|$ is a suitable norm (see Chapter 2). Moreover, if the number of hidden-layer nodes is sufficiently large, the reconstruction error $\varepsilon(x(k))$ can be made arbitrarily small on the compact set so that the bound $\|\varepsilon(x(k))\| \leq \varepsilon_N$ holds for all $x(k) \in S$.

Random Vector Functional Link Networks: The difficult task of selecting the activation functions in LIP NN so that they provide a basis is addressed by selecting the matrix V in (1.21) randomly. It is shown in Igelnik and Pao (1995) that, for these random vector functional link (RVFL) nets, the resulting function $\phi(x) = \sigma(V^T x)$ is a basis, so that the RVFL NN has the universal approximation property.

In this approach, $\sigma(\cdot)$ can be standard sigmoid functions. This approach amounts to randomly selecting the activation function scaling parameters v_{lj} and shift parameters v_{l0} in $\sigma(\sum_j v_{lj} x_j + v_{l0})$. This produces a family of L activation functions with different scaling and shifts (Kim 1996).

Number of hidden-layer neurons: The problem of determining the number of hidden-layer neurons for general fully reconnected NN (1.21) for good enough approximation has not been solved. However, for NN such as RBF or CMAC there is sufficient structure to allow a solution to this problem. The key hinges on selecting the activation functions close enough together in situations like Figure 1.9 and Figure 1.10. One solution is as follows:

Let $x \in \mathbb{R}^n$ and define uniform partitions in each component x_j . Let δ_j be the partition interval for x_j and $\delta \equiv \sqrt{\sum_{j=1}^n \delta_j^2}$. In illustration, in Figure 1.10 where $n = 2$, one has $\delta_1 = \delta_2 = 0.5$. The next result shows the maximum partition size δ allowed for approximation with a desired accuracy ε (Commuri 1996).

Theorem 1.2.1 (Partition Interval for CMAC Approximation): (Commuri 1996) Let a function $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be continuous with Lipschitz constant λ so that $\|f(x) - f(z)\| \leq \|x - z\|$ for all x, z in some compact set S of \mathbb{R}^n . Construct a CMAC with triangular receptive field functions $\phi(\cdot)$ in the recall equation (1.35). Then there exist weights W such that

$$\|f(x) - y(x)\| \leq \varepsilon$$

for all $x \in S$ if the CMAC is designed so that

$$\delta \leq \frac{\varepsilon}{m\lambda} \quad (1.39)$$

In fact, CMAC designed with this partition interval can approximate on S any continuous function smooth enough to satisfy the Lipschitz condition for the given λ . Given limits on the dimensions of S , one can translate this upper bound on δ to a lower bound on the number L of the hidden-layer neurons. Note that as the function $f(x)$ becomes less smooth, λ increases and the grid

nodes become more finely spaced resulting in an increase in the number of hidden-layer neurons L .

In Sanner and Slotine (1991) is given a similar result for designing RBF which selects the fineness of the grid partition based on a frequency domain smoothness measure for $f(x)$ instead of a Lipschitz constant smoothness measure.

1.3 NN WEIGHT SELECTION AND TRAINING

We have studied the topology of NN, and shown that they possess important properties including classification and function approximation capabilities. For an NN to function as desired, however, it is necessary to determine suitable weights and thresholds that ensure performance as desirable. For years this was a problem, especially for multilayer NN, where it was not known how to apportion the resulting errors to different layers and force the appropriate weights to change and reduce the errors — this was known as the error credit assignment problem. Today, these problems have for the most part been solved and there are very good algorithms for NN weight selection or tuning or both. References for this section include Haykin (1994), Kung (1993), Peretto (1992), and Hush and Horne (1993).

Direct computation vs. training: There are two basic approaches to determining NN weights: direct analytic computation and NN training by recursive update techniques. In the Hopfield net, for instance, the weights can be directly computed in terms of the desired outputs of the NN. In many other applications of static NN, the weights are tuned by a recursive NN training procedure. This chapter talks only of NN in open-loop applications. That is, not until later chapters do we get into the issues of tuning the NN weights while the NN is simultaneously performing in the capacity of a feedback controller to stabilize a dynamical plant.

Classification of learning schemes: Updating the weights by training the NN is known as the learning feature of NN. Learning algorithms may be carried out in continuous-time (via differential equations for the weights), or in discrete-time (via difference equations for the weights). There are many learning algorithms, and they fall into three categories. In supervised learning, the information needed for training is available *a priori*, for instance, the inputs x and the desired outputs y they should produce. This global information does not change, and is used to compute errors that can be used for updating the weights. It is said that there is a teacher that knows the desired outcomes and tunes the weights accordingly. On the other hand, in unsupervised learning (also called self-organizing behavior) the desired NN output is not known, so there is no

teacher. Instead, local data is examined and organized according to emergent collective properties. Finally, in reinforcement learning, the weights associated with a particular neuron are not changed proportionally to the output error of that neuron, but instead are changed in proportion to some reinforcement signal.

Learning and operational phases: There is a distinction between the learning phases, when the NN weights are selected (often through training), and the operational phase, when the weights are generally held constant and the inputs are presented to the NN as it performs its design function. During training the weights are often selected using prescribed inputs and outputs for the NN. In the operational phase, it is often the case that the inputs do not belong to the training set. However, in classification, for instance, the NN is able to provide the output corresponding to the exemplar to which any input is closest in some specified norm (e.g., a noisy A should be classified as an A). This ability to process inputs not necessarily in the exemplar set and provide meaningful outputs is known as the generalization property of the NN, and is closely connected to the property of associative memories that close inputs should provide close outputs.

Off-line vs. online learning: Finally, learning maybe off-line, where the preliminary and explicit learning phase occurs prior to applying the NN in its operational capacity (during which the weights are held constant), or online, where the NN functions in its intended operational capacity while simultaneously learning the weights. Off-line learning is widely used in open-loop applications such as classification and pattern recognition. By contrast, online learning is a very difficult problem, and is exemplified by closed-loop feedback control applications. There, the NN must keep a dynamical plant stable while simultaneously learning and ensuring that its own internal state (the weights) remains bounded. Various techniques from adaptive control theory are needed to successfully confront this problem. Chapter 2 describes a standard adaptive controller design in discrete-time before the NN development presentation in subsequent chapters.

1.3.1 WEIGHT COMPUTATION

In the Hopfield net, the weights can be initialized by direct computation of outer products between the desired outputs. The discrete-time Hopfield network has dynamics

$$\tau_i x_i(k+1) = p_i x_i(k+1) + \sum_{j=1}^n w_{ij} \sigma_j(x_j) + u_i(k) \quad (1.40)$$

or

$$x(k+1) = P\Gamma x(k) + \Gamma W^T \sigma(x) + \Gamma u(k) \quad (1.41)$$

with $x \in \Re^n$, $|p_i| < 1$, and $P = \text{diag}\{p_1, p_2, \dots, p_n\}$.

Suppose it is desired to design a Hopfield net that can discriminate between P prescribed bipolar pattern vectors X^1, X^2, \dots, X^P , for example, each having n entries of either +1 or -1. This requires the Hopfield net to act as an associative memory that discriminates among bipolar vectors, matching each input vector $x(0)$ presented as an initial condition with one of the P exemplar patterns X^P . It was shown by Hopfield that weights solving this problem may be selected using the Hebbian philosophy of learning as the outer products of the exemplar vectors.

$$W = \frac{1}{n} \sum_{p=1}^P X^P (X^P)^T - \frac{1}{n} PI \quad (1.42)$$

where I is the identity matrix. The purpose of the term PI is to zero out the diagonal. Note that this weight matrix W is symmetric. This formula effectively encodes the exemplar patterns in the weights of the NN and is technically an example of supervised learning. This is because the desired outputs are used to compute the weights, even though there is no explicit tuning of weights.

It can be shown that, with these weights, there are P equilibrium points in \Re^n , one at each of the exemplar vectors X^P (for instance, Hush and Horne 1993, Haykin 1994). Once the weights have been computed (the training phase), the net can be used in its operational phase, where an unknown vector $x(0)$ is presented as an initial condition, and the net state is computed as a function of time using (1.41). The net will converge to the equilibrium point X^P to which the input vector $x(0)$ is closest. (If the symmetric hard limit activation functions are used, the closest vector is defined in terms of the Hamming distance.) It is intriguing to note that the information is stored in the net using (1.41) (during the operational and the training phase). Thus, the NN functions as a biologically inspired memory device.

It can be shown that, with n as the size of the Hopfield net, one can obtain perfect recall if the number of stored exemplar patterns satisfies $P \leq n/(4 \ln n)$. For example, if there are 256 neurons in the net, then the maximum number of exemplar patterns allowed is $P = 12$. However, if a small fraction of the bits in the recalled pattern are allowed to be in error, then the capacity increases to $P \leq 0.138n$. If $P = 0.138n$ then approximately 1.6% of the bits in the recalled pattern are in error. Other weight selection techniques allow improved storage

capacity in the Hopfield net, in fact, with proper computation of W the net capacity can approach $P = n$.

Example 1.3.1 (Hopfield Net Weight Selection): In Example 1.1.3 was considered the Hopfield net

$$x(k+1) = -\frac{1}{2}x(k) + \frac{1}{2}W^T\sigma(x) + \frac{1}{2}u$$

with $x \in \Re^2$ and symmetric sigmoid activations having decay constants $g_1 = g_2 = 100$. Suppose the prescribed exemplar patterns are

$$X_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad X_2 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Then according to the training equation (1.42), one has the weight matrix

$$W = W^T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Using these weights, state trajectory phase-plane plots for various initial condition vectors $x(0)$ were shown in Figure 1.16. Indeed, in all cases, the state trajectories converged either to the point $(-1, -1)$ or to $(1, 1)$.

1.3.2 TRAINING THE ONE-LAYER NN — GRADIENT DESCENT

In this section will be considered the problem of tuning the weights in the one-layer NN shown Figure 1.4 and described by the recall equation

$$y_l = \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right); \quad l = 1, 2, \dots, L \quad (1.43)$$

or in the matrix form

$$y = \sigma(V^T x), \quad (1.44)$$

with $x = [x_1 \ x_2 \ \dots \ x_n]^T \in \Re^{n+1}$, $y \in \Re^L$, and V the matrix of weights and thresholds. A tuning algorithm for this single-layer perceptron was first derived by Rosenblatt in 1959; he used the symmetric hard limiter activation function. Widrow and Hoff studied the case of linear $\sigma(\cdot)$ in 1960 (Haykin 1994).

There are many types of training algorithms currently in use for NN; the basic type we shall discuss is error correction training. We shall introduce a matrix-calculus-based approach that is very convenient for formulating NN training algorithms. Since NN training is usually performed using digital computers, convenient forms of weights are updated by discrete iteration steps. Such digital update algorithms are extremely convenient for computer implementation and are considered in this subsection.

1.3.2.1 Gradient Descent Tuning

In this discussion the iteration index is denoted as k . One should not think of k as a time index as the iteration index is not necessarily the same as the time index.

Let $v_{lj}(k)$ be the NN weights at the iteration k so that

$$y_l(k) = \sigma \left(\sum_{j=1}^n v_{lj}(k) X_j + v_{l0}(k) \right); \quad l = 1, 2, \dots, L \quad (1.45)$$

In this equation, X_j are the components of a prescribed constant input vector X that stays the same during training of the NN. A general class of weight-update algorithms is given by the recursive update equation

$$v_{lj}(k+1) = v_{lj}(k) - \eta \frac{\partial E(k)}{\partial v_{lj}(k)}, \quad (1.46)$$

where $E(k)$ is a cost function that is selected depending on the application. In this algorithm, the weights v_{lj} are updated at each iteration number k in such a manner that the prescribed cost function decreases. This is accomplished by going downhill against the gradient $\partial E(k)/\partial v_{lj}(k)$. The positive step size parameter η is taken as less than 1 and is called the learning rate or adaptation gain.

To see that the gradient descent algorithm decreases the cost function, note that $\Delta v_{lj}(k) \equiv v_{lj}(k+1) - v_{lj}(k)$ and, to first order

$$\begin{aligned} \Delta E(k) &= E(k+1) - E(k) \\ &\cong \sum_{lj} \frac{\partial E(k)}{\partial v_{lj}(k)} \Delta v_{lj}(k) \\ &= -\eta \sum_{lj} \left(\frac{\partial E(k)}{\partial v_{lj}(k)} \right)^2 \end{aligned} \quad (1.47)$$

Techniques such as conjugate gradient take into account second-order and higher terms in this Taylor series expansion.

Taking the cost function as the least-squares NN output error a specific gradient descent algorithm is derived. Thus, let a prescribed pattern vector X be input to the NN and the desired target output associated with X be Y (cf. Example 1.2.1). Then, at iteration number k the l th component of the output error is

$$e_l(k) = Y_l - y_l(k), \quad (1.48)$$

where Y_l is the desired output and $y_l(k)$ is the actual output with input X . Define the least-squares output error cost as

$$E(k) - \frac{1}{2} \sum_{l=1}^L e_l^2(k) = \frac{1}{2} \sum_{l=1}^L (Y_l(k) - y_l(k))^2 \quad (1.49)$$

Note that the components X_l of the input X and the desired NN output components Y_l are not functions of the iteration number k (see the subsequent discussion on series vs. batch updating).

To derive the gradient descent algorithm with least-squares output error cost, the gradients with respect to the weights and thresholds are computed using the product and the chain rule as

$$\frac{\partial E(k)}{\partial v_{lj}(k)} = -e_l(k) \sigma' \left(\sum_{j=1}^n v_{lj}(k) X_j + v_{l0}(k) \right) X_j \quad (1.50)$$

$$\frac{\partial E(k)}{\partial v_{l0}(k)} = -e_l(k) \sigma' \left(\sum_{j=1}^n v_{lj}(k) X_j + v_{l0}(k) \right), \quad (1.51)$$

where Equation 1.45 and Equation 1.49 were used. The notation $\sigma'(\cdot)$ denotes the derivative of the activation function evaluated at the argument. Therefore the gradient descent algorithm for the least-squares output-error case yields the weight updates

$$v_{lj}(k+1) = v_{lj}(k) + \eta e_l(k) \sigma' \left(\sum_{j=1}^n v_{lj}(k) X_j + v_{l0}(k) \right) X_j \quad (1.52)$$

and the threshold updates

$$v_{l0}(k+1) = v_{l0}(k) + \eta e_l(k) \sigma' \left(\sum_{j=1}^N v_{lj}(k) X_j + v_{l0}(k) \right). \quad (1.53)$$

Historically, the derivative of the activation functions was not used to update the weights prior to the 1970s (see Section 1.3.3). Widrow and Hoff took linear activation functions so that the tuning algorithm becomes the least mean-square (LMS) algorithm

$$v_{lj}(k+1) = v_{lj}(k) + \eta e_l(k) X_j \quad (1.54)$$

$$v_{l0}(k+1) = v_{l0}(k) + \eta e_l(k) \quad (1.55)$$

The LMS algorithm is popularly used for training the one-layer perceptron even if nonlinear activation functions are used. It is called the perceptron training algorithm or the delta rule. Rosenblatt showed that, using the symmetric hard limit activation functions, if the classes of the input vectors are separable using linear decision boundaries, then this algorithm converges to the correct weights (Haykin 1994).

Matrix formulation: A matrix calculus approach can be used to derive the delta rule by a streamlined method that is well suited for simplifying notation. Thus, given the input–output pair (X, Y) that the NN should associate, define the NN output error vector as

$$e(k) = Y - y(k) = Y - \sigma(V^T(k)X) \in \Re^L \quad (1.56)$$

and the least-squares output-error cost as

$$E(k) = \frac{1}{2} e^T(k) e(k) - \frac{1}{2} \text{tr}\{e(k) e^T(k)\} \quad (1.57)$$

The trace of a square matrix $\text{tr}\{\cdot\}$ is defined as the sum of the diagonal elements. One uses the expression involving the trace $\text{tr}\{ee^T\}$ due to the fact that derivatives of the trace with respect to matrices are very convenient to evaluate. On the other hand, evaluating gradients of $e^T e$ with respect to weight matrices involves the use of third-order tensors, which must be managed using the Kronecker product (Lewis et al. 1993) or other machinations. A few matrix calculus identities are very useful; they are given in Table 1.1 (Lewis et al. 1999).

In terms of matrices the gradient descent algorithm is

$$V(k+1) = V(k) - \eta \frac{\partial E(k)}{\partial V(k)} \quad (1.58)$$

Write

$$E(k) = \frac{1}{2} \text{tr}\{(Y - \sigma(V^T(k)X))(Y - \sigma(V^T(k)X))^T\} \quad (1.59)$$

where $e(k)$ is the NN output error associated with input vector X using the weights $V(k)$ determined at iteration k . Assuming linear activation functions $\sigma(\cdot)$ one has

$$E(k) = \frac{1}{2} \text{tr}\{(Y - V^T(k)X)(Y - V^T(k)X)^T\} \quad (1.60)$$

Now, using the identities in Table 1.1 (especially [1.65]) one can easily determine (see problems section) that

$$\frac{\partial E(k)}{\partial V(k)} = -Xe^T(k) \quad (1.61)$$

so that the gradient descent tuning algorithm is written as

$$V(k+1) = V(k) + \eta Xe^T(k) \quad (1.62)$$

which updates both the weights and the thresholds. Recall that the first column of V^T consists of the thresholds and the first entry of X is 1. Therefore, the threshold vector b_v in (1.7) is updated according to

$$b_v(k+1) = b_v(k) + \eta e(k) \quad (1.63)$$

It is interesting to note that the weights are updated according to the outer product of the prescribed pattern vector X and the NN output error of e .

1.3.2.2 Epoch vs. Batch Updating

We have just discussed NN weight training when one input-vector/desired output-vector pair (X , Y) is given for an NN. In practical situations, there might be multiple input vectors prescribed by the user each with an associated output vector. Thus, suppose there are P desired input/output pairs (X^1, Y^1) , $(X^2, Y^2), \dots, (X^P, Y^P)$ for the NN.

TABLE 1.1
Basic Matrix Calculus and Trace Identities

Let r, s be scalars; A, B, C be matrices; and x, y, z be vectors, all dimensioned so that the following formulae are compatible. Then:

$$\text{tr}\{AB\} = \text{tr}\{BA\} \quad (1.64)$$

when the matrices have compatible dimensions

$$\frac{\partial \text{tr}\{BAC\}}{\partial A} = B^T C^T \quad (1.65)$$

$$\frac{\partial \text{tr}\{ABA^T\}}{\partial A} = 2AB \quad (1.66)$$

$$\frac{\partial s}{\partial A^T} = \left[\frac{\partial s}{\partial A} \right]^T \quad (1.67)$$

$$\frac{\partial AB}{\partial s} = \frac{\partial A}{\partial s} B + A \frac{\partial B}{\partial s} \quad \text{product rule} \quad (1.68)$$

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x} \quad \text{chain rule} \quad (1.69)$$

$$\frac{\partial s}{\partial t} = \text{tr} \left\{ \frac{\partial s}{\partial A} \cdot \frac{\partial A^T}{\partial t} \right\} \quad \text{chain rule} \quad (1.70)$$

In such situations the NN must be trained to associate each input vector with its prescribed output vector. There are many strategies for training the net in this scenario; at the two extremes are epoch updating and batch updating, also called parallel or block updating. For this discussion we shall use matrix updates, defining for $p = 1, 2, \dots, P$ the quantities

$$y^p(k) = \sigma(V^T(k)X^p) \quad (1.71)$$

$$e^p(k) = Y^p - y^p(k) = Y^p - \sigma(V^T(k)X^p) \quad (1.72)$$

$$E^p(k) = \frac{1}{2}(e^p(k))^T e^p(k) = \frac{1}{2} \text{tr}\{e^p(k)(e^p(k))^T\} \quad (1.73)$$

In epoch updating, the vectors (X^p, Y^p) are sequentially presented to the NN. At each presentation, one step of the training algorithm is performed so that

$$V(k+1) = V(k) + \eta X^p (e^p(k))^T \quad p = 1, 2, \dots, P \quad (1.74)$$

which updates both the weights and thresholds (see [1.10]). An epoch is defined as one complete run through all the P associated pairs of input. When one epoch has been completed, the pair (X^1, Y^1) is presented again and another run through all the P pairs is performed. It is expected that after a sufficient number of epochs, the output error will become small enough.

In batch updating, all P pairs are presented to the NN (one at a time) and a cumulative error is computed after all have been presented. At the end of this procedure, the NN weights are updated once. The result is

$$V(k+1) = V(k) + \eta \sum_{p=1}^P X^p (e^p(k))^T \quad (1.75)$$

In batch updating, the iteration index k corresponds to the number of times the set of patterns P is presented and the cumulative error computed. That is, k corresponds to the epoch number.

There is a very convenient way to perform batch NN weight updating using matrix manipulations. Thus, define the matrices

$$X \equiv [X^1 \quad X^2 \quad \dots \quad X^P] \quad Y \equiv [Y^1 \quad Y^2 \quad \dots \quad Y^P] \quad (1.76)$$

which contain all P prescribed input/output vectors, and the batch error matrix

$$e(k) = [e^1(k) \quad e^2(k) \quad \dots \quad e^P(k)]. \quad (1.77)$$

It is now easy to see that the NN recall can be computed using the equation

$$y(k) = \sigma(V^T(k)X), \quad (1.78)$$

where the batch output matrix is $y(k) = [y^1(k) \quad y^2(k) \quad \dots \quad y^P(k)]$. Therefore, the batch weight update can be written as

$$V(k+1) = V(k) + \eta X e^T(k) \quad (1.79)$$

This method involves the concept of presenting all P of the prescribed inputs X_p to the NN simultaneously.

It has been mentioned that the update iteration index k is not necessarily the same as the time index. In fact, one now realizes that the relation between k and the time is dependant on how one chooses to process multiple prescribed input-output pairs.

Example 1.3.2 (NN Training — a Simple Classification Example): It is desired to design a one-layer NN with two inputs and two outputs (Lewis et al. 1999) that classifies the following ten points in \mathbb{R}^2 into the four groups shown:

Group 1:

$$(0.1, 1.2), (0.7, 1.8), (0.8, 1.6)$$

Group 2:

$$(0.8, 0.6), (1.0, 0.8)$$

Group 3:

$$(0.3, 0.5), (0.0, 0.2), (-0.3, 0.8)$$

Group 4:

$$(-0.5, -1.5), (-1.5, -1.3)$$

These points are shown in Figure 1.26, where the groups are denoted respectively by +, o, \times , *. The hard limit activation function will be used as it is suitable for classification problems.

To cast this in terms tractable for NN design, encode the four groups, respectively, by 10, 00, 11, 01. Then define the input pattern matrix as

$$\begin{aligned} p &= [X^1 \quad X^2 \quad \dots \quad X^{10}] \\ &= \begin{bmatrix} 0.1 & 0.7 & 0.8 & 0.8 & 1.0 & 0.3 & 0.0 & -0.3 & -0.5 & -1.5 \\ 1.2 & 1.8 & 1.6 & 0.6 & 0.8 & 0.5 & 0.2 & 0.8 & -1.5 & -1.3 \end{bmatrix} \end{aligned}$$

and the target vector as

$$\begin{aligned} t &= [Y^1 \quad Y^2 \quad \dots \quad Y^{10}] \\ &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Then, the three points associated with the target vector $[1 \quad 0]^T$ will be assigned to the same group, and so on. The design will be carried out using the MATLAB NN Toolbox. The one-layer NN with two neurons is set up using the function

NEWP(). Weights v and biases b are initialized randomly from the interval between -1 and 1 .

```
net = newp(minmax(p), 2);
net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);
v = net.iw{1,1};
b = net.b{1};
```

The result is

$$v = \begin{bmatrix} -0.5621 & 0.3577 \\ 0.0059 & 0.3586 \end{bmatrix} \quad b = \begin{bmatrix} 0.8654 \\ -0.2330 \end{bmatrix}$$

Each output y_l of the NN yields one decision line in the \Re^2 plane as shown in Example 1.1.1. The two lines given by the random initial weights are drawn using the commands.

```
plotpv(p, t) % draws the points corresponding to the
              10 input vectors
plotpc(v, b) % superimposes the decision lines
              corresponding to weight v and bias b
```

The initial decision lines are shown in Figure 1.26.

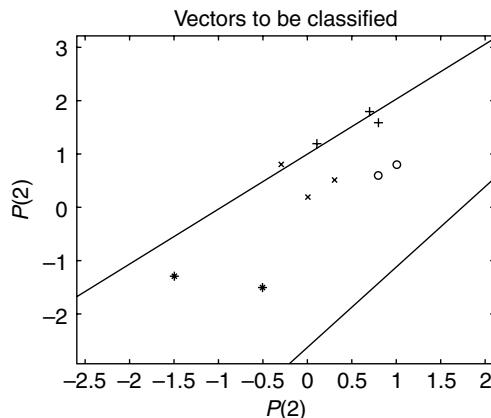


FIGURE 1.26 Pattern vectors to be classified into four groups: $+$, o , x , $*$. Also shown are the initial decision boundaries.

The NN was trained using the batch updating algorithm (1.78). The MATLAB commands are

```
net.trainParam.epochs = 3;
net.trainParam.goal = 1e-10;
net = train(net,p,t);
y = sim(net,p);
```

where `net.trainParam.epochs` specifies the number of epochs for which training should continue and `net.trainParam.goal` specifies the error goal. Recall that an epoch is one complete presentation of all ten patterns to the NN (in this case all ten are presented simultaneously using the batch update techniques discussed in connection with [1.79]).

After three epochs, the weights and biases are

$$v = \begin{bmatrix} -0.5621 & 6.4577 \\ -1.2039 & -1.6414 \end{bmatrix} \quad b = \begin{bmatrix} 0.8694 \\ 1.7670 \end{bmatrix}$$

The corresponding decision lines are shown in Figure 1.27a.

Now the numbers of epochs were increased to 20 and the NN training was continued. After 3 further epochs (e.g., 6 epochs in all) the error was small enough and training was ceased. The final weights and biases are

$$v = \begin{bmatrix} 3.8621 & 4.5577 \\ -1.2039 & -1.6414 \end{bmatrix} \quad b = \begin{bmatrix} -0.1306 \\ 1.7670 \end{bmatrix}$$

and the final decision boundaries are shown in Figure 1.27b. The plot of least-squares output error (1.73) vs. epoch is shown in Figure 1.28.

1.3.3 TRAINING THE MULTILAYER NN — BACKPROPAGATION TUNING

A one-layer NN can neither approximate general functions nor perform the X-OR operation which is basic to digital logic implementations. When it was demonstrated that the two-layer NN has both these capabilities and that a three-layer NN is sufficient for most general pattern classification applications, there was a sudden interest in multilayer NN. Unfortunately, for years it was not understood how to train a multilayer network.

The problem involved the assignment of part of the credit to each weight for the NN output errors in order to determine how to tune that

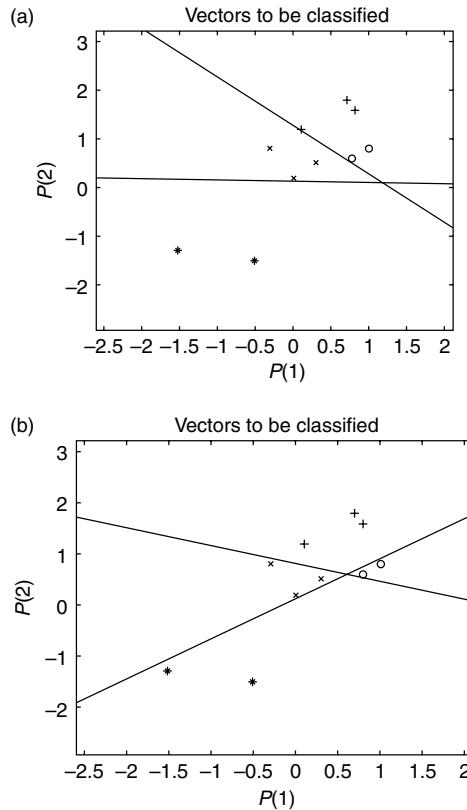


FIGURE 1.27 NN decision boundaries. (a) After three epochs of training. (b) After seven epochs of training.

weight. This so-called credit assignment problem was finally solved by several researchers (Werbos 1974, 1989, Rumelhart et al. 1986), who derived the Backpropagation Training Algorithm. The solution is surprisingly straightforward in retrospect, hinging on a simple application of calculus using the chain rule.

In Section 1.3.2 it was shown how to train a one-layer NN. There, the delta rule was derived ignoring the nonlinearity of the activation function. In this section we show how to derive the full NN weight-update rule for a multilayer NN including all activation function nonlinearities. For this application, the actuation functions selected must be differentiable. Though backpropagation enjoys great success, one must remember that it is still a gradient-based technique so that the usual caveats associated with step sizes, local minima, and so on must be kept in mind when using it (see Section 1.3.4).

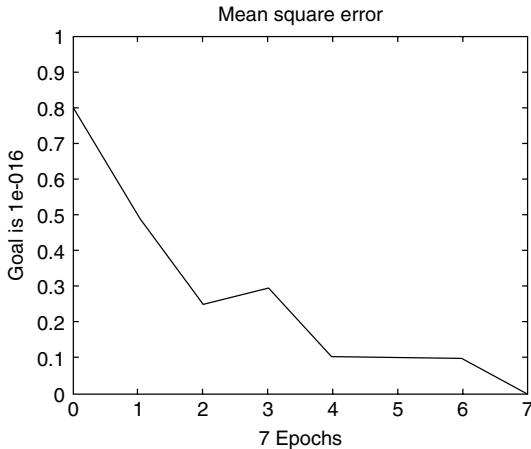


FIGURE 1.28 Least-squares NN output error vs. epoch.

1.3.3.1 Background

We shall derive the backpropagation algorithm for the two-layer NN in Figure 1.6 described by

$$y_i = \sigma \left(\sum_{l=1}^L w_{il} \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right) + w_{i0} \right) \quad i = 1, 2, \dots, m \quad (1.80)$$

The derivation is greatly simplified by defining some intermediate quantities.

In Figure 1.6 we call the layer of weights v_{lj} the first layer and the layer of weights w_{il} the second layer. The input to layer one is x_j . Define the input to layer two as

$$z_l = \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right) \quad l = 1, 2, \dots, L \quad (1.81)$$

The thresholds can more easily be dealt with by defining $x_0 \equiv 1, z_0 \equiv 1$. Then one can say

$$y_i = -\sigma \left(\sum_{l=0}^L w_{il} z_l \right) \quad (1.82)$$

$$z_l = \sigma \left(\sum_{l=0}^L w_{lj} x_j \right) \quad (1.83)$$

It is convenient at this point to begin thinking in terms of moving backward through the NN, hence the ordering of this and subsequent lists of equations. Define the outputs of layers two and one, respectively, as

$$u_i^2 = \sum_{l=0}^L w_{il} z_l \quad (1.84)$$

$$u_l^1 = \sum_{j=0}^n v_{lj} x_j \quad (1.85)$$

Then we can write

$$y_i = \sigma(u_i^2) \quad (1.86)$$

$$z_l = \sigma(u_l^1) \quad (1.87)$$

In deriving the backpropagation algorithm we shall have the occasion to differentiate the activation functions. Note therefore that

$$\frac{\partial y_i}{\partial w_{il}} = \sigma'(u_i^2) z_l \quad (1.88)$$

$$\frac{\partial y_i}{\partial z_l} = \sigma'(u_i^2) w_{il} \quad (1.89)$$

$$\frac{\partial z_l}{\partial v_{lj}} = \sigma'(u_l^1) x_j \quad (1.90)$$

$$\frac{\partial z_l}{\partial x_j} = \sigma'(u_l^1) v_{lj}, \quad (1.91)$$

where $\sigma'(\cdot)$ is the derivative of the activation function.

Part of the power of the soon-to-be-derived backpropagation algorithm is the fact that the evaluation of the activation function derivative is very easy for common $\sigma(\cdot)$. Specifically, selecting the sigmoid activation function

$$\sigma(s) = \frac{1}{1 + e^{-s}} \quad (1.92)$$

one obtains

$$\sigma'(s) = \sigma(s)(1 - \sigma(s)) \quad (1.93)$$

which is very easy to compute using simple multipliers.

1.3.3.2 Derivation of the Backpropagation Algorithm

Backpropagation weight tuning is a gradient descent algorithm, so the weights in layers two and one, respectively, are updated according to

$$w_{il}(k+1) = w_{il}(k) - \eta \frac{\partial E(k)}{\partial w_{il}(k)} \quad (1.94)$$

$$v_{lj}(k+1) = v_{lj}(k) - \eta \frac{\partial E(k)}{\partial v_{lj}(k)}, \quad (1.95)$$

with E as a prescribed cost function. In this discussion we shall conserve simplicity of notation by dispensing with the iteration index k (cf. Section 1.3.2), interpreting these equalities as replacements. The learning rates η in the two layers can of course be selected as different.

Let there be prescribed an input vector X and an associated desired output vector Y for the network. Define the least squares NN output error as

$$E(k) = \frac{1}{2} e^T(k) e(k) = \frac{1}{2} \sum_{i=1}^m e_i^2(k) \quad (1.96)$$

$$e_i(k) = Y_i(k) - y_i(k), \quad (1.97)$$

where $y_i(k)$ is evaluated using (1.80) with the components of the input pattern X_j as the NN inputs $x_i(k)$.

The required gradients of the cost E with respect to the weights are now very easily determined using the chain rule. Specifically, for the second-layer weights

$$\frac{\partial E}{\partial w_{il}} = \frac{\partial E}{\partial u_i^2} \frac{\partial u_i^2}{\partial w_{il}} = \left[\frac{\partial E}{\partial e_i} \frac{\partial e_i}{\partial y_i} \frac{\partial y_i}{\partial u_i^2} \right] \frac{\partial u_i^2}{\partial w_{il}} \quad (1.98)$$

and using the above equalities one obtains

$$\frac{\partial E}{\partial u_i^2} = -\sigma'(u_i^2)e_i \quad (1.99)$$

$$\frac{\partial E}{\partial w_{il}} = -z_l[\sigma'(u_i^2)e_i]. \quad (1.100)$$

Similarly, for the first-layer weights

$$\frac{\partial E}{\partial v_{lj}} = \frac{\partial E}{\partial u_l^1} \frac{\partial u_l^1}{\partial v_{lj}} = \left[\sum_{i=1}^m \frac{\partial E}{\partial u_i^2} \frac{\partial u_i^2}{\partial z_l} \frac{\partial z_l}{\partial u_l^1} \right] \frac{\partial u_l^1}{\partial v_{lj}} \quad (1.101)$$

$$\frac{\partial E}{\partial v_{lj}} = -\sigma'(u_l^1) \sum_{i=1}^m w_{il} [\sigma'(u_i^2) e_i] \quad (1.102)$$

$$\frac{\partial E}{\partial v_{lj}} = -X_j \left[\sigma'(u_l^1) \sum_{i=1}^m w_{il} [\sigma'(u_i^2) e_i] \right]. \quad (1.103)$$

These equations can be considerably simplified by introducing the notion of a backward recursion through the network. Thus, define the backpropagated error for layers two and one, respectively, as

$$\delta_i^2 \equiv -\frac{\partial E}{\partial u_i^2} = \sigma'(u_i^2) e_i \quad (1.104)$$

$$\delta_i^1 \equiv -\frac{\partial E}{\partial u_i^1} = \sigma'(u_i^1) \sum_{l=1}^m w_{il} \delta_l^2 \quad (1.105)$$

Assuming the sigmoid activation functions are used, the backpropagated errors can be computed as

$$\delta_i^2 = y_i(1 - y_i)e_i \quad (1.106)$$

$$\delta_i^1 = z_l(1 - z_l) \sum_{l=1}^m w_{il} \delta_l^2. \quad (1.107)$$

Combining these equations one obtains the backpropagation algorithm given in Table 1.2. There, the algorithm is given in terms of a forward recursion through the NN to compute the output, then a backward recursion to determine the backpropagated errors, and finally a step to determine the weight updates. Such two-pass algorithms are standard in DSP and optimal estimation theory. In fact, one should particularly examine optimal smoothing algorithms contained, for instance, in Lewis (1986). The backpropagation algorithm may be employed using series or batch processing of multiple input/output patterns (see Section 1.3.3.1), and may be modified to use adaptive step size η or momentum training (see Section 1.3.3.3).

Note that the threshold updates are given by

$$w_{i0} = w_{i0} + \eta \delta_i^2 \quad (1.108)$$

$$w_{l0} = w_{l0} + \eta \delta_l^1 \quad (1.109)$$

In many applications the NN has no activation functions in the output layer (e.g., the activation function is linear in [1.111]). Then one must use simply $\delta_i^2 = e_i$ in the equations for backpropagation.

TABLE 1.2
Backpropagation Using Sigmoid Activation Functions:
Two-Layer Network

The following iterative procedure should be repeated until the NN output error has become sufficiently small. Series of batch processing of multiple input/output patterns (X, Y) may be used. Adaptive learning rate η and momentum terms may be added.

Forward Recursion to Compute NN Output

Present input pattern X to the NN and compute the NN output using

$$z_l = \sigma \left(\sum_{j=0}^n v_{lj} X_j \right); \quad l = 1, 2, \dots, L \quad (1.110)$$

$$y_i = \sigma \left(\sum_{l=0}^L w_{il} z_l \right); \quad i = 1, 2, \dots, m \quad (1.111)$$

with X_0 and z_0 , where Y is the desired output pattern.

Backward Recursion for Backpropagated Errors

$$e_i = Y_i - y_i; \quad i = 1, 2, \dots, m \quad (1.112)$$

$$\delta_i^2 = y_i(1 - y_i)e_i; \quad i = 1, 2, \dots, m \quad (1.113)$$

$$\delta_l^1 = z_l(1 - z_l) \sum_{i=1}^m w_{il} \delta_i^2; \quad l = 1, 2, \dots, L \quad (1.114)$$

Computation of the NN Weight and Threshold Updates

$$w_{il} = w_{il} + \eta z_l \delta_i^2; \quad i = 1, 2, \dots, m; \quad l = 0, 1, \dots, L \quad (1.115)$$

$$v_{lj} = v_{lj} + \eta X_j \delta_l^1; \quad l = 1, 2, \dots, L; \quad j = 0, 1, \dots, n \quad (1.116)$$

In terms of signal vectors and weight matrices one may write the backpropagation algorithm as follows:

The forward recursion becomes

$$z = \sigma(V^T X) \quad (1.117)$$

$$y = \sigma(W^T z) \quad (1.118)$$

and the backward recursion is

$$e = Y - y \quad (1.119)$$

$$\delta^2 = \text{diag}\{y\}(I - \text{diag}\{y\})e \quad (1.120)$$

$$\delta^1 = \text{diag}\{z\}(I - \text{diag}\{z\})W\delta^2 \quad (1.121a)$$

where y is an m -vector, $\text{diag}\{y\}$ is an $m \times m$ diagonal matrix having the entries y_1, y_2, \dots, y_m on the diagonal. The weight and threshold updates are

$$w = w + yz(\delta^2)^T \quad (1.121b)$$

$$v = v + yx(\delta^1)^T \quad (1.121c)$$

At this point one notices quite an interesting occurrence. The forward recursion of the backpropagation algorithm is based, of course, on the NN weight matrices; however, the backward recursion is based on the transposes of the weight matrices. Moreover, it is accomplished by working backward through the transposed NN. In systems theory the dual, backward system is known as the adjoint system. This system enjoys some very special properties in relation to the original system, many associated with determining solutions to optimality and control problems (Lewis and Syrmos 1995). Such notions have not yet been fully explored in the context of the NN.

An intriguing concept is that of the adjoint NN for training. This backpropagation network was discussed by Narendra and Parthasarathy (1990) and is depicted in Figure 1.29. The adjoint training net is based on the transposes of the NN weight matrices and contains multipliers. In this respect, it is very similar to various optimal control and adaptive filtering control schemes wherein the computation and tuning of the feedback control gains is carried out in outer loops containing multipliers. The multiplier is fundamental to higher-level and intelligent control. In the 1940s, Norbert Wiener introduced his new field of Cybernetics. It was he who said that developments on two fronts were required

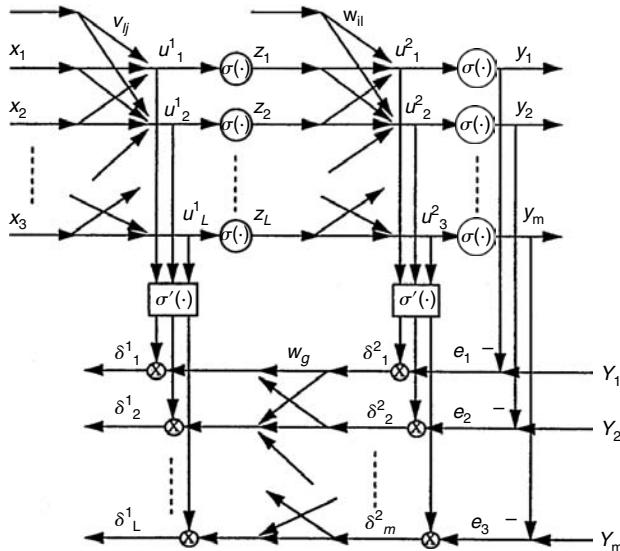


FIGURE 1.29 The adjoint (backpropagation) neural network.

prior to further advances in system theory, increasing computing power and the theory of the multiplier (Wiener 1948).

By now several improvements have been made on the backpropagation algorithm given here. A major increase in speed is offered by the Levenberg–Marquardt algorithm, which combines gradient descent and the Gauss–Newton algorithm. The next section discusses some other techniques for improving backpropagation.

Example 1.3.3 (NN Function Approximation): It is known that a two-layer NN (Lewis et al. 1999) with sigmoid activation functions can approximate arbitrarily accurately any smooth functions (see Section 1.2.2). In this example, it is desired to design a two-layer NN to approximate the function shown in Figure 1.30, so that the NN has one input x and one output y . The hidden-layer activation functions will be hyperbolic tangent and the output-layer activation function will be linear.

The NN weights will be determined using backpropagation training with batch updates. First, exemplar input pattern and target output vectors must be selected. Select therefore the input vectors X to correspond to the abscissa x of the function graph and the target outputs corresponding to the ordinate or function values $y = f(x)$. A sampling interval of 0.1 is selected, so that $X = p$ is a row vector of 21 values $Y = t$ are determined, shown by \circ on the graph

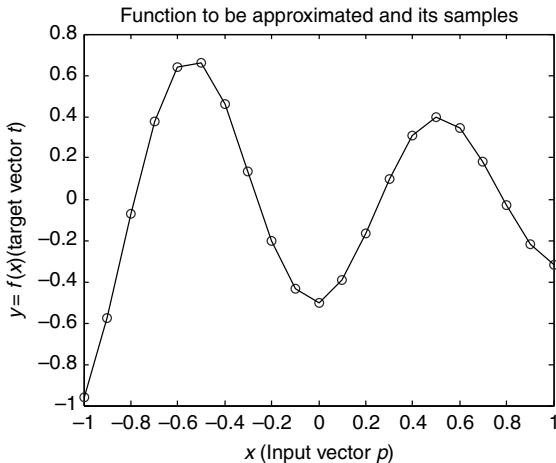


FIGURE 1.30 Function $y = f(x)$ to be approximated by two-layer NN and its samples for training.

in Figure 1.30. The MATLAB commands to set up the input and target output vectors are

```
p = -1:0.1:1; t
= [-0.960 -0.577 -0.073 0.377 0.641 0.660 0.461 0.134
-0.201 -0.434 -0.500...
-0.393 -0.165 0.099 0.307 0.396 0.345 0.182 -0.031
-0.219 -0.320];
```

Five hidden-layer neurons were selected (see comments at the end of this example).

The NN weights were initialized using

```
[v,bv,w,bw] =
initff(p,5,'tansig',1,'purelin') ;
with v, bv the first-layer weight matrix and bias
vector, and w, bw the second-layer weight matrix and
bias vector. Now, the output of the NN using these
random weights was determined and plotted using
y0 = simuff(p,v,bv,'tansig',w,bw,'purelin');
plot(p,y0,'-',p,t,'o')
set(gca,' xlabel',text(0,0,'x (input vector p)'))
set(gca,' ylabel',text(0,0,'Samples of f(x) and actual
NN output'))
```

The result is shown in Figure 1.31a.

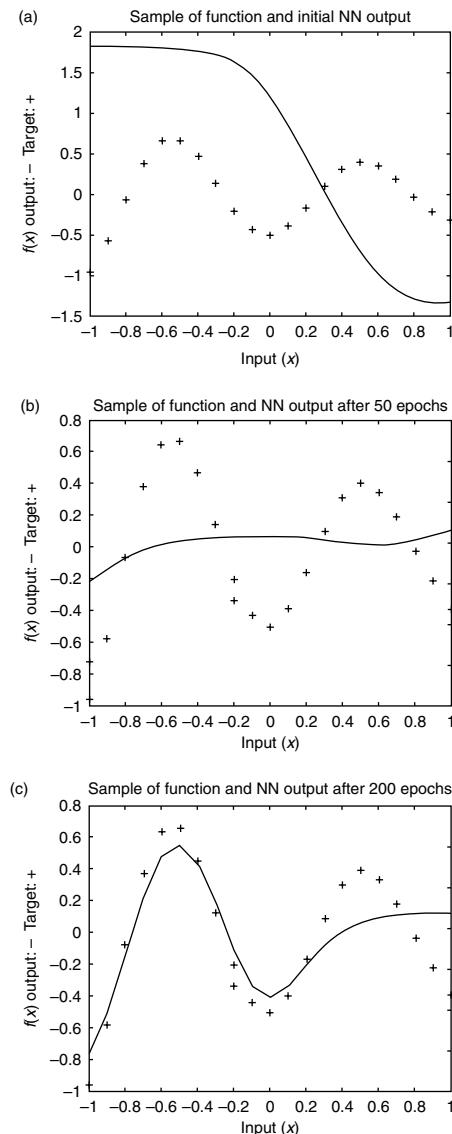


FIGURE 1.31 Samples of $f(x)$ and actual NN output. (a) Using initial random weights. (b) After training for 50 epochs. (c) After training for 200 epochs. (d) After training for 873 epochs. (e) After training for 24 epochs using Levenberg–Marquardt backpropagation.

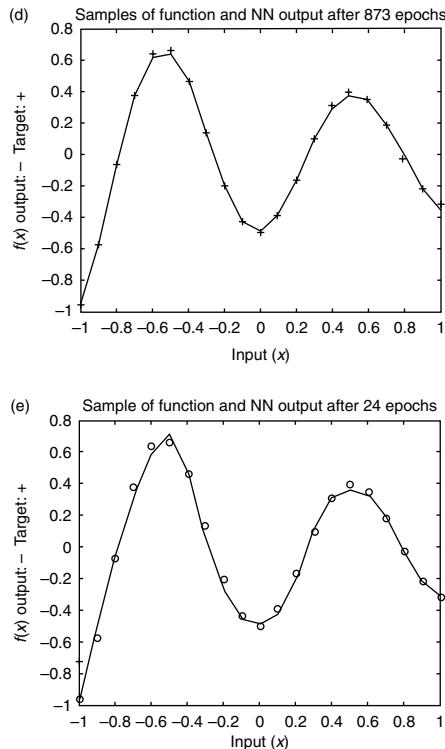


FIGURE 1.31 Continued.

The NN was now trained using the backpropagation algorithm (1.119)–(1.121c) with batch updating (see [1.79]). The MATLAB command is

```
tp = [10 50 0.005 0.01];
[v,bv,w,bw]=trainbp(v,bv,'tansig',w,bw,'purelin',p,tp);
```

The least-squares output error is computed every 10 epochs and the training is to be carried out for a maximum of 50 epochs. The training is stopped when the least-squares output error goes below 0.005, and that the learning rate η is 0.01. After training, the NN output was plotted and is displayed in Figure 1.31b.

This procedure was repeated, plotting the NN output after 200 epochs and after 873 epochs, when the least-squares output error fell below 0.005. The results are shown in Figure 1.31c. The final weights after the training was

complete were

$$v = \begin{bmatrix} 3.6204 \\ 3.8180 \\ 3.5548 \\ 3.0169 \\ 3.6398 \end{bmatrix} \quad bv = \begin{bmatrix} -2.7110 \\ 1.2214 \\ -0.7778 \\ 2.1751 \\ 2.9979 \end{bmatrix}$$

$$w = [-0.6334 \quad -1.2985 \quad 0.8719 \quad 0.5937 \quad 0.9906] \quad bw = [-1.0295]$$

To obtain the plots shown in Figure 1.31, including the final plot shown in Figure 1.31d, a refined input vector p was used corresponding to samples at a uniform spacing of 0.01 on the interval $[-1, 1]$.

Alternately, the desired function can also be approximated using

```
net = newff(minmax(p), [5,1], {'tansig', 'purelin'},
    'trainlm');
net.trainParam.show = 10;
net.trainParam.lr = 0.01;
net.trainParam.epochs = 50;
net.trainParam.goal = 0.005;
net = train(net,p,t);
ylabel('Sum-Squared Error, Goal: 0.005');
title('Sum Squared Network Error for 50 epochs');
y0 = sim(net,p);
figure;
plot(p,t,'o',p,y0,'-')
title('Samples of function and NN output after
50 epochs');
xlabel('Input (x)');
ylabel('f(x) Output: - Target: +');
```

The TRAIN() function uses the Levenberg–Marquardt backpropagation algorithm. NN minimization problems are usually hard to solve. The Levenberg–Marquardt algorithm is used in such cases, it converges much faster than steepest descent backpropagation. With the new algorithm the desired result shown in Figure 1.31e was obtained within just 24 epochs. The NN output was simply obtained by using MATLAB function *sim()* with the new p vector. This shows clearly that, after training, the NN will interpolate between values used in the original p that was used for training, determining correct outputs for samples not in the training data. This important property is

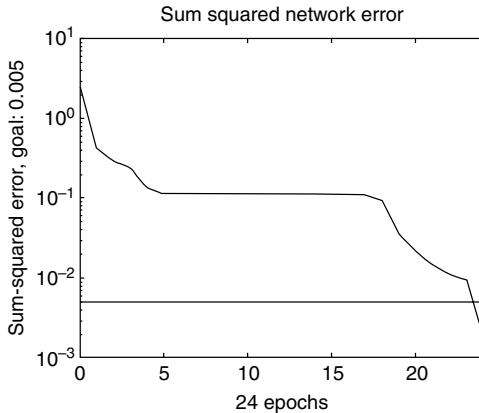


FIGURE 1.32 Least-squares NN output error as a function of training epoch.

known as the generalization property, and is closely connected to the associative memory property that close inputs should produce close NN outputs. The least-squares output error is plotted as a function of training epochs in Figure 1.32.

This example was initially performed using three hidden-layer neurons. It was found that even after several thousand epochs of training, the NN was unable to approximate the function. Therefore, the number of hidden-layer neurons was increased to five and the procedure was repeated. Using MATLAB, it took about 15 min to run this entire example and make all plots.

Example 1.3.4 (NN Approximation): Using MLP NN trained with back-propagation to map the following nonlinear function:

$$f(x, y) = \sin(\pi x) \cos(\pi y) \quad x \in (-2, 2) \text{ and } y \in (-2, 2)$$

First we can see the function is absolutely nonlinear, MATLAB as following can draw its shape:

```
figure(1);
[X, Y] = meshgrid(-2:0.1:2);
z = sin(pi*X).*cos(pi*Y);
mesh(X, Y, z);
title('Function Graphics');
```

Figure 1.33 shows the original nonlinear function.

```
% Generate Input & Target data. Totally 2000 groups.

for i=1:2000
    P(:,i) = 4*(rand(2,1)-.5);
    T(:,i)=sin(pi*P(2*i-1))*cos(pi*P(2*i));
end
```

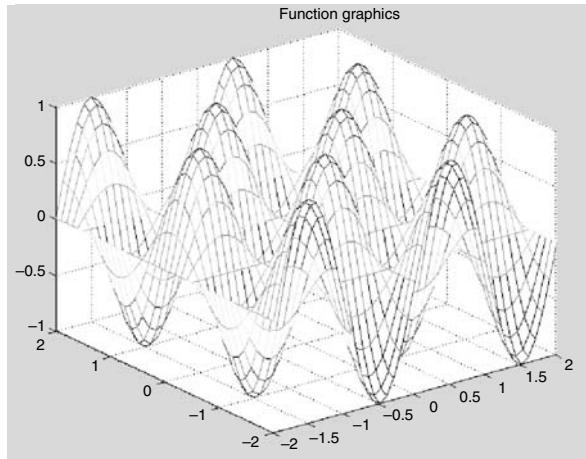


FIGURE 1.33 Nonlinear function to be approximated.

```
% BP training (1).
% Here a two-layer feed-forward network is created.
% The network's
% input ranges from [-2 to 2]. The first layer has twenty
% TANSIG
% neurons, the second layer has one PURELIN neuron.
% The TRAINGD (Basic gradient descent)
% network training function is to be used.
```

```
net1=newff(minmax(P),[20,1],{'tansig','purelin'},
            'traingd');
net1.inputWeights{:, :}.initFcn = 'rands';
net1.layerWeights{:, :}.initFcn= 'rands';
net1.trainParam.show = 50;
net1.trainParam.epochs = 1000;
net1.trainParam.goal = 1e-5;
[net1,tr]=train(net1,P,T);
```

We can compare the performance of our trained network by depicting as a graph the NN output function shape.

```
a=zeros(41,41);
[X,Y] = meshgrid(-2:0.1:2);
for i = 1 : 1681
    a(i) = sim(net1,[X(i);Y(i)]);
end
mesh(X,Y,a);
title('Net1 result');
```

Figure 1.34 illustrates the NN output after training.

```
% BP training (2).
% Now we use TRAINGDM (Gradient descent with momentum)
% training
%function.
%This time we introduce validation set.

for i=1:2000
    P(:,i) = 4*(rand(2,1)-.5);
    T(:,i)=sin(pi*P(2*i-1))*cos(pi*P(2*i));
end
for i=1:50
    P1(:,i) = 4*(rand(2,1)-.5);
    T1(:,i)=sin(pi*P1(2*i-1))*cos(pi*P1(2*i));
end
val.P=P1;
val.T=T1;
net2=newff(minmax(P),[10,1],{'tansig','purelin'},
            'traingdm');
net2.inputWeights{:, :}.initFcn = 'rands';
net2.layerWeights{:, :}.initFcn= 'rands';
net2.trainParam.show = 50;
net2.trainParam.epochs = 1000;
net2.trainParam.goal = 1e-5;
[net2,tr]=train(net2,P,T,[],[],val);

b=zeros(41,41);
[X,Y] = meshgrid(-2:0.1:2);
for i = 1 : 1681
    b(i) = sim(net2,[X(i);Y(i)]);
end
mesh(X,Y,b);
title('Net2 result');
```

Figure 1.35 depicts the NN output after 1000 epochs of training. You may want to find the detailed information in the network toolbox online manual <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet.html>. Sometimes we need to try a different scale of training set or use a different training algorithm. It is obvious the above two solution still not provided good results. You can try this example by yourself to see what combination of training algorithm, activation function, input layer's neuron number, and training set's scale can give the best result. The following in Figure 1.36

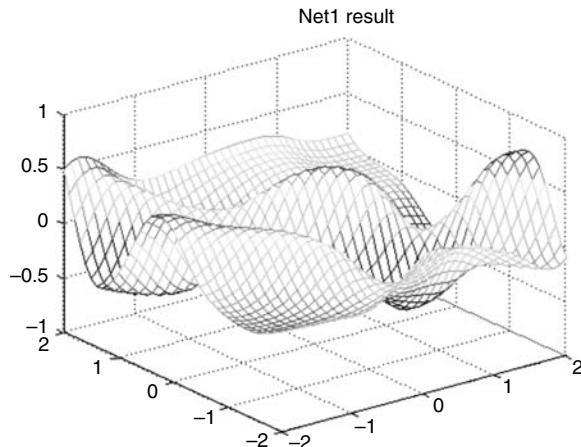


FIGURE 1.34 NN output after training.

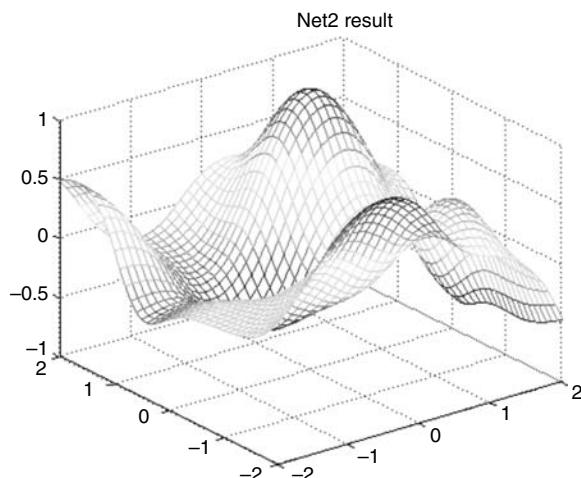


FIGURE 1.35 NN output after 1000 epochs.

was obtained using Levenberg–Marquardt backpropagation training using 40 neurons.

1.3.3.3 Improvements on Gradient Descent

Several improvements can be made to correct deficiencies in gradient descent NN training algorithms. These can be applied at each layer of

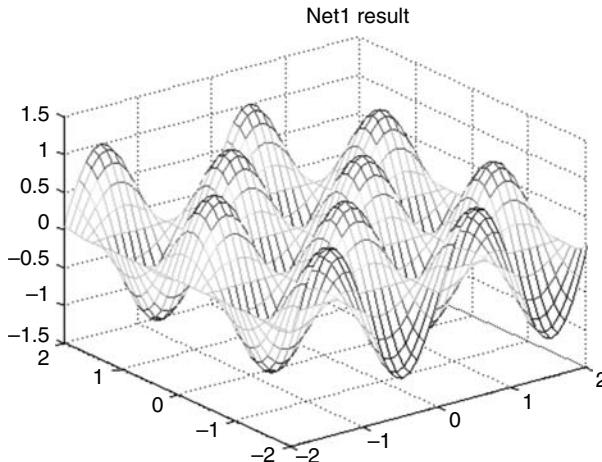


FIGURE 1.36 NN approximation using Levenberg–Marquardt backpropagation.

a multilayer NN when using backpropagation tuning. Two major issues are that gradient-based minimization algorithms provide only a local minimum, and the verification (1.46) that gradient descent decreases the cost function is based on an approximation. Improvements in performance are given by selecting better initial conditions, using learning with momentum, and using an adaptive learning rate η . References for this section include Goodwin and Sin (1984), Haykin (1994), Kung (1993), and Peretto (1992). All these refinements are available in the MATLAB NN Toolbox (1995).

Better initial conditions: The NN weights and thresholds are typically initialized to small random (positive and negative) values. A typical error surface graph in one-dimensional (1D) is given in Figure 1.37, which shows a local minimum and a global minimum. If the weight is initialized as shown in Case 1, there is a possibility that the gradient descent algorithm might find the local minimum, rolling downhill to the shallow bowl. Several authors have determined better techniques to initialize the weights than by random selection, particularly for the multilayer NN. Among these are Nguyen and Widrow, whose techniques are used, for instance, in MATLAB. Such improved initialization techniques can also significantly speedup convergence of the weights to their final values.

Learning with momentum: An improved version of gradient descent is given by the Momentum Gradient Algorithm.

$$V(k+1) = \beta V(k) + \eta(1-\beta)Xe^T(k) \quad (1.122)$$

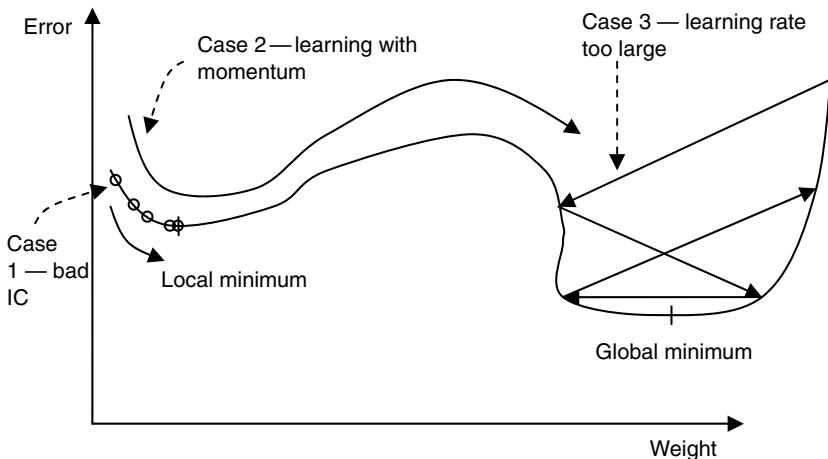


FIGURE 1.37 Typical 1D NN error surface $e = Y - \sigma(V^T X)$.

with positive momentum parameter $\beta < 1$ and positive learning rate $\eta < 1$; β is generally selected near 1 (e.g., 0.95). This corresponds in discrete-time dynamical system to moving the system pole from $z = 1$ to the interior of the unit circle, and adds stability in a manner similar to friction effects in mechanical systems.

Momentum adds a memory effect so that the NN in effect responds not only to the local gradient, but also to recent trends in the error surface. As shown by the next example, without momentum the NN can get stuck in a local minimum; adding momentum can help the NN ride through local minima. For instance, referring to Figure 1.37, using momentum as in Case 2 will cause the NN to slide through the local minimum, coming to rest at the global minimum. In the MATLAB NN Toolbox are some examples showing that learning with momentum can significantly speedup and improve the performance of backpropagation.

Adaptive learning rate: If the learning rate η is too large, the NN can overshoot the minimum cost value, jumping back and forth over the minimum and failing to converge, as shown in Figure 1.37 Case 3. Moreover, it can be shown that the learning rate in an NN layer must decrease as the number of neurons in that layer increases. Apart from correcting these problems, adapting the learning rate can significantly speedup the convergence of the weights. Such notions are standard in adaptive control theory (Goodwin and Sin 1984).

The gradient descent algorithm with adaptive learning rate is given by

$$V(k+1) = V(k) + \eta(k)x e^T(k) \quad (1.123)$$

Two techniques for selecting the adaptive learning rate $\eta(k)$ are now given.

The learning rate in any layer of weights of an NN is limited by the number of input neurons to that layer (Jagannathan and Lewis 1995). A learning rate that takes this into account is given by

$$\eta(k) = v \frac{1}{\|z\|^2}, \quad (1.124)$$

where $0 < v < 1$ and z is the input vector to the layer. As the number of input neurons to the layer increases, the norm gets larger (note that $z \in \mathbb{R}^{L+1}$, with L the number of neurons in the input to the layer). This is nothing but the standard projection method in adaptive control (Goodwin and Sin 1984).

Another technique to adapt η is given as follows. If the learning rate is too large, the NN can overshoot the minimum and never converge (see Figure 1.37 Case 3). Various standard techniques from optimization theory can be used to correct this problem; they generally rely on reducing the learning rate as a minimum is approached. The following technique increases the learning rate if the cost $E(k)$ (see [1.57]) is decreasing. If the cost increases during any iteration, however the old weights are retained and the learning step size is repeatedly reduced until the cost decreases on that iteration.

$$1 \quad V(k + 1) = V(k) + \eta(k)xe^T(k)$$

If $E(k + 1) < E(k)$; retain $V(k + 1)$ and increase learning step size

$$\eta(k + 1) = x(1 + \alpha)\eta(k)$$

Go to 2

If $E(k + 1) > E(k)$; reject $V(k + 1)$ and decrease learning step size

$$\eta(k) = (1 - \alpha)\eta(k)$$

Go to 1

$$2 \quad k = k + 1$$

Go to next iteration (1.125)

The positive parameter α is generally selected as about 0.05. Various modifications of this technique are possible.

Safe learning rate: A safe learning rate can be derived as follows. Let z be the input vector to the layer of weights being tuned, and the number of neurons in the input be L so that $z \in \mathbb{R}^{L+1}$. If the activation function is bounded by one (see Figure 1.3), then $\|z\|^2 < L + 1$, and the adaptive learning rate (1.124) is

always bounded below by

$$\eta(k) = v \frac{1}{L+1}. \quad (1.126)$$

That is, taking $v = 1$ in (1.126) provides a safe maximum allowed learning rate in an NN layer with L input neurons; a safe learning rate η for that layer is less than $1/(L + 1)$.

1.3.4 Hebbian Tuning

In the 1940s D.O. Hebb proposed a tuning algorithm based on classical conditioning experiments in psychology and by the associative memory paradigm which these observations suggest (Peretto 1992). In this subsection, we shall dispense with the iteration index k , interpreting the weight-update equations as replacements. Consider the one-layer NN in Figure 1.4 with recall equation

$$\underline{y}_l = \sigma \left(\sum_{j=1}^n v_{lj} x_j + v_{l0} \right); \quad l = 1, 2, \dots, L \quad (1.127)$$

Suppose first that the NN is to discriminate among P patterns X^1, X^2, \dots, X^P each in \Re^n and having components $X_i^p, i = 1, 2, \dots, n$. In this application, the net is square so that $L = n$. A pattern X^p is stable if its stabilization parameters are all positive.

$$\sum_{j \neq l} v_{lj} X_l^p X_j^p > 0; \quad l = 1, 2, \dots, n \quad (1.128)$$

The stabilization parameters are a measure of how well imprinted the pattern X^p is with respect to the l th neuron in a given NN. Define therefore the cost as

$$E = - \sum_{p=1}^P \sum_{j,l=1}^n v_{lj} X_l^p X_j^p \quad (1.129)$$

which, if minimized gives large stabilization parameters.

Using this cost in the gradient algorithm (1.46) yields the Hebbian tuning rule

$$v_{lj} = v_{lj} + \eta \sum_{p=1}^P X_l^p X_j^p \quad (1.130)$$

In matrix terms this may be written as

$$V = V + \eta \sum_{p=1}^P X^P (X^P)^T, \quad (1.131)$$

whence it is seen that the update for the weight matrix is given in terms of the outer product of the desired pattern vectors. This is a recursive technique in the same spirit as Hopfield's direct computation formula (1.42).

Various extensions have been made to this Hebbian or outer product training technique in the case of nonsquare NN and multilayer NN. For instance, if $L \neq n$ in a one-layer net, and the N is to associate P patterns X^p , each in \Re^n , with P target outputs Y^p , each in \Re^L , a modified Hebbian tuning rule is given by

$$V = V + \eta \sum_{p=1}^P X^p (Y^p)^T, \quad (1.132)$$

or by

$$V = V + \eta \sum_{p=1}^P X^p (e^p)^T, \quad (1.133)$$

where the output error for the pattern p is given by $e^p = Y^p - y^p$, with y^p the actual NN output given when the NN input is X^p .

The two-layer NN of Figure 1.6 has the recall equation

$$z = \sigma(V^T x) \quad (1.134)$$

$$y = \sigma(W^T z), \quad (1.135)$$

with $z \in \Re^L$ the hidden-layer output vector. Suppose the NN is to associate the input pattern X to the output vector Y . Define the output error as $e = Y - y$, with

y the output when $x = X$. Then, a tuning rule based on the Hebbian philosophy is given by

$$W = W + \eta z e^T \quad (1.136)$$

$$V = V + \eta X z^T. \quad (1.137)$$

Unfortunately, this multilayer Hebbian training algorithm has not been shown to converge, and this has often been documented as leading to problems.

1.4 NN LEARNING AND CONTROL ARCHITECTURES

Neural network architectures and learning schemes are discussed in detail in Miller et al. (1991) and White and Sofge (1992). In the current literature, the NN learning schemes have been categorized into three paradigms: unsupervised learning, supervised learning, and reinforcement learning.

1.4.1 UNSUPERVISED AND REINFORCEMENT LEARNING

The unsupervised learning methods do not require an explicit teacher to guide the NN learning process. Several adaptive control schemes, for instance (Lewis et al. 1999, He and Jagannathan 2003), use unsupervised learning online. Unlike the unsupervised learning method, both supervised and reinforcement learning require a teacher to provide training signals though these methods fundamentally differ. In supervised learning, an explicit signal is provided by the teacher throughout to guide the learning process whereas in the case of reinforcement learning, the role of the teacher is more evaluative than instructional (Lewis et al. 2002). The explicit signal from the teacher is used to alter the behavior of the learning system in the case of supervised training. On the other hand, the current measure of system performance provided by the teacher in the case of reinforcement learning is not explicit. Therefore, the measure of performance does not help the learning system respond to the signal by altering its behavior. Since detailed information of the system and its behavior is not needed, unsupervised and reinforcement learning methods are potentially useful to feedback control systems.

Reinforcement learning is based on the notion that if an action is followed by a satisfactory state, or by an improvement in the state of affairs, then the tendency to produce that action should be strengthened (i.e., reinforced). Extending this idea to allow action selections to depend upon state information introduces aspects of feedback control and associative learning. The idea of adaptive critic (Werbos 1991, 1992, Barto 1992) is an extension of this general idea of

reinforcement learning. The adaptive critic NN architecture uses a critic NN in high-level supervisory capacity that critiques the system performance over time and tunes a second action NN in the feedback control loop. This two-tier structure is based on human biological structures in the cerebello-rubrospinal system. The critic NN can select either the standard Bellman equation or a simple weighted sum of tracking errors as the performance index and it tries to minimize the index. In general, the critic conveys much less information than the desired output required in supervisory learning. Nevertheless, their ability to generate correct control actions makes adaptive critics prime candidates for controlling complex nonlinear systems (Lewis et al. 2002) as presented in this book.

Tracking error-based control techniques, for instance (Lewis et al. 1999) use unsupervised training and they do not allow the designer to specify a desired performance index or a utility function. In adaptive critic NN-based methods, backpropagation algorithm is used to train the NN off-line so that the critic NN generates a suitable signal to tune the action generating NN. The adaptive critic-based NN control schemes that provide guaranteed performance analytically do not exist until now for nonlinear discrete-time systems.

1.4.2 COMPARISON OF THE TWO NN CONTROL ARCHITECTURES

Feedforward NNs are used as building blocks in both tracking error and adaptive critic-based NN architectures. In the case of tracking error-based control methodology as presented in Section 6.2.1, tracking error is used as a feedback signal to tune the NN weights online. The only objective there is to reduce the tracking error, and therefore no performance criterion is set. On the contrary, adaptive critic NN architectures use a reinforcement learning signal generated by a critic NN. The critic signal can be generated using a more complex optimization criterion such as a Bellman or Hamilton–Jacobi–Bellman equation though a simple weighted tracking error function can also be used. Consequently, adaptive critic NN architecture results in considerable computational overhead due to the addition of a second NN for generating the critic signal. It is also important to note the use of supervisor in the actor-critic architecture (Rosenstein and Barto 2004). Here the supervisor provides an additional source of evaluation feedback. Such supervised critic architecture is covered in this book. Current work is underway to eliminate the action NN without losing the functionality. In fact, in Chapter 6, a single critic NN output (also see Chapter 9) with no action NN is used to tune two action-generating NN weights in order to reduce the computational overhead. Finally, in the NN weight-tuning schemes that are presented in this book, the NNs are tuned online compared to standard work in the adaptive critic NN literature (Werbos 1991, 1992) where an explicit off-line learning phase is typically employed. In fact, providing off-line training

to the NNs would indeed enhance the rate of convergence of the controllers. However, for many real-time control applications, it is very difficult to provide desired outputs when a nonlinear function is unknown. Therefore NN control techniques normally use online tuning of weights. Finally, Lyapunov-based analysis is normally used to prove the closed-loop stability of the controller design covered in this book.

REFERENCES

- Abdallah, C.T., *Engineering Applications of Chaos*, Lecture and Personal Communication, Nov. 1995.
- Albus, J.S., A new approach to manipulator control: the cerebral model articulation controller (CMAC), *Trans. ASME J. Dynam. Syst., Meas., Contr.*, 97, 220–227, 1975.
- Barron, A.R., Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Info. Theory*, 39, 930–945, 1993.
- Barto, A.G., Reinforcement learning and adaptive critic methods, *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., pp. 469–492, Van Nostrand Reinhold, New York, 1992.
- Becker, K.H. and Dörfler, M., *Dynamical Systems and Fractals*, Cambridge University Press, Cambridge, MA, 1988.
- Commuri, S., *A Framework for Intelligent Control of Nonlinear Systems*, Ph.D. Dissertation, Department of Electrical engineering, The University of Texas at Arlington, Arlington, TX, May 1996.
- Cybenko, G., Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, 2, 303–314, 1989.
- Goodwin, C.G. and Sin, K.S., *Adaptive Filtering, Prediction, and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Haykin, S., *Neural Networks*, IEEE Press and Macmillan, New York, 1994.
- He, P. and Jagannathan, S., Adaptive critic neural network-based controller for nonlinear systems with input constraints, *Proceedings of the IEEE Conference on Decision and Control*, 2003.
- Hornik, K., Stinchcombe, M., and White, H., Multilayer feedforward networks are universal approximators, *Neural Networks*, vol. 2, pp. 359–366, 1989.
- Hush, D.R. and Horne, B.G., Progress in supervised neural networks, *IEEE Signal Proces. Mag.*, 8–39, 1993.
- Igelnik, B. and Pao, Y.H., Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, 6, 1320–1329, 1995.
- Jagannathan, S. and Lewis, F.L., Multilayer discrete-time neural network controller for a class of nonlinear system, *Proceedings of IEEE International Symposium on Intelligent Control*, Monterey, CA, Aug. 1995.
- Kim, Y.H., *Intelligent Closed-Loop Control Using Dynamic Recurrent Neural Network and Real-Time Adaptive Critic*, Ph.D. Dissertation Proposal, Department of

- Electrical engineering, The University of Texas at Arlington, Arlington, TX, Sept. 1996.
- Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice Hall, Englewood Cliffs, NJ, 1992.
- Kung, S.Y., *Digital Neural Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
- Levine, D.S., *Introduction to Neural and Cognitive Modeling*, Lawrence Erlbaum Pub., Hillsdale, NJ, 1991.
- Lewis, F.L., *Optimal Estimation*, Wiley, New York, 1986.
- Lewis, F.L. and Syrmos, V.L., *Optimal Control*, 2nd ed., Wiley, New York, 1995.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.
- Lewis, F.L., Campos, J., and Selmic, R., Neuro-fuzzy control of industrial systems with actuator nonlinearities, *SIAM*, Philadelphia, 2002.
- Lewis, F.L., Jagannathan, S., and Yesiderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, London, 1999.
- Lippmann, R.P., An introduction to computing with neural nets, *IEEE ASSP Mag.*, 4–22, 1987.
- MATLAB version 7, July 2004, The Mathworks, Inc., 24 Prime Park Way, Natick, MA.
- MATLAB Neural Network Toolbox, version 2.0, 1995, The Mathworks, Inc., 24 Prime Park Way, Natick, MA.
- Miller, W.T. III, Sutton, R.S., and Werbos, P.J., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1991.
- Narendra, K.S. and Parthasarathy, K., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, 4–27, March 1990.
- Narendra, K.S., Adaptive control using neural networks, in *Neural Networks for Control*, Miller, W.T., Sutton, R.S., Werbos, P.J., Eds., MIT Press, Cambridge, MA, pp. 115–142, 1991.
- Narendra, K.S., Adaptive control of dynamical systems using neural networks, in *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 141–183, 1991.
- Narendra, K.S. and Parthasarathy, K., Gradient methods for the optimization of dynamical systems containing neural networks, *IEEE Trans. Neural Netw.*, 2, 252–262, 1991.
- Park, J. and Sandberg, I.W., Universal approximation using radial-basis-function networks, *Neural Comp.*, 3, 246–257, 1991.
- Peretto, P., *An Introduction to the Modeling of Neural Networks*, Cambridge University Press, Cambridge, MA, 1992.
- Rosenstein, M.T. and Barto, A.G., Supervised actor-critic reinforcement learning, in *Handbook of Learning and Approximate Dynamic Programming*, Si, J., Barto, A.G., Powell, W.B., and Wunsch, D., Eds., IEEE Press, 2004.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation, in *Parallel Distributed Processing*, Rumelhart, D.E. and McClelland, J.L., Eds., MIT Press, Cambridge, MA, 1986.

- Sadegh, N., A perceptron network for functional identification and control of nonlinear systems, *IEEE Trans. Neural Netw.*, 4, 982–988, 1993.
- Sanner, R.M. and Slotine, J.-J.E., Stable adaptive control and recursive identification using radial gaussian networks, *Proceedings of IEEE Conference on Decision and Control*, Brighton, 1991.
- Simpson, P.K., Foundations of neural networks, in *Artificial Neural Networks, Paradigms, Applications, and Hardware Implementation*, Sanchez-Sinencio, E., Ed., IEEE Press, pp. 3–24, 1992.
- Weiner, N., *Cybernetics: Or Control and Communication in the Animal and the Machine*, MIT Press, Cambridge, MA, 1948.
- Werbos, P.J., *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*, Ph.D. Thesis, Committee on Applied Mathematics, Harvard University, 1974.
- Werbos, P.J., Back propagation: past and future, *Proc. 1988 Int. Conf. Neural Netw.*, 1, 1343–1353, 1989.
- White, D.A. and Sofage, D.A., Eds., *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, 1992.
- Widrow, B. and Lehr, M., Thirty years of adaptive neural networks: perceptrons, madaline and backpropagation, *Proc. IEEE*, 78, 1415–1442, 1990.

PROBLEMS

SECTION 1.1

1.1-1: *Logical operations using adaline NN.* A neuron with linear activation function is defined as ADALINE NN. The output of such an NN for two inputs is described by $y = w_1x_1 + w_2x_2 + w_0$. Select the weights to design a one-layer NN that implement: (a) logical AND operation and (b) logical OR operation.

SECTION 1.2

1.2-1: *Dynamical NN.* A dynamical NN with internal neuron dynamics is given in Figure 1.11. Write down the dynamical equations.

1.2-2: *Chaotic behavior.* Some chaotic behavior was displayed for a simple discrete-time NN. Perform some experimentation with this system, making phase-plane plots by modifying the plant and weight matrices with different activation functions.

SECTION 1.3

1.3-1: *Perceptron NN.* Write a MATLAB program to implement a one-layer perceptron network whose algorithm is given in (1.62). Rework Example 1.3.2.

1.3-2: *Backpropagation using tangent hyperbolic and RBF functions.* Derive the backpropagation algorithm using (a) tangent hyperbolic activation functions and (b) RBF activation functions. Repeat Example 1.3.3 with RBF activation functions. Compare your result with original Example 1.3.3. Use MATLAB NN tool box.

1.3-3: *Backpropagation algorithm programming.* Write your own backpropagation program in MATLAB to implement backpropagation algorithm.

2 Background and Discrete-Time Adaptive Control

In this chapter, we provide a brief background on dynamical systems, mainly covering the topics that will be important in a discussion of standard discrete-time adaptive control and neural network (NN) applications in closed-loop control of dynamical systems. It is quite common for noncontrol engineers working in NN system and control applications to have little understanding of feedback control and dynamical systems. Many of the phenomena they observe are not due to properties of NN but to properties of feedback control systems. NN applications in dynamical systems are a complex area with several facets. An incomplete understanding of any one of these can lead to incorrect conclusions being drawn, with inaccurate attributions of causes — many are convinced that often the exploratory, regulatory, and behavioral phenomena observed in NN control systems are completely due to the NN, while in fact most are due to the rather remarkable nature of feedback itself. Included in this chapter are discrete-time systems, computer simulation, norms, stability and passivity definitions, and discrete-time adaptive control (referred to as self-tuning regulators [STRs]).

2.1 DYNAMICAL SYSTEMS

Many systems in nature, including neurobiological systems, are dynamical in nature, in the sense that they are acted upon by external inputs, have internal memory, and behave in certain ways that are captured by the notion of the development of activities through time. According to the notion of systems defined by Alfred North Whitehead (1953), it is an entity distinct from its environment, whose interactions with the environment can be characterized through input and output signals. An intuitive feel for dynamic systems is provided by Luenberger (1979), which includes many examples.

2.1.1 DISCRETE-TIME SYSTEMS

If the time index is an integer k instead of a real number t , the system is said to be of discrete-time. A general class of discrete-time systems can be

described by the nonlinear ordinary difference equation in discrete-time state space form

$$x(k+1) = f(x(k), u(k)), \quad y(k) = h(x(k), u(k)) \quad (2.1)$$

where $x(k) \in \Re^n$ is the internal state vector, $u(k) \in \Re^m$ is the control input, and $y(k) \in \Re^p$ is the system output.

These equations may be derived directly from an analysis of the dynamical system or process being studied, or they may be sampled or discretized versions of continuous-time dynamics of a nonlinear system. Today, controllers are implemented in digital form by using embedded hardware making it necessary to have a discrete-time description of the controller. This may be determined by design, based on the discrete-time system dynamics. Sampling of linear systems is well understood since the work of Ragazzani and coworkers in the 1950s, with many design techniques available. However, sampling of nonlinear systems is not an easy topic. In fact, the exact discretization of nonlinear continuous dynamics is based on the Lie derivatives and leads to an infinite series representation (see e.g., Kalkkuhl and Hunt 1996). Various approximation and discretization techniques use truncated versions of the exact series.

2.1.2 BRUNOVSKY CANONICAL FORM

Let $x(k) = [x_1(k) \cdots x_n(k)]^T$, a special form of nonlinear dynamics is given by the class of systems in discrete Brunovsky canonical form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= x_3(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) \\ y(k) &= h(x(k)) \end{aligned} \quad (2.2)$$

As seen from Figure 2.1 this is a chain or cascade of unit delay elements z^{-1} , that is, a shift register. Each delay element stores information and requires an initial condition. The measured output $y(k)$ can be a general function of the states as shown, or can have more specialized forms such as

$$y(k) = h(x_1(k)) \quad (2.3)$$

The discrete Brunovsky canonical form may equivalently be written as

$$x(k+1) = Ax(k) + bf(x(k)) + bg(x(k))u(k) \quad (2.4)$$

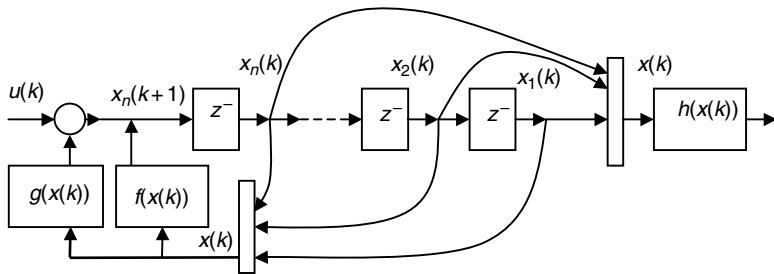


FIGURE 2.1 Discrete-time single-input Brunovsky form.

where

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \vdots & & \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad (2.5)$$

A discrete-time form of the more general version may also be written. It is a system with m -parallel chains of delay elements of lengths n_1, n_2, \dots (e.g., m shift registers), each driven by one of the control inputs.

Many practical systems occur in the continuous-time Brunovsky form. However, if a system of the continuous Brunovsky form (Lewis et al. 1999) is sampled, the result is not the general form (2.2). Under certain conditions, general discrete-time systems of the form (2.1) can be converted to discrete Brunovsky canonical form systems (see e.g., Kalkkuhl and Hunt 1996).

2.1.3 LINEAR SYSTEMS

A special and important class of dynamical systems is the discrete-time linear time invariant (LTI) system

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \quad (2.6)$$

with A, B, C constant matrices of general form (e.g., not restricted to [2.5]). An LTI is denoted by (A, B, C) . Given an initial state $x(0)$ the solution to the

LTI system can be explicitly written as

$$x(k) = A^k x(0) + \sum_{j=0}^{k-1} A^{k-j-1} B u(j) \quad (2.7)$$

The next example shows the relevance of these solutions and demonstrates that the general discrete-time nonlinear systems are even easier to simulate on a computer than continuous-time systems, as no integration routine is needed.

Example 2.1.1 (Discrete-Time System — Savings Account): Discrete-time descriptions can be derived from continuous-time systems by using Euler's approximation or system discretization theory (Lewis et al. 1999). However, many phenomena are naturally modeled using discrete-time dynamics including population growth/decline, epidemic spread, economic systems, and so on. The dynamics of the savings account using compound interest are given by the first-order system

$$x(k+1) = (1+i)x(k) + u(k)$$

where i represents the interest rate over each interval, k is the interval iteration number, and $u(k)$ is the amount of the deposit at the beginning of the k th period. The state $x(k)$ represents the account balance at the beginning of interval k .

a. Analysis

According to (2.7), if equal annual deposits are made of $u(k) = d$, the account balance is

$$x(k) = (1+i)^k x(0) + \sum_{j=0}^{k-1} (1+i)^{k-j-1} d$$

with $x(0)$ being the initial amount in the account. Using the standard series summation formula

$$\sum_{j=0}^{k-1} a^j = \frac{1-a^k}{1-a}$$

one derives

$$\begin{aligned}
 x(k) &= (1+i)^k x(0) + d(1+i)^{k-1} \sum_{j=0}^{k-1} \frac{1}{(1+i)^j} \\
 &= (1+i)^k x(0) + d(1+i)^{k-1} \left[\frac{1 - 1/(1+i)^k}{1 - 1/(1+i)} \right] \\
 &= (1+i)^k x(0) + d \left[\frac{(1+i)^{k-1} - 1}{i} \right]
 \end{aligned}$$

the standard formula for complex interest with constant annuities of d .

b. Simulation

It is very easy to simulate a discrete-time system. No numerical integration driver program is needed in contrast to the continuous-time case. Instead, a simple do loop can be used. A complete MATLAB® program that simulates the compound interest dynamics is given by

```
%Discrete-Time Simulation program for Compound
Interest~Dynamics
d=100; i=0.08; % 8% interest rate
x(1)=1000;
for k=1:100
    x(k+1)=(1+i)*x(k)
end
k=[1:101];
plot(k,x);
```

2.2 MATHEMATICAL BACKGROUND

2.2.1 VECTOR AND MATRIX NORMS

We assume the reader is familiar with norms, both vector and induced matrix norms (Lewis et al. 1993). We denote any suitable vector norm by $\|\cdot\|$. When required to be specific we denote the p -norm by $\|\cdot\|_p$. Recall that for any vector $x \in \Re^n$

$$\|x\|_1 = \sum_{i=1}^n |x_i| \quad (2.8)$$

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} \quad (2.9)$$

$$\|x\|_\infty = \max_i |x_i| \quad (2.10)$$

The 2-norm is the standard Euclidean norm.

Given a matrix A , its induced p -norm is denoted by $\|A\|_p$. Let $A = [a_{ij}]$, recall that the induced 1-norm is the maximum absolute column sum

$$\|A\|_1 = \max_i \sum_j |a_{ij}| \quad (2.11)$$

and the induced ∞ -norm is the maximum absolute row sum

$$\|A\|_\infty = \max_i \sum_j |a_{ij}| \quad (2.12)$$

The induced matrix p -norm satisfies the inequality, for any vector x ,

$$\|A\|_p \leq \|A\|_p \|x\|_p \quad (2.13)$$

and for any two matrices A, B one also has

$$\|AB\|_p \leq \|A\|_p \|B\|_p \quad (2.14)$$

Given a matrix $A = [a_{ij}]$, the Frobenius norm is defined as the root of the sum of the squares of all the elements:

$$\|A\|_F^2 \equiv \sum a_{ij}^2 = \text{tr}(A^T A) \quad (2.15)$$

with $\text{tr}(\cdot)$ the matrix trace (i.e., sum of diagonal elements). Though the Frobenius norm is not an induced norm, it is compatible with the vector 2-norm so that

$$\|Ax\|_2 \leq \|A\|_F \|x\|_2 \quad (2.16)$$

Singular value decomposition: The matrix norm $\|A\|_2$ induced by the vector 2-norm is the maximum singular value of A . For a general $m \times n$ matrix A , one may write the singular value decomposition (SVD)

$$A = U \Sigma V^T \quad (2.17)$$

where U is $m \times n$, V is $n \times n$, and both are orthogonal, that is,

$$\begin{aligned} U^T U &= UU^T = I_m \\ V^T V &= VV^T = I_n \end{aligned} \tag{2.18}$$

where I_n is the $n \times n$ identity matrix. The $m \times n$ singular value matrix has the structure

$$\Sigma = \text{diag}\{\sigma_1, \sigma_2, \dots, \sigma_r, 0, \dots, 0\} \tag{2.19}$$

where r is the rank of A and σ_i are the singular values of A . It is conventional to arrange the singular values in a nonincreasing order, so that the largest singular value is $\sigma_{\max}(A) = \sigma_1$. If A is full rank, then r is equal to either m or n , whichever is smaller. Then the minimum singular value is $\sigma_{\min}(A) = \sigma_r$ (otherwise the minimum singular value is equal to zero).

The SVD generalizes the notion of eigenvalues to general nonsquare matrices. The singular values of A are the (positive) square roots of the nonzero eigenvalues of AA^T , or equivalently $A^T A$.

Quadratic forms and definiteness: Given an $n \times n$ matrix Q the quadratic form $x^T Q x$, with x as an n -vector, will be important for stability analysis in this book. The quadratic form can in some cases have certain properties that are independent of the vector x selected. Four important definitions are:

$$\begin{aligned} Q \text{ is positive definite, denoted } Q > 0 && \text{if } x^T Q x > 0 \quad \forall x \neq 0 \\ Q \text{ is positive semidefinite, denoted } Q \geq 0 && \text{if } x^T Q x \geq 0 \quad \forall x \\ Q \text{ is negative definite, denoted } Q < 0 && \text{if } x^T Q x < 0 \quad \forall x \neq 0 \\ Q \text{ is negative semidefinite, denoted } Q \leq 0 && \text{if } x^T Q x \leq 0 \quad \forall x \end{aligned} \tag{2.20}$$

If Q is symmetric, then it is positive definite if and only if, all its eigenvalues are positive and positive semidefinite if and only if, all its eigenvalues are nonnegative. If Q is not symmetric, tests are more complicated and involve determining the minors of the matrix. Tests for negative definiteness and semidefiniteness may be found by noting that Q is negative (semi) definite if and only if $-Q$ is positive (semi) definite.

If Q is a symmetric matrix, its singular values are the magnitudes of its eigenvalues. If Q is a symmetric positive semidefinite matrix, its singular values and its eigenvalues are the same. If Q is positive semidefinite then, for any vector

x one has the useful inequality

$$\sigma_{\min}(Q)\|x\|^2 \leq x^T Q x \leq \sigma_{\max}(Q)\|x\|^2 \quad (2.21)$$

2.2.2 CONTINUITY AND FUNCTION NORMS

Given a subset $S \subset \Re^n$, a function $f(x) : S \rightarrow \Re^m$ is continuous on $x_0 \in S$ if for every $\varepsilon > 0$ there exists a $\delta(\varepsilon, x_0) > 0$ such that $\|x - x_0\| < \delta(\varepsilon, x_0)$ implies that $\|f(x) - f(x_0)\| < \varepsilon$. If δ is independent of x_0 then the function is said to be uniformly continuous. Uniform continuity is often difficult to test. However, if $f(x)$ is continuous and its derivative $f'(x)$ is bounded, then it is uniformly continuous.

A function $f(x) : \Re^n \rightarrow \Re^m$ is differentiable if its derivative $f'(x)$ exists. It is continuously differentiable if its derivative exists and is continuous. $f(x)$ is said to be locally Lipschitz if, for all $x, z \in S \subset \Re^n$, one has

$$\|f(x) - f(z)\| < L\|x - z\| \quad (2.22)$$

for some finite constant $L(S)$ where L is known as a Lipschitz constant. If $S = \Re^n$ then the function is globally Lipschitz.

If $f(x)$ is globally Lipschitz then it is uniformly continuous. If it is continuously differentiable, it is locally Lipschitz. If it is differentiable, it is continuous. For example, $f(x) = x^2$ is continuously differentiable. It is locally but not globally Lipschitz. It is continuous but not uniformly continuous.

Given a function $f(t) : [0, \infty) \rightarrow \Re^n$, according to Barbalat's Lemma, if

$$\int_0^\infty f(t)dt \leq \infty \quad (2.23)$$

and $f(t)$ is uniformly continuous, then $f(t) \rightarrow 0$ as $t \rightarrow \infty$.

Given a function $f(t) : [0, \infty) \rightarrow \Re^n$, its L_p (function) norm is given in terms of the vector norm $\|f(t)\|_p$ at each value of t by

$$\|f(\cdot)\|_p = \left(\int_0^\infty \|f(t)\|_p^p dt \right)^{1/p} \quad (2.24)$$

and if $p = \infty$

$$\|f(\cdot)\|_\infty = \sup_t \|f(t)\|_\infty \quad (2.25)$$

If the L_p norm is finite we say $f(t) \in L_p$. Note that a function is in L_∞ if, and only if, it is bounded. For detailed treatment, refer to Lewis et al. (1993, 1999).

In the discrete-time case, let $Z_+ = \{0, 1, 2, \dots\}$ be the set of natural numbers and $f(k) : Z_+ \rightarrow \Re^n$. The l_p (function) norm is given in terms of the vector $\|f(k)\|_p$ at each value of k by

$$\|f(\cdot)\|_p = \left(\sum_{k=0}^{\infty} \|f(k)\|_p^p \right)^{1/p} \quad (2.26)$$

and if $p = \infty$

$$\|f(\cdot)\|_{\infty} = \sup_k \|f(k)\|_{\infty} \quad (2.27)$$

If the l_p norm is finite, we say $f(k) \in l_p$. Note that a function is in l_{∞} if, and only if, it is bounded.

2.3 PROPERTIES OF DYNAMICAL SYSTEMS

In this section are discussed some properties of dynamical systems, including stability and passivity. For observability and controllability, please refer to Goodwin and Sin (1984) and Astrom and Wittenmark (1989). If the original open-loop system is controllable and observable, then feedback control system can be designed to meet desired performance. If the system has certain passivity properties, this design procedure is simplified and additional closed-loop properties such as robustness can be guaranteed. On the other hand, properties such as stability may not be present in the original open-loop system, but are design requirements for closed-loop performance.

2.3.1 STABILITY

Stability, along with robustness (see Subsection 2.4.4), is a performance requirement for closed-loop systems. In other words, though the open-loop stability properties of the original system may not be satisfactory, it is desired to design a feedback control system such that the closed-loop stability is adequate. We will discuss stability for discrete-time systems, but the same definitions also hold for continuous-time systems with obvious modifications.

Consider the dynamical system

$$x(k + 1) = f(x(k), k) \quad (2.28)$$

where $x(k) \in \Re^n$, which might represent either an uncontrolled open-loop system, or a closed-loop system after the control input $u(k)$ has been specified

in terms of the state $x(k)$. Let the initial time be k_0 , and the initial condition be $x(k_0) = x_0$. This system is said to be nonautonomous since the time k appears explicitly. If k does not appear explicitly in $f(\cdot)$, then system is autonomous. A primary cause of explicit time dependence in control systems is the presence of time-dependent disturbances $d(k)$.

A state x_e is an equilibrium point of the system $f(x_e, k) = 0$, $k \geq k_0$. If $x_0 = x_e$, so that the system starts out in the equilibrium state, then it will forever remain there. For linear systems, the only possible equilibrium point is $x_e = 0$; for nonlinear systems, x_e may be nonzero. In fact, there may be an equilibrium set, such as a limit cycle.

Asymptotic stability: An equilibrium point x_e is locally asymptotically stable (AS) at k_0 if there exists a compact set $S \subset \Re^n$ such that, for every initial condition $x_0 \in S$, one has $\|x(k) - x_e\| \rightarrow 0$ as $k \rightarrow \infty$. That is, the state $x(k)$ converges to x_e . If $S = \Re^n$ so that $x(k) \rightarrow x_e$ for all $x(k_0)$, then x_e is said to be globally asymptotically stable (GAS) at k_0 . If the conditions hold for all k_0 , the stability is said to be uniform (e.g., UAS, GUAS).

Asymptotic stability is a very strong property that is extremely difficult to achieve in closed-loop systems, even using advanced feedback controller design techniques. The primary reason is the presence of unknown but bounded system disturbances. A milder requirement is provided as follows:

Lyapunov stability: An equilibrium point x_e is stable in the sense of Lyapunov (SISL) at k_0 if for every $\varepsilon > 0$ there exists $\delta(\varepsilon, x_0)$ such that $\|x_0 - x_e\| < \delta(\varepsilon, k_0)$ implies that $\|x(k) - x_e\| < \varepsilon$ for $k \geq k_0$. The stability is said to be uniform (e.g., uniformly SISL) if $\delta(\cdot)$ is independent of k_0 ; that is, the system is SISL for all k_0 .

It is extremely interesting to compare these definitions to those of function continuity and uniform continuity. SISL is a notion of continuity for dynamical systems. Note that for SISL there is a requirement that the state $x(k)$ be kept arbitrarily close to x_e by starting sufficiently close to it. This is still too strong a requirement for closed-loop control in the presence of unknown disturbances. Therefore, a practical definition of stability to be used as a performance objective for feedback controller design in this book is as follows:

Boundedness: This is illustrated in Figure 2.2. The equilibrium point x_e is said to be uniformly ultimately bounded (UUB) if there exists a compact set $S \subset \Re^n$ so that for all $x_0 \in S$ there exists a bound $\mu \geq 0$, and a number $N(\mu, x_0)$ such that $\|x(k)\| \leq \mu$ for all $k \geq k_0 + N$. The intent here is to capture the notion that for all initial states in the compact set S , the system trajectory eventually reaches, after a lapsed time of N , a bounded neighborhood of x_e .

The difference between UUB and SISL is that in UUB the bound μ cannot be made arbitrarily small by starting closer to x_e . In fact, the Vander Pol oscillator is UUB but not SISL. In practical closed-loop applications, μ depends on the

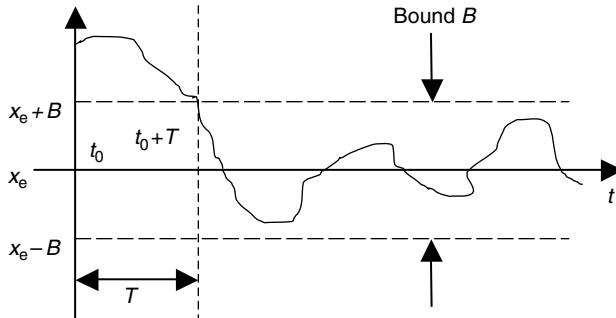


FIGURE 2.2 Illustration of UUB.

disturbance magnitudes and other factors. If the controller is suitably designed, however, μ will be small enough for practical purposes. The term *uniform* indicates that N does not depend upon k_0 . The term *ultimate* indicates that the boundedness property holds after a time lapse N . If $S = \Re^n$, the system is said to be globally UUB (GUUB).

A note on autonomous systems and linear systems: If the system is autonomous so that

$$x(k+1) = f(x(k)) \quad (2.29)$$

where $f(x(k))$ is not an explicit function of time, the state trajectory is independent of the initial time. This means that if an equilibrium point is stable by any of the three definitions, the stability is automatically uniform. Nonuniformity is only a problem with nonautonomous systems.

If the system is linear so that

$$x(k+1) = A(k)x(k) \quad (2.30)$$

with $A(k)$ is an $n \times n$ matrix, then the only possible equilibrium point is the origin.

For LTI systems, matrix A is time-invariant. Then, the system poles are given by the roots of the characteristic equation

$$\Delta(z) = |zI - A| = 0 \quad (2.31)$$

where $|\cdot|$ is the matrix determinant and z is the Z transform variable. For LTI systems, AS corresponds to the requirement that all the system poles stay within the unit disc (i.e., none of them are allowed on the unit disc). SISL corresponds

to marginal stability, that is, all the poles are within the unit disc and those on the unit disc are not repeated.

2.3.2 PASSIVITY

The passivity notions defined here are used later in Lyapunov proofs of stability. Discrete-time Lyapunov proofs are considerably more complex than their continuous-time counterparts; therefore, the required passivity notions on discrete-time are more complex.

Some aspects of passivity (Goodwin and Sin 1984) will subsequently be important. The set of time instants of interest is $Z_+ = \{0, 1, 2, \dots\}$. Consider the Hilbert space $l_2^n(Z_+)$ of sequences $y : \mathbb{N}_+ \rightarrow \mathbb{R}^n$ with inner product $\langle \cdot, \cdot \rangle$ defined by

$$\langle y, u \rangle = \sum_{k=0}^{\infty} y^T(k)u(k)$$

A norm on $l_2^n(Z_+)$ is defined by $\|u\| = \sqrt{\langle u, u \rangle}$. Let P_T denote the operator that truncates the signal u at time T :

$$P_T = \begin{cases} u(k), & (k < T) \\ 0, & (k \geq T) \end{cases}$$

The basic signal space $l_{2e}^n(Z_+)$ is given by an extension of $l_2^n(Z_+)$ according to

$$l_{2e}^n(Z_+) = \{u : Z_+ \rightarrow \mathbb{R}^n \mid \forall T \in Z_+, P_T u \in l_2^n(Z_+)\}$$

It is convenient to use the notation $u_T = P_T u$ and $\langle y, u \rangle_T = \langle y_T, u_T \rangle$.

Define the energy supply function $E : l_{2e}^n(Z_+) \times l_{2e}^n(Z_+) \times Z_+ \rightarrow \mathbb{R}$. A useful energy function E is defined here in quadratic form as

$$E(u, y, T) = \langle y, Su \rangle_T + \langle u, Ru \rangle_T$$

with S and R as appropriately defined matrices. Define the first difference of a function $L(k) : Z_+ \rightarrow \mathbb{R}$ as

$$\Delta L(k) \equiv L(k+1) - L(k) \quad (2.32)$$

A discrete-time system (e.g., [2.8]) with input $u(k)$ and output $y(k)$ is said to be *passive* if it verifies an equality of the power form

$$\Delta L(k) = y^T(k)Su(k) + u^T(k)Ru(k) - g(k) \quad (2.33)$$

for some $L(k)$ that is lower bounded, some function $g(k) \geq 0$, and appropriately defined matrices R and S . That is

$$\sum_{k=0}^T (y^T(k)Su(k) + u^T(k)Ru(k)) \geq \sum_{k=0}^T g(k) - \gamma^2 \quad (2.34)$$

for all $T \geq 0$ and some $\gamma \geq 0$. In other words,

$$E(u, y, T) \geq \sum_{k=0}^T g(k) - \gamma_0^2, \quad \forall T \geq 0.$$

We say the system is dissipative if it is passive and in addition

$$\begin{aligned} E(u, y, T) \neq 0 \Rightarrow \sum_{k=0}^T (y^T(k)Su(k) + u^T(k)Ru(k)) \neq 0 & \text{ implies} \\ \sum_{k=0}^T g(k) > 0 \end{aligned} \quad (2.35)$$

for all $T \geq 0$.

A special sort of dissipativity occurs if $g(k)$ is a quadratic function of $\|x(k)\|$ with bounded coefficients, where $x(k)$ is the internal state of the system. We call this state strict passivity (SSP). Then

$$\sum_{k=0}^T (y^T(k)Su(k) + u^T(k)Ru(k)) \geq \sum_{k=0}^T (\|x(k)\|^2 + \text{LOT}) - \gamma^2 \quad (2.36)$$

for all $T \geq 0$ and some $\gamma \geq 0$ where LOT denotes lower-order terms in $\|x(k)\|$. Then, the l_2 norm of the state is overbounded in terms of the l_2 inner product of output and input (i.e., the power delivered to the system). We use SSP to conclude some internal boundedness properties of the system without the usual assumption of observability (e.g., persistence of excitation) that is required in standard adaptive control approaches.

2.3.3 INTERCONNECTIONS OF PASSIVE SYSTEMS

To get an indication of the importance of passivity, consider two passive systems placed into a feedback configuration as shown in Figure 2.3. Then,

$$\begin{aligned} \Delta L_1 &= y_1^T(k)u_1(k) - g_1(k) \\ \Delta L_2 &= y_2^T(k)u_2(k) - g_2(k) \\ u_1(k) &= u(k) - y_2(k) \\ u_2(k) &= y_1(k) \end{aligned} \quad (2.37)$$

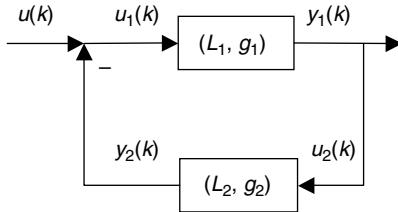


FIGURE 2.3 Two passive systems in feedback interconnection.

and it is very easy to verify that

$$\Delta(L_1 + L_2) = y_1^T(k)u(k) - (g_1(k) + g_2(k)) \quad (2.38)$$

That is, the feedback configuration is also in power form and hence passive. Properties that are preserved under feedback are extremely important for controller design.

If both systems in Figure 2.3 are state strict passive, then the closed-loop system is SSP. However, if only one subsystem is SSP and the other only passive, the combination is only passive and not generally SSP.

It also turns out that parallel combinations of systems in power form are still in power form. Series interconnection does not generally preserve passivity.

2.4 NONLINEAR STABILITY ANALYSIS AND CONTROLS DESIGN

For LTI systems it is straightforward to investigate stability by examining the locations of the poles in the s -plane. However, for nonlinear or nonautonomous (e.g., time-varying) systems there are no direct techniques. The (direct) Lyapunov approach provides methods for studying the stability of nonlinear systems and shows how to design control systems for such complex nonlinear systems. For more information see (Lewis et al. 1993), which deals with robot manipulator control, as well as (Landau 1979; Goodwin and Sin 1984; Sastry and Bodson 1989; Slotine and Li 1991), which have proofs and many excellent examples in continuous and discrete-time.

2.4.1 LYAPUNOV ANALYSIS FOR AUTONOMOUS SYSTEMS

The autonomous (time-invariant) dynamical system

$$x(k+1) = f(x(k)) \quad (2.39)$$

$x \in \mathbb{R}^n$, could represent a closed-loop system after the controller has been designed. In Section 2.3.1 we defined several types of stability. We shall show here how to examine stability properties using a generalized energy approach. An isolated equilibrium point x_e can always be brought to the origin by redefinition of coordinates; therefore, let us assume without loss of generality that the origin is an equilibrium point. First, we give some definitions and results. Then some examples are presented to illustrate the power of the Lyapunov approach.

Let $L(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ be a scalar function such that $L(0) = 0$, and S be a compact subset of \mathbb{R}^n . Then $L(x)$ is said to be

- Locally positive definite if $L(x) > 0$ when $x \neq 0$, for all $x \in S$.
(Denoted $L(x) > 0$.)
- Locally positive semidefinite if $L(x) \geq 0$ when $x \neq 0$, for all $x \in S$.
(Denoted $L(x) \geq 0$.)
- Locally negative definite if $L(x) < 0$ when $x \neq 0$, for all $x \in S$.
(Denoted $L(x) < 0$.)
- Locally negative semidefinite if $L(x) \leq 0$ when $x \neq 0$, for all $x \in S$.
(Denoted $L(x) \leq 0$.)

An example of a positive definite function is the quadratic form $L(x) = x^T Px$, where P is any matrix that is symmetric and positive definite. A definite function is allowed to be zero only when $x = 0$, a semidefinite function may vanish at points where $x \neq 0$. All these definitions are said to hold globally if $S = \mathbb{R}^n$.

A function $L(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ with continuous partial differences (or derivatives) is said to be a Lyapunov function for the system (2.39), if, for some compact set $S \subset \mathbb{R}^n$, one has locally:

$$L(x) \text{ is positive definite,} \quad L(x) > 0 \quad (2.40)$$

$$\Delta L(x) \text{ is negative semidefinite,} \quad \Delta L(x) \leq 0 \quad (2.41)$$

where $\Delta L(x)$ is evaluated along the trajectories of (2.39) (as shown in an upcoming example). That is,

$$\Delta L(x(k)) = L(x(k+1)) - L(x(k)) \quad (2.42)$$

Theorem 2.4.1 (Lyapunov Stability): If there exists a Lyapunov function for a system (2.39), then the equilibrium point is SISL.

This powerful result allows one to analyze stability using a generalized notion of energy. The Lyapunov function performs the role of an energy function. If $L(x)$ is positive definite and its derivative is negative semidefinite, then $L(x)$ is nonincreasing, which implies that the state $x(t)$ is bounded. The next

result shows what happens if the Lyapunov derivative is negative definite — then $L(x)$ continues to decrease until $\|x(k)\|$ vanishes.

Theorem 2.4.2 (Asymptotic Stability): If there exists a Lyapunov function $L(x)$ for system (2.39) with the strengthened condition on its derivative

$$\Delta L(x) \text{ is negative definite, } \Delta L(x) < 0 \quad (2.43)$$

then the equilibrium point is AS.

To obtain global stability results, one needs to expand the set S to all of \Re^n , but also required is an additional radial unboundedness property.

Theorem 2.4.3 (Global Stability):

- a. Globally SISL: If there exists a Lyapunov function $L(x)$ for the system (2.39) such that (2.40) and (2.41) hold globally and

$$L(x) \rightarrow \infty \quad \text{as } \|x\| \rightarrow \infty \quad (2.44)$$

then the equilibrium point is globally SISL.

- b. Globally AS: If there exists a Lyapunov function $L(x)$ for a system (2.39) such that (2.40) and (2.43) hold globally and also the unboundedness condition (2.44) holds, then the equilibrium point is GAS.

The global nature of this result of course implies that the equilibrium point mentioned is the *only* equilibrium point.

The next examples show the utility of the Lyapunov approach and make several points. Among the points of emphasis are those stating that the Lyapunov function is intimately related to the energy properties of a system, and that Lyapunov techniques are closely related to the passivity notions in Section 2.3.2.

Example 2.4.1 (Local and Global Stability):

- a. *Local Stability*

Consider the system

$$\begin{aligned} x_1(k+1) &= x_1(k) \left(\sqrt{x_1^2(k) + x_2^2(k) - 2} \right) \\ x_2(k+1) &= x_2(k) \left(\sqrt{x_1^2(k) + x_2^2(k) - 2} \right) \end{aligned}$$

Stability for nonlinear discrete-time systems can be examined by selecting the quadratic Lyapunov function candidate

$$L(x(k)) = x_1^2(k) + x_2^2(k)$$

which is a direct realization of an energy function and has first difference

$$\Delta L(x(k)) = x_1^2(k+1) - x_1^2(k) + x_2^2(k+1) - x_2^2(k)$$

Evaluating this along the system trajectories simply involves substituting the state differences from the dynamics to obtain, in this case,

$$\Delta L(x(k)) = -(x_1^2(k) + x_2^2(k))(1 - x_1^2(k) - x_2^2(k))$$

which is negative as long as

$$\|x(k)\| = \sqrt{x_1^2(k) + x_2^2(k)} < 1$$

Therefore, $L(x(k))$ serves as a (local) Lyapunov function for the system, which is locally AS. The system is said to have a domain of attraction with a radius of one. Trajectories beginning outside $\|x(k)\| = 1$ in the phase plane cannot be guaranteed to converge.

b. *Global Stability*

Consider now the system

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

where the states satisfy $(x_1(k)x_2(k))^2 < 1$.

Selecting the Lyapunov function candidate

$$L(x(k)) = x_1^2(k) + x_2^2(k)$$

which is a direct realization of an energy function and has first difference

$$\Delta L(x(k)) = x_1^2(k+1) - x_1^2(k) + x_2^2(k+1) - x_2^2(k)$$

Evaluating this along the system trajectories simply involves substituting the state differences from the dynamics to obtain, in this case,

$$\Delta L(x(k)) = -(x_1^2(k) + x_2^2(k))(1 - x_1^2(k)x_2^2(k))$$

Applying the constraint, the system is globally stable since the states are restricted.

Example 2.4.2 (Lyapunov Stability): Consider now the system

$$\begin{aligned} x_1(k+1) &= x_1(k) - x_2(k) \\ x_2(k+1) &= \sqrt{2x_1(k)x_2(k) - x_1^2(k)} \end{aligned}$$

Selecting the Lyapunov function candidate

$$L(x(k)) = x_1^2(k) + x_2^2(k)$$

which is a direct realization of an energy function and has first difference

$$\Delta L(x(k)) = x_1^2(k+1) - x_1^2(k) + x_2^2(k+1) - x_2^2(k)$$

Evaluating this along the system trajectories simply involves substituting the state differences from the dynamics to obtain, in this case,

$$\Delta L(x(k)) = -x_1^2(k)$$

This is only negative semidefinite (note that $\Delta L(x(k))$ can be zero when $x_2(k) \neq 0$). Therefore, $L(x(k))$ is a Lyapunov function, but the system is only shown by this method to be SISL — that is, $\|x_1(k)\|, \|x_2(k)\|$ are both bounded.

2.4.2 CONTROLLER DESIGN USING LYAPUNOV TECHNIQUES

Though we have presented Lyapunov analysis only for unforced systems in the form (2.39), which have no control input, these techniques also provide a powerful set of tools for designing feedback control systems of the form

$$x(k+1) = f(x(k)) + g(x(k))u(k) \quad (2.45)$$

Thus, select a Lyapunov function candidate $L(x) > 0$ and differentiate along the system trajectories to obtain

$$\begin{aligned}\Delta L(x) &= L(x(k+1)) - L(x(k)) = x^T(k+1)x(k+1) - x^T(k)x(k) \\ &= (f(x(k)) + g(x(k))u(k))^T(f(x(k)) + g(x(k))u(k)) - x^T(k)x(k)\end{aligned}\quad (2.46)$$

Then, it is often possible to ensure that $\Delta L \leq 0$ by appropriate selection of $u(k)$. When this is possible, it generally yields controllers in state-feedback form, that is, where $u(k)$ is a function of the states $x(k)$.

Practical systems with actuator limits and saturation often contain discontinuous functions including the signum function defined for scalars $x \in \Re$ as

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (2.47)$$

shown in Figure 2.4, and for vectors $x = [x_1 \ x_2 \ \dots \ x_n]^T \in \Re^n$ as

$$\text{sgn}(x) = [\text{sgn}(x_i)] \quad (2.48)$$

where $[z_i]$ denotes a vector z with components z_i . The discontinuous nature of such functions often makes it impossible to apply input/output feedback linearization where differentiation is required. In some cases, controller design can be carried out for systems containing discontinuities using Lyapunov techniques.

Example 2.4.3 (Controller Design by Lyapunov Analysis): Consider the system

$$\begin{aligned}x_1(k+1) &= x_2(k)\text{sgn}(x_1(k)) \\ x_2(k+1) &= \sqrt{x_1(k)x_2(k) + u(k)}\end{aligned}$$

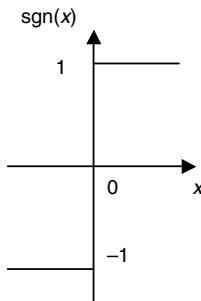


FIGURE 2.4 Signum function.

having an actuator nonlinearity. A control input has to be designed using feedback linearization techniques (i.e., cancels all nonlinearities). A stabilizing controller can be easily designed using Lyapunov techniques.

Select the Lyapunov function candidate

$$L(x(k)) = x_1^2(k) + x_2^2(k)$$

and evaluate

$$\Delta L(x(k)) = x_1^2(k+1) - x_1^2(k) + x_2^2(k+1) - x_2^2(k)$$

Substituting the system dynamics in the above equation results in

$$\Delta L(x(k)) = x_2^2(k) \operatorname{sgn}^2(x_1(k)) - x_1^2(k) + (x_1(k)x_2(k) + u(k)) - x_2^2(k)$$

Now select the feedback control

$$u(k) = -x_2^2(k) \operatorname{sgn}^2(x_1(k)) + x_1^2(k) - x_1(k)x_2(k)$$

This yields,

$$\Delta L(x(k)) = -x_2^2(k)$$

so that $L(x(k))$ is rendered a (closed-loop) Lyapunov function. Since $\Delta L(x(k))$ is negative semidefinite, the closed-loop system with this controller is SISL.

It is important to note that by slightly changing the controller, one can also show global asymptotic stability of the closed-loop system. Moreover, note that this controller has elements of feedback linearization (discussed in the Chapter 3) in that the control input $u(k)$ is selected to cancel nonlinearities. However, no difference of the right-hand side of the state equation is needed in the Lyapunov approach except that the right-hand side becomes quadratic, which makes it hard to design controllers and show stability. This will be a problem for the discrete-time systems and we will be presenting how to select suitable Lyapunov function candidates for complex systems when standard adaptive control and NN-based controllers are deployed. Finally, there are some issues in this example, such as the selection of the discontinuous control signal, which could cause chattering. In practice, the system dynamics act as a low-pass filter, so that the controllers work well.

Lyapunov analysis and controls design for linear systems: For general nonlinear systems it is not always easy to find a Lyapunov function. Thus, failure to find a Lyapunov function may be because the system is not stable, or because

the designer simply lacks insight and experience. However, in the case of LTI systems

$$x(k+1) = Ax \quad (2.49)$$

Lyapunov analysis is simplified, and a Lyapunov function is easy to find, if one exists.

Stability analysis: Select as a Lyapunov function candidate the quadratic form

$$L(x(k)) = \frac{1}{2}x^T(k)Px(k) \quad (2.50)$$

where P is a constant symmetric positive definite matrix. Since $P > 0$, then $x^T Px$ is a positive function. This function is a generalized norm, which serves as a system energy function. Then,

$$\Delta L(x(k)) = L(x(k+1)) - L(x(k)) = \frac{1}{2}[x^T(k+1)Px(k+1) - x^T(k)Px(k)] \quad (2.51)$$

$$= \frac{1}{2}x^T(k)[A^T PA - P]x(k) \quad (2.52)$$

For stability one requires negative semidefiniteness. Thus, there must exist a symmetric positive semidefinite matrix Q such that

$$\Delta L(x) = -x^T(k)Qx(k) \quad (2.53)$$

This results in the next theorem.

Theorem 2.4.4 (Lyapunov Theorem for Linear Systems): The system (2.49) is SISL, if there exist matrices $P > 0$, $Q \geq 0$ that satisfy the Lyapunov equation

$$A^T PA - P = -Q \quad (2.54)$$

If there exists a solution such that both P and Q are positive definite, the system is AS.

It can be shown that this theorem is both necessary and sufficient. That is, for LTI systems, if there is no Lyapunov function of the quadratic form (2.50), then there is no Lyapunov function. This result provides an alternative to examining the eigenvalues of the A matrix.

Lyapunov design of LTI feedback controllers: These notions offer a valuable procedure for LTI control system design. Note that the closed-loop system with state feedback

$$x(k+1) = Ax(k) + Bu(k) \quad (2.55)$$

$$u = -Kx \quad (2.56)$$

is SISL if, and only if, there exist matrices $P > 0$, $Q \geq 0$ that satisfy the closed-loop Lyapunov equation

$$(A - BK)^T P (A - BK) - P = -Q \quad (2.57)$$

If there exists a solution such that both P and Q are positive definite, the system is AS.

Now suppose there exist $P > 0$, $Q > 0$ that satisfy the Riccati equation

$$P(k) = A^T P(k+1) (I + BR^{-1}B^T P(k+1))^{-1} A + Q \quad (2.58)$$

Select now the feedback gain as

$$K(k) = -(R + B^T P(k+1)B)^{-1} B^T P(k+1) A \quad (2.59)$$

and the control input as

$$u(k) = -K(k)x(k) \quad (2.60)$$

for some matrix $R > 0$.

These equations verify that this selection of the control input guarantees closed-loop asymptotic stability.

Note that the Riccati equation depends only on known matrices — the system (A, B) and two symmetric design matrices Q, R that need to be selected positive definite. There are many good routines that can find the solution P to this equation provided that (A, B) is controllable (e.g., MATLAB). Then, a stabilizing gain is given by (2.59). If different design matrices Q, R are selected, different closed-loop poles will result. This approach goes far beyond classical frequency domain or root locus design techniques in that it allows the determination of stabilizing feedbacks for complex multivariable systems by simply solving a matrix design equation. For more details on this linear quadratic (LQ) design technique see Lewis and Syrmos (1995).

2.4.3 LYAPUNOV ANALYSIS FOR NONAUTONOMOUS SYSTEMS

We now consider nonautonomous (time-varying) dynamical systems of the form

$$x(k+1) = f(x(k), k), \quad k \geq k_0 \quad (2.61)$$

$x \in \Re^n$. Assume again that the origin is an equilibrium point. For nonautonomous systems the basic concepts just introduced still hold, but the explicit time dependence of the system must be taken into account. The basic issue is that the Lyapunov function may now depend on time. In this situation, the definitions of definiteness must be modified, and the notion of decrescence is needed.

Let $L(x(k), k) : \Re^n \times \Re \rightarrow \Re$ be a scalar function such that $L(0, k) = 0$, and S be a compact subset of \Re^n . Then $L(x(k), k)$ is said to be

- Locally positive definite if $L(x(k), k) \geq L_0(x(k))$ for some time-invariant positive definite $L_0(x(k))$, for all $k \geq 0$ and $x \in S$. (Denoted $L(x(k), k) > 0$.)
- Locally positive semidefinite if $L(x(k), k) \geq L_0(x(k))$ for some time-invariant positive semidefinite $L_0(x(k))$, for all $k \geq 0$ and $x \in S$. (Denoted $L(x(k), k) \geq 0$.)
- Locally negative definite if $L(x(k), k) \leq L_0(x(k))$ for some time-invariant negative definite $L_0(x(k))$, for all $k \geq 0$ and $x \in S$. (Denoted $L(x(k), k) < 0$.)
- Locally negative semidefinite if $L(x(k), k) \leq L_0(x(k))$ for some time-invariant negative semidefinite $L_0(x(k))$, for all $k \geq 0$ and $x \in S$. (Denoted $L(x(k), k) \leq 0$.)

Thus, for definiteness of time-varying functions, a time-invariant definite function must be dominated. All these definitions are said to hold globally if $S \in \Re^n$.

A time-varying function $L(x(k), k) : \Re^n \times \Re \rightarrow \Re$ is said to be decrescent if $L(0, k) = 0$, and there exists a time-invariant positive definite function $L_1(x(k))$ such that

$$L(x(k), k) \leq L_1(x(k)), \quad \forall k \geq 0 \quad (2.62)$$

The notions of decrescence and positive definiteness for time-varying functions are depicted in Figure 2.5.

Example 2.4.4 (Decrescent Function): Consider the time-varying function

$$L(x(k), k) = x_1^2(k) + \frac{x_2^2(k)}{3 + \sin kT}$$

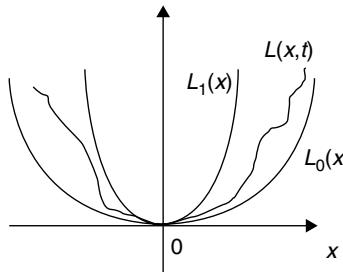


FIGURE 2.5 Time-varying function $L(x(k), k)$ that is positive definite ($L_0(x(k)) < L(x(k), k)$) and decrescent ($L(x(k), k) \leq L_1(x(k))$).

Note that $2 \leq 3 + \sin kT \leq 4$, so that

$$L(x(k), k) \geq L_0(x(k)) \equiv x_1^2(k) + \frac{x_2^2(k)}{4}$$

and $L(x(k), k)$ is globally positive definite. Also,

$$L(x(k), k) \leq L_1(x(k)) \equiv x_1^2(k) + x_2^2(k)$$

so that it is decrescent.

Theorem 2.4.5 (Lyapunov Results for Nonautonomous Systems):

- a. *Lyapunov Stability:* If, for system (2.61), there exists a function $L(x(k), k)$ with continuous partial derivatives, such that for x in a compact set $S \subset \Re^n$

$$L(x(k), k) \text{ is positive definite,} \quad L(x(k), k) > 0 \quad (2.63)$$

$$\Delta L(x(k), k) \text{ is negative semidefinite,} \quad \Delta L(x(k), k) \leq 0 \quad (2.64)$$

then the equilibrium point is SISL.

- b. *Asymptotic Stability:* If, furthermore, condition (2.64) is strengthened to

$$\Delta L(x(k), k) \text{ is negative definite,} \quad \Delta L(x(k), k) < 0 \quad (2.65)$$

then the equilibrium point is AS.

- c. *Global Stability*: If the equilibrium point is SISL or AS, if $S = \mathbb{R}^n$ and in addition the radial unboundedness condition holds:

$$L(x(k), k) \rightarrow \infty \quad \text{as } \|x(k)\| \rightarrow \infty, \quad \forall k \quad (2.66)$$

then the stability is global.

- d. *Uniform Stability*: If the equilibrium point is SISL or AS, and in addition $L(x(k), k)$ is decrescent (e.g., [2.62] holds), then the stability is uniform (e.g., independent of k_0).

The equilibrium point may be both uniformly and globally stable — for example, if all the conditions of the theorem hold, then one has GUAS.

2.4.4 EXTENSIONS OF LYAPUNOV TECHNIQUES AND BOUNDED STABILITY

The Lyapunov results so far presented have allowed the determination of SISL, if there exists a function such that $L(x(k), k) > 0$, $\Delta L(x(k), k) \leq 0$, and AS, if there exists a function such that $L(x(k), k) > 0$, $\Delta L(x(k), k) < 0$. Various extensions of these results allow one to determine more about the stability properties by further examining the deeper structure of the system dynamics.

UUB analysis and controls design: We have seen how to demonstrate that a system is SISL or AS using Lyapunov techniques. However, in practical applications there are often unknown disturbances or modeling errors, which makes it hard to expect even SISL for a closed-loop system. Typical examples are systems of the form

$$x(k+1) = f(x(k), k) + d(k) \quad (2.67)$$

with $d(k)$ an unknown but bounded disturbance. A more practical notion of stability is UUB. The next result shows that UUB is guaranteed if the Lyapunov derivative is negative outside some bounded region of \mathbb{R}^n .

Theorem 2.4.6 (UUB by Lyapunov Analysis): If, for system (2.67), there exists a function $L(x, k)$ with continuous partial differences such that for x in a compact set $S \subset \mathbb{R}^n$

$$\begin{aligned} L(x(k), k) \text{ is positive definite,} \quad L(x(k), k) &> 0 \\ \Delta L(x(k), k) < 0 & \quad \text{for } \|x\| > R \end{aligned}$$

for some $R > 0$ such that the ball of radius R is contained in S , then the system is UUB and the norm of the state is bounded to within a neighborhood of R .

In this result note that ΔL must be strictly less than zero outside the ball of radius R . If one only has $\Delta L(x(k), k) \leq 0$ for all $\|x\| > R$, then nothing may be concluded about the system stability.

For systems that satisfy the theorem, there may be some disturbance effects that push the state away from the equilibrium. However, if the state becomes too large, the dynamics tend to pull it back toward the equilibrium. Due to these two opposing effects that balance when $\|x\| \approx R$, the time histories tend to remain in the vicinity of $\|x\| = R$. In effect, the norm of the state is effectively or practically bounded by R .

The notion of the ball outside which ΔL is negative should not be confused with that of domain of attraction — in Example 2.4.1a. It was shown there that the system is AS as long as one has $\|x_0\| < 1$, defining a domain of attraction of radius one.

The next example shows how to use this result. They make the point that it can also be used as a control design technique where the control input is selected to guarantee that the conditions of the theorem hold.

Example 2.4.5 (UUB of Linear Systems with Disturbance): It is common in practical systems to have unknown disturbances, which are often bounded by some known amount. Such disturbances result in UUB and require the UUB extension for analysis. Suppose the system

$$x(k+1) = Ax(k) + d(k)$$

has A stable and a disturbance $d(k)$ that is unknown but bounded so that $\|d(k)\| < d_M$, with the bound d_M known.

Select the Lyapunov function candidate

$$L(x(k)) = x^T(k)Px(k)$$

and evaluate

$$\begin{aligned} \Delta L(x(k)) &= x^T(k+1)Px(k+1) - x^T(k)Px(k) \\ &= x^T(k)(A^T PA - P)x(k) + 2x^T(k)A^T Pd(k) + d^T(k)Pd(k) \\ &= -x^T(k)Qx(k) + 2x^T(k)A^T Pd(k) + d^T(k)Pd(k) \end{aligned}$$

where (P, Q) satisfy the Lyapunov equation

$$A^T PA - P = -Q$$

One may now use the norm equalities to write

$$\begin{aligned}\Delta L(x(k)) &\leq -[\sigma_{\min}(Q)\|x(k)\|^2 - 2\|x(k)\|\sigma_{\max}(A^T P)\|d(k)\| \\ &\quad - \sigma_{\max}(P)\|d(k)\|^2]\end{aligned}$$

which is negative as long as

$$\|x(k)\| \geq \frac{\sigma_{\max}(A^T P)d_M + \sqrt{\sigma_{\max}^2(A^T P)d_M^2 + \sigma_{\min}(Q)\sigma_{\max}(P)d_M^2}}{\sigma_{\min}(Q)}$$

Thus, if the disturbance magnitude-bound increases, the norm of the state will also increase.

Example 2.4.6 (UUB of Closed-Loop System): The UUB extension can be utilized to design stable closed-loop systems. The system described by

$$x(k+1) = x^2(k) - 10x(k) \sin x(k) + d(k) + u(k)$$

is excited by an unknown disturbance whose magnitude is bounded so that $\|d(k)\| < d_M$. To find a control that stabilizes the system and mitigates the effect of disturbances, select the control input as,

$$u(k) = -x^2(k) + 10x(k) \sin x(k) + k_v x(k)$$

This helps cancel the sinusoidal nonlinearity and provides a stabilizing term yielding the closed-loop system

$$x(k+1) = k_v x(k) + d(k).$$

Select the Lyapunov function candidate

$$L(x(k)) = x^2(k)$$

whose first difference is given by

$$\Delta L(x(k)) = x^2(k+1) - x^2(k)$$

Evaluating the first difference along the closed-loop system trajectories yields

$$\Delta L(x(k)) \leq -x^2(k)(1 - k_{v \max}^2) - 2x(k)k_v d(k) + d^2(k)$$

which is negative as long as

$$\|x(k)\| > \frac{k_{v \max} d_M + \sqrt{k_{v \max}^2 d_M^2 + (1 - k_{v \max}^2) d_M^2}}{(1 - k_{v \max}^2)}$$

which after simplification results in

$$\|x(k)\| > \frac{(1 + k_{v \max})}{(1 - k_{v \max}^2)} d_M$$

The UUB bound can be made smaller by moving the closed-loop poles near the origin. Placing the poles at the origin will result in a deadbeat controller and it should be avoided at all circumstances.

2.5 ROBUST IMPLICIT STR

In the last few sections, we have seen the basics of Lyapunov stability techniques and passivity, and their applicability to the feedback controller design of nonlinear discrete-time systems. The suite of nonlinear design tools includes adaptive controllers. Adaptive controllers are designed when dynamic systems have certain unknown parameters. Adaptive controllers are typically designed using Lyapunov stability analysis and by using suitable parameter update algorithms. Parameter adaptation laws are nonlinear and therefore the overall closed-loop system becomes nonlinear. Parameter update schemes have to be carefully selected in order to ensure that the actual parameters converge to their true values while the controller is steering the system to perform certain regulation or tracking tasks.

Many industrial processes have unknown parameters and therefore adaptive control is an important area. Adaptive controllers in discrete-time are referred to as STRs. Research in adaptive control has resulted in several important developments in the last three decades. A number of books present the adaptive control techniques both in continuous- and discrete-time (Landau 1979; Goodwin and Sin 1984; Narendra and Annaswamy 1989; Sastry and Bodson 1989). The progress of adaptive control theory and the availability of microprocessors have led to a series of successful applications in the last two decades in the areas of robotics, aircraft control, process control, estimation, and the like. However, despite remarkable successes, discrete-time adaptive techniques developed in the first two decades can be applied only to systems operating under ideal conditions, which is clearly a limitation.

In the late 1980s, there was a surge in the development of robust adaptive control techniques with respect to noise, unmodeled dynamics, and disturbances (Ortega et al. 1985). Despite the success of robust adaptive control for discrete-time systems, several of these techniques are only applicable when the plant has a stable inverse and a fixed delay and is strictly positive real; these are stringent assumptions. In addition, successful applications have required a careful selection of the adaptation mechanisms and sampling frequency (Goodwin 1991; Landau 1993). Currently, research in the area of adaptive control is directed in the development of general-purpose robust adaptive controllers that can be applied to a wide range of systems including nonlinear systems operating in adverse conditions (Jagannathan and Lewis 1996). For a detailed survey on gain scheduling, model reference adaptive control, and STRs see Åström (1983, 1987) and Landau (1993).

Considerable research has been conducted in parameter estimation (Åström 1987), and explicit-based STR and implicit-based STR design for many industrial applications. Unfortunately, little literature is available about the use of implicit STR designs that yield guaranteed performance even for linear systems. Kanellakopoulos (1994) points out that very few results exist for discrete-time nonlinear systems, where sampling-related problems are not present and one has to impose linear growth conditions on the nonlinearities to provide global stability. Therefore much effort is being devoted to the analysis of STR in the presence of unmodeled dynamics and bounded disturbances (Landau 1993). Especially, in the presence of noise, high-frequency dynamics, and bounded disturbances, most of these parameter updates have to be modified to accommodate the variation in the system dynamics.

In continuous-time systems, the estimation and control are combined in the direct model reference adaptive systems (MRAS) and Lyapunov proofs are available to guarantee stability of the tracking error as well as boundedness of the parameter estimates. By contrast, in the discrete-time case the Lyapunov proofs are so intractable that simultaneous demonstration of stable tracking and bounded estimates is not available (Åström and Wittenmark 1989) for a long time. Instead, the certainty equivalence (CE) principle is invoked to decompose the problem into an estimation part and a controller part. Then, various techniques such as least-squares and averaging are employed to show the stability and bounded estimates.

Therefore the STR design (Ren and Kumar 1994) is usually carried out as a nonlinear stochastic problem rather than a deterministic approach. In fact, Kumar (1990) examined the stability, convergence, asymptotic optimality, and self-tuning properties of stochastic adaptive control schemes based on least-squares estimates of the unknown parameters using CE principle for linear systems. Later, Guo and Chen (1991) have shown for the first time the convergence, stability, and optimality for the original self-tuning regulator proposed

by Åström and Wittenmark in 1973 as a stochastic adaptive control problem using the CE control law.

To confront all these issues head on, in this section, a Lyapunov-based stability approach is formulated for an STR in order to control discrete-time nonlinear system. Specifically, an implicit design of STR attempted in Jagannathan and Lewis (1996) is taken and the stability of the closed-loop system is presented using the Lyapunov technique, since little about the application of STR in direct closed-loop application that yield guaranteed performance is discussed in the literature. By guaranteed we mean that both the tracking errors and the parameter estimates are bounded. This approach will indeed overcome the sector-bound restriction that is common in the discrete-time control literature. In addition, note that in the continuous-time case, the Lyapunov function is chosen so that its derivative is linear in the parameter error (provided that the system is linear in the parameters) and in the derivative of the parameter estimates (Kanellakopoulos 1994). This crucial property is not present in the difference of a discrete-time Lyapunov function which is a major problem. However, in this section, this problem is indeed overcome by appropriately combining the terms and completing the squares in the first difference of the Lyapunov function. For the first time in the literature, CE assumption was relaxed in the work of Jagannathan and Lewis (1996). Finally, this section will set the stage for the more advanced NN-based adaptive controllers that are covered in subsequent chapters.

The proposed adaptive scheme from Jagannathan and Lewis (1996) is composed of an implicit STR incorporated into a dynamical system, where the structure comes from tracking error/passivity notions. It is shown that the gradient-based tuning algorithm yields a passive STR. This, if coupled with the dissipativity of the dynamical system, guarantees the boundedness of all the signals in the closed-loop system under a persistency of excitation (PE) condition (Section 2.5.2). However, PE is difficult to guarantee in an adaptive system for robust performance. Unfortunately, if PE does not hold, the gradient-based tuning generally does not guarantee tracking and bounded parameters. Moreover, it is found here that the maximum permissible tuning rate for gradient-based algorithms decreases with an increase in the upper bound on the regression vector; this is a major drawback. A projection algorithm (Section 2.5.3) is shown to easily correct the problem. New modified update tuning algorithms introduced in Section 2.5.5 avoid the need for PE by making the STR robust, that is, state strict passive.

2.5.1 BACKGROUND

Let \mathbb{R} denote the real numbers, \mathbb{R}^n denote the real n -vectors, and $\mathbb{R}^{m \times n}$ the real $m \times n$ matrices. Let S be a compact simply connected subset of \mathbb{R}^n . With maps $f : S \rightarrow \mathbb{R}^k$, define $C^k(S)$ as the space such that f is continuous.

We denote by $|\cdot|$ any suitable vector norm. Given a matrix $A = [a_{ij}] \in \Re^{n \times m}$, the Frobenius norm is as defined in Section 2.2.1. The associated inner product is defined as $\langle A, B \rangle_F = \text{tr}(A^T B)$. The Frobenius norm, $\|A\|_F$, which is denoted by $\|\cdot\|$ throughout this section until unless specified explicitly, is nothing but the vector 2-norm over the space defined by stacking the matrix columns into a vector, so that it is compatible with the vector 2-norm, that is $\|Ax\| \leq \|A\| \|x\|$.

2.5.1.1 Adaptive Control Formulation

At sampling instant k , let the plant input be denoted by $u(k)$ and the output by $y(k)$. The general input–output representation of a simple adaptive control scheme conveniently expressed in matrix format for multi-input and multi-output (MIMO) system is

$$y(k+1) = \theta^T \phi(k) \quad (2.68)$$

with $\theta \in \Re^{n \times m}$, $y(k) \in \Re^{n \times 1}$, and $\phi(k) \in \Re^{m \times 1}$, being the regressor. The adaptive scheme can be further extended to nonlinear systems $f(x(k))$ that can be expressed as linear in the unknown parameters. Here the regression vector is a nonlinear function of past outputs and inputs.

A general nonlinear function $f(x) \in C^k(U)$ can be written with a linear in the unknown parameters assumption as

$$f(x(k)) = \theta^T \phi(x(k)) + \varepsilon(k) \quad (2.69)$$

with $\varepsilon(k)$ a parameter or functional reconstitution error vector that includes all the uncertainties during estimation. If there exists a fixed number N_2 denoting the number of past values of output and input, and constant parameters such that $\varepsilon = 0$ for all $x \in U$ then $f(x)$ is in the parameter or functional range of the adaptation scheme. In general, given a constant real number $\varepsilon \geq 0$, $f(x(k))$ is within an ε_N range of the adaptation scheme if there exists N_2 and constant parameters so that for all $x \in \Re^n$, (2.69) holds with $\|\varepsilon(k)\| \leq \varepsilon_N$. Note that the selection of N_2 , which is usually assumed in the adaptive literature as the delay bank, for a specified $U \in \Re^n$, and the functional reconstruction error-bound ε_N , are current topics of research. This formulation is more general than standard STR schemes, where it is assumed that the functional or parameter reconstruction error $\varepsilon(k)$ is equal to zero. The result is the applicability of this scheme to a wide class of systems, as well as guaranteed robustness properties.

Define the estimated output as

$$\hat{y}(k+1) = \hat{\theta}^T(k) \phi(k) \quad (2.70)$$

In the remainder of this chapter, parameter update laws are derived based on the Lyapunov technique, so that the closed-loop system is stable.

2.5.1.2 Stability of Dynamical Systems

In order to formulate the discrete-time controller, the following stability notations are needed. Consider the linear discrete time-varying system given by

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= C(k)x(k) \end{aligned} \quad (2.71)$$

where $A(k)$, $B(k)$, and $C(k)$ are appropriately dimensioned matrices.

Lemma 2.5.1: Define $\psi(k_1, k_0)$ as the state transition matrix corresponding to $A(k)$ for the system (2.71), that is, $\psi(k_1, k_0) = \prod_{k=k_0}^{k_1-1} A(k)$. Then if $\|\psi(k_1, k_0)\| < 1, \forall k_1, k_0 \geq 0$, the system (2.71) is exponentially stable.

Proof: See Ioannou and Kokotovic (1983).

Linear Systems: A plant for the MIMO case can be rewritten in the form of (2.71) as

$$y(k+1) = \theta^T \phi(k) + \beta_0 u(k) + d(k) \quad (2.72)$$

with $y(k) \in \Re^n$, $\theta \in \Re^{n \times (n+m-1)}$, $\phi(k) \in \Re^{(n+m-1) \times 1}$, $\beta_0 \in \Re^{n \times n}$, $u(k) \in \Re^n$, and $d(k) \in \Re^n$. Here, the disturbance vector $d(k)$ is bounded by the known bound. The regression vector comprises of both the past values of the output and the input. Then the following mild assumption is made, similar to many adaptive control techniques.

Assumption 2.5.1: The gain matrix β_0 is known beforehand.

Given a desired trajectory $y_d(k+1)$, define the output tracking error at time instant $k+1$ as

$$e(k+1) = y(k+1) - y_d(k+1) \quad (2.73)$$

Using (2.72) in (2.73), the error dynamics can be rewritten as

$$e(k+1) = \theta^T \phi(k) + \beta_0 u(k) + d(k) - y_d(k+1) \quad (2.74)$$

Select $u(k)$ in (2.74) as

$$u(k) = \beta_0^{-1}[-\hat{\theta}^T(k)\phi(k) + y_d(k+1) + k_v e(k)] \quad (2.75)$$

with k_v a closed-loop constant gain matrix. Then the error dynamics (2.74) can be represented as

$$e(k+1) = k_v e(k) + \hat{\theta}^T(k)\phi(k) + d(k) \quad (2.76)$$

where $\tilde{\theta} = \theta - \hat{\theta}$. This is an error system wherein the output tracking error is driven by the parameter estimation error. Note that in (2.75), the gain matrix is considered to be known. This assumption can be relaxed by estimating the gain matrix as well. However, one has to assure that the inverse of the gain matrix exists in all cases. In other words, one has to guarantee the boundedness of the matrix away from zero, and this topic is addressed using NN in Chapter 3.

Equation 2.76 can be further expressed for nonideal conditions as

$$e(k+1) = k_v e(k) + \tilde{\theta}^T(k)\phi(k) + \varepsilon(k) + d(k) \quad (2.77)$$

where $\varepsilon(k)$ is the parameter estimation error, whose bound $\|\varepsilon(k)\| \leq \varepsilon_N$ is known.

Dynamics of the nonlinear MIMO system: Consider a MIMO system given by

$$y(k+1) = f(y(k), \dots, y(k-n+1)) + \sum_{j=0}^{m-1} \beta_j u(k-j) + d(k) \quad (2.78)$$

where $y(k) \in \Re^n$, $f(\cdot) \in \Re^n$, and $\beta_j \in \Re^{n \times n}$. The disturbance is considered to be bounded, with a known upper bound. Note also that the nonlinear function is assumed to be expressed as linear in the unknown parameters.

Case I: $\beta_j, j = 1, \dots, m-1$ are known. Given a desired trajectory $y_d(k+1)$, define the output tracking error at the time instant $k+1$ as (2.73). Using (2.78) in (2.73) one obtains

$$e(k+1) = f(y(k), \dots, y(k-n+1)) + \beta_0 u(k) + \sum_{i=1}^{m-1} \beta_i u(k-i) - y_d(k+1) \quad (2.79)$$

Select the input $u(k)$ as

$$u(k) = \beta_0^{-1} \left[-\hat{f}(y(k), \dots, y(k-n+1)) - \sum_{i=1}^{m-1} \beta_i u(k-i) + y_d(k+1) + k_v e(k) \right] \quad (2.80)$$

And using (2.78), (2.79) can be expressed as

$$e(k+1) = k_v(k) + \tilde{f}(\cdot) + \varepsilon(k) + d(k) \quad (2.81)$$

which is exactly the form given by (2.77) using the linearity or the unknown parameters assumption for the function $f(\cdot)$. Equation 2.81 can then be expressed as (2.77), where the regression matrix in (2.81) is a function of only past values of the output, whereas in (2.77), it is a function of both past values of both input and output.

Case II: $\beta_j, j = 1, \dots, m-1$ are unknown. Given the desired trajectory, select the output $u(k)$ as

$$u(k) = \beta_0^{-1} \left[-\hat{f}(y(k), \dots, y(k-n+1)) - \sum_{i=1}^{m-1} \hat{\beta}_i u(k-i) + y_d(k+1) + k_v e(k) \right] \quad (2.82)$$

where $\hat{\beta}_j, j = 1, \dots, m-1$, are estimates of the unknown parameters β_j . Then (2.79) can be expressed as

$$e(k+1) = k_v(k) + \tilde{f}(\cdot) + \varepsilon(k) + d(k) + \sum_{j=0}^{m-1} \tilde{\beta}_j u(k-j) \quad (2.83)$$

where $\tilde{\beta}_j, j = 1, \dots, m-1$, are the errors in parameters. Then using the linearity-in-parameters assumption for the function $f(\cdot)$, (2.83) can be rewritten as

$$e(k+1) = k_v(k) + \sum_{i=0}^{n-1} \tilde{\alpha}_i y(k-i) + \sum_{j=0}^{m-1} \tilde{\beta}_j u(k-j) + \varepsilon(k) + d(k) \quad (2.84)$$

Equation 2.84 can be expressed in the form (2.77) by combining the second and third terms in (2.84), where $\tilde{\theta}^T(k) = [\tilde{\alpha}_i(k) \ \tilde{\beta}_j(k)]^T$ in (2.84) is given by

$$\tilde{\theta}(k) = \begin{bmatrix} \tilde{\alpha}_{0,0}(k) & \dots & \tilde{\alpha}_{0,n-1}(k) & \tilde{\beta}_{0,1}(k) & \dots & \tilde{\beta}_{0,m-1}(k) \\ \vdots & & \vdots & & & \\ \tilde{\alpha}_{n-1,0}(k) & \dots & \tilde{\alpha}_{n-1,n-1}(k) & \tilde{\beta}_{n-1,1}(k) & \dots & \tilde{\beta}_{n-1,m-1}(k) \end{bmatrix} \quad (2.85)$$

In the above two cases, the plant is represented in input–output form. However, it is possible that in many situations the plant may not be expressible in the above form, but it can be expressed in a specified structural form. In addition, when several systems are interconnected, hyperstability theory is essential to guarantee boundedness of outputs and states. In such a case, one needs to show the property of dissipativity of the plant as well as the passivity property of the adaptation mechanism in order to prove the bounded-input–bounded-output stability. It may or may not be possible to show that a particular nonlinear system is dissipative unless one is careful in representing the plant in a particular fashion. Similarly, not all parameter updates can be shown to have the property of passivity. To this end, for a class of nonlinear systems given in the next subsection, one needs to employ the filtered tracking error notion (Slotine and Li 1991), which is quite common in the robotics control literature to show the dissipativity of the original nonlinear system.

Dynamics of the mnth order MIMO discrete-time nonlinear system: Dynamics are given by

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= f(x(k)) + \beta_0 u(k) + d(k) \end{aligned} \quad (2.86)$$

where $x(k) = [x_1(k) \cdots x_n(k)]^T$ with $x_i(k) \in \Re^n$, $i = 1, \dots, n$, $\beta_0 \in \Re^{n \times n}$, $u(k) \in \Re^{n \times n}$, and $d(k) \in \Re^m$ denotes a disturbance vector acting on the system at the instant k , with $\|d(k)\| \leq d_M$ a known constant.

Given a desired trajectory $x_{nd}(k)$ and its delayed values, define the tracking error as

$$e_n(k) = x_n(k) - x_{nd}(k) \quad (2.87)$$

It is typical in robotics to define a so-called filtered tracking error as $r(k) \in \mathbb{R}^m$ and given by

$$r(k) = e_n(k) + \lambda_1 e_{n-1}(k) + \cdots + \lambda_{n-1} e_1(k) \quad (2.88)$$

where $e_{n-1}(k), \dots, e_1(k)$ are the delayed values of the error $e_n(k)$, and $\lambda_1, \dots, \lambda_{n-1}$ are constant matrices selected so that $|z^{n-1} + \lambda_1 z^{n-2} + \cdots + \lambda_{n-1}|$ is stable. Equation 2.88 can be further expressed as

$$r(k+1) = e_n(k+1) + \lambda_1 e_{n-1}(k-1) + \cdots + \lambda_{n-1} e_1(k+1) \quad (2.89)$$

Using (2.86) in (2.89), the dynamics of the m th order MIMO system can be written in terms of the tracking error as

$$\begin{aligned} r(k+1) &= f(x(k)) - x_{nd}(k+1) + \lambda_1 e_n(k) + \cdots + \lambda_{n-1} e_2(k) \\ &\quad + \beta_0 u(k) + d(k) \end{aligned} \quad (2.90)$$

Define the control input $u(k)$ in (2.90) as

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{f}(x(k)) + k_v r(k) - \lambda_1 e_n(k) - \cdots - \lambda_{n-1} e_2(k)] \quad (2.91)$$

with the closed-loop gain matrix k_v and $\hat{f}(x(k))$ an estimate of $f(x(k))$. Then the closed-loop error system becomes

$$r(k+1) = k_v r(k) + \tilde{f}(x(k)) + d(k), \quad (2.92)$$

where the functional estimation error is given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)) \quad (2.93)$$

This is an error system in which the filtered tracking error is driven by the functional estimation error. Using the linearity in the parameter assumption on $f(\cdot)$ and $\hat{f}(\cdot)$, (2.93) can be further expressed as

$$r(k+1) = k_v r(k) + \tilde{\theta}^T(k) \phi(k) + \varepsilon(k) + d(k) \quad (2.94)$$

In the remainder of this chapter, (2.77) and (2.94) are used to focus on selecting STR tuning algorithms that guarantee the stability of the output tracking error $r(k)$ in (2.94). Then, since (2.88), with the input considered as $r(k)$ and the output $e(k)$, describes a stable system and standard technique (Slotine and Li 1991) guarantee that $e(k)$ exhibits stable behavior.

2.5.2 STR DESIGN

In this section, stability analysis by Lyapunov's direct method is carried out for a family of parameter-tuning algorithms for implicit STR design developed based on the gradient rule. These tuning paradigms yield a passive STR, yet PE is generally needed for suitable performance. Unfortunately, PE cannot generally be tested for or guaranteed in the inputs, so that these gradient-based parameter-tuning algorithms are generally doomed to failure. Modified tuning paradigms are proposed to make the STR robust so that PE is not needed. Finally, for guaranteed stability, the gradient-based parameter-tuning algorithms must slow down with an increase in the upper bound on the regressor. By employing a projection algorithm, it is shown that the tuning rate can be made independent of the regressor.

Assume that there exist constant parameters for the STR. Then the nonlinear function in (2.86) can be written as

$$f(x(k)) = \theta^T \phi(k) + \varepsilon(k) \quad (2.95)$$

where $\|\varepsilon(k)\| < \varepsilon_N$, with the bounding constant ε_N known. This scenario allows one to select a simple STR structure and thereafter compensating for the increased magnitude of ε_N by using the gain term k_v , as will be seen.

2.5.2.1 Structure of the STR and Error System Dynamics

Define the approximate output as in (2.71) and functional estimate by

$$\hat{f}(x(k)) = \hat{\theta}^T(k) \phi(k) \quad (2.96)$$

with $\hat{\theta}(k)$ the current value of the weights. Take θ to be the matrix of constant parameters required in (2.96) and assume they are bounded by known values so that

$$\|\theta\| \leq \theta_{\max} \quad (2.97)$$

Then, the error in the parameters during estimation, also called the parameter estimation error, is given by

$$\tilde{\theta}(k) = \theta - \hat{\theta}(k) \quad (2.98)$$

Select the control input $u(k)$ for the system (2.72) to be (2.75), so that the closed-loop tracking error system is rewritten for convenience as

$$e(k+1) = k_v e(k) + \bar{e}_i(k) + \varepsilon(k) + d(k) \quad (2.99)$$

where $\bar{e}_i(k)$ is defined as the identification error and is given in (2.102). Similarly, select the control input $u(k)$ for the system (2.86) to be

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{\theta}^T(k)\phi(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) + k_v r(k)] \quad (2.100)$$

Then the closed-loop filtered error dynamics become

$$r(k+1) = k_v r(k) + \bar{e}_i(k) + \varepsilon(k) + d(k), \quad (2.101)$$

where the identification error denoted in (2.101) is given by

$$\bar{e}_i(k) = \hat{\theta}^T(k)\phi(k) \quad (2.102)$$

The next step is to determine the parameter update laws for the error systems derived in (2.99) and (2.101) so that the tracking performance of the closed-loop error dynamics is guaranteed.

2.5.2.2 STR Parameter Updates

A family of STR tuning paradigms, including the gradient rule that guarantee the stability of the closed-loop systems (2.99) and (2.101) are presented in this section. It is required to demonstrate that the tracking error $e(k)$ for (2.99) and $r(k)$ for (2.101) is suitably small and that the STR parameters $\hat{\theta}(k)$ remain bounded, for then the control $u(k)$ is bounded. In order to proceed further, the following definitions are needed.

Lemma 2.5.2: If $A(k) = I - \alpha\phi(k)\phi^T(k)$ in (2.99), where $0 < \alpha < 2$ and $\phi(k)$ is the regression matrix, then $\|\psi(k_1, k_0)\| < 1$ is guaranteed if there is an $L > 0$ such that $\sum_{k=k_0}^{k_1+L-1} \phi(k)\phi^T(k) > 0$ for all k . Then Lemma 2.5.1 guarantees the exponential stability of the system (2.99).

Proof: See Ioannou and Kokotovic (1983).

Definition 2.5.1: An input sequence $x(k)$ is said to be persistently exciting (Ioannou and Kokotovic 1983) if there are $\lambda > 0$ and an integer $k_1 \geq 1$ such that

$$\lambda_{\min} \left[\sum_{k=k_0}^{k_1+k-1} x(k)x^T(k) \right] > \lambda \quad \forall k_0 \geq 0 \quad (2.103)$$

where $\lambda_{\min}(P)$ represents the smallest eigenvalue of P . Note that PE is exactly the stability condition needed in Lemma 2.5.2.

In the following, it is first assumed that the STR reconstruction error-bound ε_N and the disturbance-bound d_M are nonzero. Theorem 2.5.1 gives two alternative parameter-tuning algorithms, one based on a modified functional or parameter estimation error and the other based on the tracking error, showing that both the tracking error and the error in the parameter estimates are bounded if a PE condition holds. Throughout this paper, for convenience, tracking error denotes both output and filtered-error tracking errors.

Theorem 2.5.1 (STR with PE Condition): Let the desired trajectories be $y_d(k + 1)$ for the case of (2.99), $x_{nd}(k + 1)$ for (2.101), and the initial conditions be bounded in a compact set U . Let the STR functional or parameter reconstruction error- and the disturbance-bounds ε_N and d_M , respectively, be known constants. Consider the parameter tuning provided by either

$$(a) \quad \theta(k + 1) = \hat{\theta}(k) + \alpha\phi(k)\bar{f}^T(k) \quad (2.104)$$

where $\bar{f}(k)$ is defined as the parameter augmented error for the error system (2.99) as

$$\bar{f}(k) = y_d(k + 1) - u(k) - \hat{\theta}(k)\phi(k) \quad (2.105)$$

and $\bar{f}(k)$ is defined as the functional augmented error for the error system (2.101) as

$$\bar{f}(k) = x_n(k + 1) - u(k) - \hat{f}(x(k)) \quad (2.106)$$

or

$$(b) \quad \hat{\theta}(k + 1) = \hat{\theta}(k) + \alpha\phi(k)e^T(k + 1) \quad (2.107)$$

for the error system (2.99)

$$\hat{\theta}(k + 1) = \hat{\theta}(k) + \alpha\phi(k)r^T(k + 1) \quad (2.108)$$

for the error system (2.101), where $\alpha > 0$ is a constant adaptation gain. Let the past inputs and outputs contained in the regression vector $\phi(k)$ be persistently

exciting and let the following conditions hold:

$$\begin{aligned}\alpha \|\phi(k)\|^2 &< 1 \\ k_{v\max} &< \frac{1}{\sqrt{\eta}},\end{aligned}\quad (2.109)$$

where η is given for algorithm (a) as

$$\eta = 1 + \frac{1}{1 - \alpha \|\phi(k)\|^2} \quad (2.110)$$

and for algorithm (b) as

$$\eta = \frac{1}{1 - \alpha \|\phi(k)\|^2} \quad (2.111)$$

Then the output tracking error $e(k)$ in (2.99) and filtered tracking error $r(k)$ in (2.101) and the errors in parameter estimates $\tilde{\theta}(k)$ are UUB, and the practical bounds given explicitly for both $e(k)$ and $r(k)$, denoted here by b_t are obtained from Appendix 2.A as

$$b_t = \frac{(\eta - 1)}{1 - \eta k_{v\max}^2} (\varepsilon_N + d_M) \left(\eta k_{v\max} + \sqrt{\frac{1 - k_{v\max}^2}{\eta - 1}} \right) \quad (2.112)$$

for algorithm (a) and

$$b_t = \frac{1}{1 - \eta k_{v\max}^2} (\varepsilon_N + d_M) (\eta k_{v\max} + \sqrt{\eta}) \quad (2.113)$$

for algorithm (b). Moreover, the bounds for $\tilde{\theta}(k)$ can be obtained from (2.A.9) and (2.A.13), respectively, for algorithms (a) and (b).

Proof: See Appendix 2.A.

Outline of proof: In the proof, it is first demonstrated by using a Lyapunov function, the tracking error dynamics (2.99) and (2.101) and the parameter updates (2.104) and (2.106) for algorithm (a), and (2.104) and (2.106) for algorithm (b), that the output tracking error $e(k)$, and the filtered tracking error function $r(k)$ are bounded. In addition, it is necessary to show that the parameter estimates are also bounded. In order to prove the boundedness of the error in

the parameter estimates, the PE condition, the bound on the tracking error, and the dynamics in the error in parameter estimates are considered. Using these, in fact it is shown that the parameters of the STR are bounded.

Note from (2.112) and (2.113) that the tracking error increases with the STR reconstruction error bound ε_N and the disturbance-bound d_M , yet small tracking errors may be achieved by selecting small gains k_v . In other words, placing the closed-loop error poles closer to the origin inside the unit circle forces smaller tracking errors. Selecting $k_{v\max} = 0$ results in a deadbeat controller, but this should be avoided since it is not robust.

Remarks:

1. It is important to note that in this theorem there is no CE assumption for the controller, in contrast to standard work in discrete-time adaptive control (Astrom and Wittenmark 1989). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption, and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in the Appendix (Section 2.A) of this chapter is of the form

$$J = r^T(k)r(k) + \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k)\tilde{\theta}(k)]$$

which weights the tracking errors, $r(k)$ and the parameter estimation errors for the controller, $\tilde{\theta}(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

2. The parameter updating rules (2.104), (2.107), and (2.108) are non-standard schemes that were derived from Lyapunov analysis and do not include an extra term which is normally used to provide robustness due to the coupling in the proof between the tracking errors and parameter estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required if the PE condition is applied.
3. Condition (2.109) can be checked easily. The maximum singular value of the controller gain $k_{v\max}$ and the parameter adaptation gain has to satisfy (2.109) in order for the closed-loop system to be stable.

This is a unique relationship between the controller gain and the parameter adaptation matrix. This condition states that for faster tuning of parameters, the closed-loop poles should be far inside the unit disc. On the other hand, such constraints do not exist for controller design parameters and parameter adaptation gains in continuous-time. This explains why the design parameters for the adaptive controllers in continuous-time are selected somewhat arbitrarily.

2.5.3 PROJECTION ALGORITHM

The adaptation gain $\alpha > 0$ is a constant parameter in the update laws presented in (2.104), (2.107), and (2.108). These update laws correspond to the gradient rule (Åström and Wittenmark 1989). The theorem reveals that update tuning mechanisms employing the gradient rule have a major drawback. In fact, using (2.109), an upper bound on the adaptation gain can be obtained as

$$\alpha < \frac{1}{\|\phi(k)\|^2} \quad (2.114)$$

Since $\phi(k) \in \Re^{N_2}$, with N_2 the number of past values of inputs and outputs in the regressor, it is evident that the upper bound on the adaptation gain depends upon the bound on the regression vector. Specifically, if there are N_2 past values of input and output in the regression vector and if the maximum value of the regression vector is \sqrt{M} then the bound on the adaptation gain in order to assure stability of the closed-loop system is given by

$$0 < \alpha < \frac{1}{M} \quad (2.115)$$

In other words, the upper bound on the adaptation gain for the case of gradient-based tuning decreases with an increase in the upper bound on the regression vector, so that adaptation must slow down owing to the addition of more past values of input and output in the regression vector for guaranteed performance.

This major drawback can be easily overcome by modifying the update law in order to obtain a projection algorithm (Goodwin and Sin 1984; Åström and Wittenmark 1989). Replace the constant adaptation gain by

$$\alpha = \frac{\xi}{\xi + \|\phi(k)\|^2} \quad (2.116)$$

where $0 < \xi < 1$ and $\xi > 0$ are constants. Note that ξ is now the new adaptation gain, and it is always true that

$$\frac{\xi}{\xi + \|\phi(k)\|^2} \|\phi(k)\|^2 < 1 \quad (2.117)$$

hence guaranteeing (2.109) for every N_2 .

Finally, it is very interesting to note that for identification purposes alone the upper bound on the adaptation gain in (2.109) is well known to be less than two (Goodwin and Sin 1984; Åström and Wittenmark 1989), whereas, for the purpose of control, the upper bound on the adaptation gain is further reduced to one owing to the constraints imposed by the tracking error system.

2.5.4 IDEAL CASE: NO DISTURBANCES AND NO STR RECONSTRUCTION ERRORS

The next result discusses the behavior of the closed-loop system in the idealized case of no STR parameter and functional reconstruction errors with no unmodeled disturbances in the dynamics of the system. In this case, PE is not needed.

Theorem 2.5.2 (Ideal Case with No STR Reconstruction Errors and Bounded Disturbances): Let the desired trajectory $y_d(k+1)$ for (2.80) and $x_{nd}(k+1)$ for (2.91) be bounded and the STR parameter and functional reconstruction error-bound ε_N and the disturbance-bound d_M , be equal to zero. Let the parameter tuning be provided by either

$$(a) \quad \hat{\theta}(k+1) = \hat{\theta}(k) + \alpha\phi(k)\bar{f}^T(k) \quad (2.118)$$

or

$$(b) \quad \hat{\theta}(k+1) = \hat{\theta}(k) + \alpha\phi(k)e^T(k+1) \quad (2.119)$$

for the error system (2.99) and

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \alpha\phi(k)r^T(k+1) \quad (2.120)$$

for the error system (2.101), where $\alpha > 0$ is a constant adaptation.

Then the tracking errors $e(k)$ and $r(k)$, respectively, for (2.99) and (2.101) asymptotically approach zero and the parameter estimates are bounded provided the condition (2.109) holds and (2.110) of Theorem 2.5.1 holds, with η as given for Algorithm (a) and Algorithm (b) in (2.110) and (2.111), respectively.

Proof: Since the functional reconstruction error and the disturbances are all zero; these new assumptions yield the error systems

$$e(k+1) = k_v e(k) + \bar{e}_i(k) \quad (2.121)$$

for the error system (2.99), whereas

$$r(k+1) = k_v r(k) + \bar{e}_i(k) \quad (2.122)$$

for the error system (2.101). Although the proof is shown for the error system (2.122), the proof for (2.121) is exactly the same as for (2.122).

Algorithm (a): Selecting the Lyapunov function candidate (2.A.1) with the new assumptions and the update tuning mechanism (2.118) results in the following first difference:

$$\begin{aligned}
\Delta J &= -r^T(k)(I - k_v^T k_v)r(k) + \bar{e}_i^T(k)\bar{e}_i(k) \\
&\quad + 2[k_v r(k)]^T \bar{e}_i(k) - [2 - \alpha \phi^T(k)\phi(k)]\bar{e}_i^T(k)\bar{e}_i(k) \\
&\leq -r^T(k) \left[I - \left(1 + \frac{1}{1 - \alpha \|\phi(k)\|^2} \right) k_v^T k_v \right] r(k) - [1 - \alpha \|\phi(k)\|^2] \\
&\quad \times [\bar{e}_i(k) - k_v r(k)]^T [\bar{e}_i(k) - k_v r(k)] \\
&\leq -(1 - \eta k_{v\max}^2) \|r(k)\|^2 - [1 - \alpha \|\phi(k)\|^2] \|\bar{e}_i(k) - k_v r(k)\|^2 \quad (2.123)
\end{aligned}$$

where η is given by (2.110). Since $J > 0$ and $\Delta J \leq 0$, this shows stability in the sense of Lyapunov, provided the condition (2.109) holds, so that $r(k)$ and $\bar{\theta}(k)$ (and hence $\hat{\theta}(k)$) are bounded if $r(k_0)$ and $\bar{\theta}(k_0)$ are bounded in the compact set U . In addition, on summing both sides of (2.123), one notes that as $e(k+1) = k_v e(k) + \bar{e}_i(k)$, $k \rightarrow \infty$, the tracking error $\|r(k)\| \rightarrow 0$ (Lin and Narendra 1980).

Algorithm (b): For the case of the weight-tuning mechanism given in (2.120), select the Lyapunov function candidate as (2.A.1), and use the new assumptions as well as the update law in (2.A.2) to obtain

$$\begin{aligned}
\Delta J &= -r^T(k)\{I - [1 + \alpha \phi^T(k)\phi(k)]k_v^T k_v\}r(k) + 2\alpha \phi^T(k)\phi(k)[k_v r(k)]^T \bar{e}_i(k) \\
&\quad - [1 - \alpha \phi^T(k)\phi(k)]\bar{e}_i^T(k)\bar{e}_i(k) \\
&= -r^T(k) \left\{ I - [1 - \alpha \phi^T(k)\phi(k)] + \frac{[\alpha \phi^T(k)\phi(k)]^2}{[1 - \alpha \phi^T(k)\phi(k)]} k_v^T k_v \right\} r(k) \\
&\quad - [1 - \alpha \phi^T(k)\phi(k)] \left[\bar{e}_i(k) - \frac{[\alpha \phi^T(k)\phi(k)]^2}{[1 - \alpha \phi^T(k)\phi(k)]} k_v r(k) \right]^T \\
&\quad \times \left[\bar{e}_i(k) - \frac{[\alpha \phi^T(k)\phi(k)]^2}{[1 - \alpha \phi^T(k)\phi(k)]} k_v r(k) \right] \\
&\leq -(1 - \eta k_{v\max}^2) \|r(k)\|^2 - [1 - \alpha \|\phi(k)\|^2] \\
&\quad \times \left\| \bar{e}_i(k) - \frac{\alpha \|\phi(k)\|^2}{1 - \alpha \|\phi(k)\|^2} k_v r(k) \right\|^2 \quad (2.124)
\end{aligned}$$

where η is given by (2.111). Since $J > 0$ and $\Delta J \leq 0$, this shows stability in the sense of Lyapunov, provided the condition (2.109) holds, so that $r(k)$ and $\tilde{\theta}(k)$ (and hence $\hat{\theta}(k)$) are bounded if $r(k_0)$ and $\tilde{\theta}(k_0)$ are bounded in the compact set U . In addition, on summing both sides of (2.124), one notes that as $r(k+1) = k_v r(k) + \bar{e}_i(k)$, $k \rightarrow \infty$, the tracking error $\|r(k)\| \rightarrow 0$ (Lin and Narendra 1980).

Note that now for guaranteed closed-loop stability, it is not necessary that the past inputs and outputs contained in the regression vector be PE. Equation 2.118 and Equation 2.120 are nothing but the gradient-based parameter-tuning algorithms. Theorem 2.5.2 indicates that gradient-based parameter updates suffice when the parameter or functional reconstruction error $\varepsilon(k)$ and disturbances $d(k)$ are zero. However, Theorem 2.5.1 reveals the failure of standard gradient-based parameter tuning in the presence of STR reconstruction errors and bounded disturbances. Therefore, gradient-based tuning updates used in an STR that cannot exactly reconstruct certain unknown parameters because of the presence of nonlinearities $f(\cdot)$ or uncertainties in the estimation process with bounded unmodeled disturbances, cannot be guaranteed to yield bounded estimates. Then the PE condition is required to guarantee boundedness of the parameter estimates. However, it is very difficult to guarantee or verify the PE of the regression vector $\phi(k)$. This possible unboundedness of the parameter estimates when PE fails to hold is known as parameter drift (Åström and Wittenmark 1989; Narendra and Annaswamy 1989; Slotine and Li 1991). In the next section, improved parameter-tuning paradigms for the STR are presented so that PE is not required.

2.5.5 PARAMETER-TUNING MODIFICATION FOR RELAXATION OF PE CONDITION

Approaches such as σ -modification (Ioannou and Kokotovic 1983) or ε -modification (Narendra and Annaswamy 1987) are available for the robust adaptive control of continuous systems, for which the PE condition is not needed. The property of robustness in the update laws is needed or arises from the persistent excitation due to the disturbances or changes in the reference signal.

Modifications have been suggested in the update schemes for the discrete-time adaptive control; these schemes include parameter bounding, deadzone, and leakage techniques (Cook 1994; Åström and Wittenmark 1997). In the case of parameter bounding, the parameters are forced to remain in a fixed set. This will, however, require that prior knowledge be available about the bounds on the unknown parameters. Another well-known approach to avoiding parameter drift is to switch off the parameter estimation when the tracking error

is large. However, the size of the deadzone is dependent upon the bound on the disturbance and a priori knowledge of the reference signal and the like.

The other way to avoid the parameter drift problem is to add an additional term by shifting the equilibria and it is sometimes called leakage in discrete-time adaptive control, whereas it is called σ -modification in continuous-time. However, the bound on the unknown parameters and the a priori knowledge of a certain resonant term is needed. In addition, this approach will force the origin to no longer be an equilibrium point. Finally, all these schemes in discrete-time (Goodwin and Sin 1984; Astrom and Wittenmark 1997) are guaranteed to perform well by empirical studies only with no convergence or stability proofs, whereas their continuous-time counterparts are guaranteed to perform successfully both analytically and by simulation studies. Therefore, in this paper, an approach similar to ε -modification is derived for discrete-time systems and the boundedness of both tracking error and errors in parameter estimates are guaranteed through Lyapunov analysis. In fact, the following theorem shows two tuning algorithms that overcome the need for PE.

Theorem 2.5.3 (STR with No PE Condition Requirement): Assume the hypotheses presented in Theorem 2.5.1, and consider the modified tuning algorithms provided by either

$$(a) \quad \hat{\theta}(k+1) = \theta(k) + \alpha\phi(k)\bar{f}^T(k) - \Gamma\|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}(k) \quad (2.125)$$

or

$$\hat{\theta}(k+1) = \theta(k) + \alpha\phi(k)e^T(k+1) - \Gamma\|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}(k) \quad (2.126)$$

for the error system (2.99) and

$$\hat{\theta}(k+1) = \theta(k) + \alpha\phi(k)r^T(k+1) - \Gamma\|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}(k) \quad (2.127)$$

for the error system (2.101), with $\Gamma > 0$ a design parameter. Then the output and filtered tracking errors $e(k)$ and $r(k)$, respectively, for the error systems (2.99) and (2.101) and the STR parameter estimates $\hat{\theta}(k)$ are UUB, and the practical bounds for both $e(k)$ and $r(k)$ and $\tilde{\theta}(k)$, denoted here by b_t and b_θ , respectively, are given by

$$\begin{aligned} b_t = & \frac{1}{(1 - \eta k_{v \max}^2)} + \left[\kappa k_{v \max} (\varepsilon_N + d_M) \right. \\ & \left. + \sqrt{\kappa^2 k_{v \max}^2 (\varepsilon_N + d_M)^2 + \rho(1 - \eta k_{v \max}^2)} \right] \end{aligned} \quad (2.128)$$

$$b_\theta = \frac{\Gamma(1 - \Gamma)\theta_{\max} + \sqrt{\Gamma^2(1 - \Gamma)^2\theta_{\max}^2 + \Gamma(2 - \Gamma)\rho}}{\Gamma(2 - \Gamma)} \quad (2.129)$$

for algorithm (a), and

$$b_t = \frac{1}{1 - \bar{\sigma} k_{v \max}^2} \left[\gamma k_{v \max} + \sqrt{\rho_1(1 - \bar{\sigma} k_{v \max}^2)} \right] \quad (2.130)$$

$$b_\theta = \frac{\Gamma(1 - \Gamma)\theta_{\max} + \sqrt{\Gamma^2(1 - \Gamma)^2\theta_{\max}^2 + \Gamma(2 - \Gamma)\bar{\theta}}}{\Gamma(2 - \Gamma)} \quad (2.131)$$

for algorithm (b), provided the following conditions hold:

$$\alpha \|\phi(k)\|^2 < 1 \quad (2.132)$$

$$0 < \Gamma < 1 \quad (2.133)$$

$$k_{v \max} < \frac{1}{\sqrt{\eta}} \quad (2.134)$$

for algorithm (a),

$$k_{v \max} < \frac{1}{\sqrt{\bar{\sigma}}} \quad (2.135)$$

for algorithm (b), where η is given in (2.109) for algorithm (a) and $\bar{\sigma}$ for algorithm (b) is given by

$$\bar{\sigma} = \eta + \frac{1}{1 - \alpha \phi_{\max}^2} [\Gamma^2(1 - \alpha \phi_{\max}^2)^2 + 2\alpha \Gamma \phi_{\max}^2(1 - \alpha \phi_{\max}^2)] \quad (2.136)$$

where $\|\phi(k)\| \leq \phi_{\max}$ with η in (2.136) given by (2.111). Note that κ and ρ in (2.129) and (2.130) with ρ_1 and θ in (2.131) and (2.132) are presented in (2.A.16), (2.A.20) and (2.A.26), (2.A.27), respectively.

Remark: No PE condition is needed with the modified tuning algorithms.

Proof: See Appendix 2.A.

Outline of the proof. It is first demonstrated by using a Lyapunov function and the parameter updates (2.125) and (2.127) along with the tracking error dynamics (2.99) and (2.101), that the tracking error is bounded. Then it can be shown by simple manipulation that the errors in the parameter estimates are also bounded without a PE condition. The parameter updates in Theorem 2.5.3 include an additional term similar to the ε -modification approach for the continuous-time systems. It is found that an exact replica of the ε -modification term used for the

continuous-time case does not yield a bound on the parameters for the discrete-time case. However, using (2.125) and (2.127), it can be shown that both the tracking error and STR parameter errors are bounded.

Note that the STR reconstruction error ε_N and the bounded disturbances d_M determine the bounds on $\|e(k)\|$ or $\|r(k)\|$ and $\|\tilde{\theta}(k)\|$. Small tracking error bounds may be achieved by placing the closed-loop poles inside the unit circle and near the origin through the selection of the largest gain eigenvalue k_{vmax} . On the other hand, the STR parameter error estimates are fundamentally bounded by θ_{\max} , the known bound on the ideal parameter θ . The parameter Γ offers a design trade-off between the relative eventual magnitudes of $\|e(k)\|$ or $\|r(k)\|$ and $\|\tilde{\theta}(k)\|$; a smaller Γ yields a smaller $\|e(k)\|$ or $\|r(k)\|$ and a larger $\|\tilde{\theta}(k)\|$ and vice versa.

Remarks:

1. It is important to note that in this theorem also there is no PE condition and CE assumption for the adaptive controller, in contrast to standard work in discrete-time adaptive control (Astrom and Wittenmark 1989). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption, and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in Appendix 2.A of this chapter is of the form

$$J = r^T(k)r(k) + \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k)\tilde{\theta}(k)]$$

which weights the tracking errors, $r(k)$, the parameter estimation errors for the controller, $\tilde{\theta}(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

2. The parameter updating rules (2.125) through (2.127) are nonstandard schemes that were derived from Lyapunov analysis and do include an extra term referred to as discrete-time ε -modification (Jagannathan and Lewis 1996), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and parameter estimation error terms. The Lyapunov proof

demonstrates that the additional term in the weight tuning is not required if the PE condition is applied.

3. Conditions (2.132) through (2.135) can be checked easily similar to Theorem 2.5.2. The maximum singular value of the controller gain $k_v \max$ and the parameter adaptation gain has to satisfy (2.134) in order for the closed-loop system to be stable. This is a unique relationship between the controller gain and the parameter adaptation matrix. This condition states that for faster tuning of parameters, the closed-loop poles should be far inside the unit disc. On the other hand, such constraints do not exist for controller designs in continuous-time.

2.5.6 PASSIVITY PROPERTIES OF THE STR

The closed-loop error system (2.99) and (2.101) can be redrawn now in the standard feedback configuration as opposed to the STR in the previous section. Passivity is essential in a closed-loop system, since it guarantees the boundedness of signals, and hence suitable performance, even in the presence of additional unforeseen bounded disturbances (i.e., STR robustness). Therefore in this section, passivity properties of the STR, and hence of the closed-loop system, are guaranteed through the parameter-tuning algorithms.

In general, the closed-loop tracking error system can also be expressed as

$$e(k + 1) = k_v e(k) + \xi_0(k) \quad (2.137)$$

for the case of (2.99), whereas

$$r(k + 1) = k_v r(k) + \xi_0(k) \quad (2.138)$$

for the case of (2.101), with

$$\xi_0(k) = \tilde{\theta}(k)\phi(k) + \varepsilon(k) + d(k) \quad (2.139)$$

The following dissipative result holds for this system.

Theorem 2.5.4 (Passivity of Tracking Error System): The closed-loop tracking error systems (2.137), (2.138) are state strict passive from $\xi_0(k)$ to $k_v e(k)$ for the system (2.137) and to $k_v r(k)$ for the system (2.138) provided that

$$\lambda_{\max}(k_v^2) < 1 \quad (2.140)$$

The following proof is given for the error system (2.138); that for (2.137) is similar and hence omitted.

Proof: Select a Lyapunov function candidate

$$J = r^T(k)r(k) \quad (2.141)$$

The first difference is given by

$$\Delta J = r^T(k+1)r(k+1) - r^T(k)r(k) \quad (2.142)$$

Substituting (2.138) into (2.142) yields

$$\Delta J = -r^T(k)(I - k_v^T k_v)r(k) + 2r^T(k)k_v \xi_0 + \xi_0^T(k)\xi_0(k) \quad (2.143)$$

Note that (2.143) is in power form defined in (2.33) with $g(k)$ a quadratic function of the state $r(k)$. Hence (2.138) is a state strict passive system.

Remark: Note that for the error system (2.101), rewritten as (2.138), passivity is shown from the input to the output $k_v r(k)$. Unless the nonlinear system (2.86) is expressed in the filtered tracking error fashion, it is not possible to show the passivity properties of the closed-loop system. Even when the filtered tracking error systems (2.99) and (2.101) are SSP, the overall system is not passive unless the parameter update laws guarantee the passivity of the lower block. It is usually difficult to demonstrate that the errors in parameter updates are passive. However, in the following theorem, it is in fact shown that the gradient-based tuning algorithms (2.104) and (2.107) yield a passive STR.

Theorem 2.5.5 (Passivity of the Parameter Tuning Requiring PE Condition): The weight tuning algorithms (2.104) to (2.109) make the map from $\varepsilon(k) + d(k)$ for algorithm (a) and $k_v e(k) + \varepsilon(k) + d(k)$ or $k_v r(k) + \varepsilon(k) + d(k)$ for algorithm (b) to $-\tilde{\theta}^T(k)\phi(k)$ a passive map.

Proof:

Algorithm (a). Define a Lyapunov function candidate

$$J = \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k)\tilde{\theta}(k)] \quad (2.144)$$

whose first difference is given by

$$\Delta J = \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k+1)\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\tilde{\theta}(k)] \quad (2.145)$$

Substituting the weight-update law (2.104) into (2.145) yields

$$\begin{aligned}
 \Delta J &= \frac{1}{\alpha} \text{tr}\{\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T[I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) - \tilde{\theta}^T(k)\tilde{\theta}(k) \\
 &\quad - 2\alpha\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T\phi(k)[\varepsilon(k) + d(k)]^T \\
 &\quad + \alpha^2[\varepsilon(k) + d(k)]\phi^T(k)\phi(k) \times [\varepsilon(k) + d(k)]^T\} \\
 &= \frac{1}{\alpha} \text{tr}\{-\alpha[2 - \alpha\phi^T(k)\phi(k)] \times [-\tilde{\theta}^T(k)\phi(k)][-\tilde{\theta}^T(k)\phi(k)]^T \\
 &\quad + 2\alpha[1 - \alpha\phi(k)\phi^T(k)][-\tilde{W}^T(k)\phi(x(k))] \times [\varepsilon(k) + d(k)]^T \\
 &\quad + \alpha^2[\phi^T(x(k)\phi(x(k)))] [\varepsilon(k) + d(k)][\varepsilon(k) + d(k)]^T\} \tag{2.146}
 \end{aligned}$$

Note that (2.146) is in power form (2.33) as long as the condition (2.109) holds. This in turn guarantees the passivity of the weight-tuning mechanism (2.104).

Algorithm (b). Select the Lyapunov function (2.144). Use (2.108) in (2.145) to obtain

$$\begin{aligned}
 \Delta J &= \{-[2 - \alpha\phi^T(k)\phi(k)][-\tilde{\theta}^T(k)\phi(k)] \times [-\tilde{\theta}^T(k)\phi(k)]^T \\
 &\quad + 2[1 - \alpha\phi(k)\phi^T(k)] \times [-\tilde{\theta}^T(k)\phi(k)]^T[k_v r(k) + \varepsilon(k) + d(k)] \\
 &\quad + \alpha\phi^T(k)\phi(k)[k_v r(k) + \varepsilon(k) + d(k)]^T \times [k_v r(k) + \varepsilon(k) + d(k)]\} \tag{2.147}
 \end{aligned}$$

which is in power form (2.33) for discrete-time systems as long as the condition (2.109) holds.

Here, only the passivity of the error system (2.138) is given. The passivity of the parameter error system (2.137) can be shown similarly, and hence it is omitted.

Thus, the parameter error blocks (2.101) through (2.108) are passive and the closed-loop tracking error system (2.99) and (2.101) are dissipative; this guarantees the dissipativity of the closed-loop system (Landau 1979). By employing the passivity theorem (Slotine and Li 1991), one can conclude that the input/output signals of each block are bounded as long as the external inputs are bounded. However, though dissipative, the closed-loop system is not state strict passive, so this does not yield boundedness of the internal states of the lower block (i.e., $\tilde{\theta}(k)$) unless PE is not needed with the modified update algorithms (2.125) to (2.127) of Theorem 2.5.6.

Theorem 2.5.6 (SSP of Modified Parameter-Tuning Schemes): The weight tuning mechanisms (2.126) through (2.128) make the map from $\varepsilon(k) + d(k)$ for algorithm (a) and $k_v e(k) + \varepsilon(k) + d(k)$ or $k_v r(k) + \varepsilon(k) + d(k)$ for algorithm (b) to $-\tilde{\theta}^T(k)\phi(k)$ a state strict passive map.

Proof:

Algorithm (a). The revised dynamics relative to $\tilde{\theta}(k)$ using (2.125) are given by

$$\begin{aligned}\tilde{\theta}(k+1) = & [I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) - \alpha\phi(x(k))[\varepsilon(k) + d(k)]^T \\ & + \Gamma \|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}(k)\end{aligned}\quad (2.148)$$

Select the Lyapunov function candidate (2.144), and use (2.148) in (2.145) to obtain

$$\begin{aligned}\Delta J = & -[2 - \alpha\phi^T(k)\phi(k)][-\tilde{\theta}^T(k)\phi(k)] \times [-\tilde{\theta}^T(k)\phi(k)]^T \\ & + 2[1 - \alpha\phi(k)\phi^T(k) - \Gamma \|I - \alpha\phi(k)\phi^T(k)\|] \\ & \times [-\tilde{\theta}^T(k)\phi(k)]^T[\varepsilon(k) + d(k)] + \alpha\phi^T(k)\phi(k)[\varepsilon(k) + d(k)]^T \\ & \times [\varepsilon(k) + d(k)] - \chi + \gamma_0^2\end{aligned}\quad (2.149)$$

where χ is a constant given in Jagannathan and Lewis (1996)

$$\gamma_0^2 = \frac{\Gamma^2}{\alpha}(1 - \alpha\phi_{\max}^2)^2\theta_{\max}^2 \quad (2.150)$$

Note (2.149) is in power form (2.33) for discrete-time systems. Similarly, it can be shown using (2.127) that the parameter update law is SSP.

It should be noted that SSP of both the system dynamics and the parameter-tuning block does guarantee SSP of the closed-loop system, so that the norms of the internal states are bounded in terms of the power delivered to each block. Then boundedness of input/output signals assures state boundedness even without PE.

An STR is defined to be passive if, in the error formulation, it guarantees the passivity of the lower subsystem. Then, an extra PE condition is needed to guarantee boundedness of the parameter estimate. Note that (i) dissipativity of the error system is needed to guarantee boundedness of the parameter estimates. An STR is defined to be robust if, in the error formulation, it guarantees the SSP of the lower subsystem. Then no extra PE condition is required for boundedness of the parameter estimate. Note that (i) dissipativity of the error

system is needed in addition to tracking stability to guarantee bounded parameters and (ii) the STR passivity properties are dependent on the parameter-tuning algorithm used.

2.5.7 CONCLUSIONS

A family of implicit STR has been developed for the control of a class of nonlinear dynamical systems. The STR has a structure derived from passivity/tracking error notions, and offers guaranteed performance. Gradient-rule-based tuning has been shown to yield a passive STR, so that it performs well under ideal conditions of no parameter or functional reconstruction errors, unmodeled bounded disturbances and no uncertainties in the estimation process. In addition, it has been found that the adaptation gain in the standard gradient-rule-based parameter update algorithms must decrease with increasing number of past values of inputs and outputs in the regression vector, so that adaptation must slow down.

In order to overcome the above deficiencies, a family of improved parameter-tuning algorithms has been derived. The improved tuning paradigms consist of a gradient-based update term plus a correction term similar to the ε -modification approach in the case of continuous-time adaptive control. The improved tuning algorithms make the STR state strict passive, so that bounded parameter estimates are guaranteed in practical nonideal situations. Furthermore, the adaptation gain is modified to obtain a projection algorithm so that the adaptation rate is independent of the number of past values of inputs and outputs in the regression vector. Simulation results in discrete-time have been presented in order to illustrate the performance of the controller even in the presence of bounded disturbances. Finally, this section has introduced a comprehensive theory in the development of identification, prediction, and adaptive control schemes for discrete-time systems based on Lyapunov analysis.

REFERENCES

- Åström, K.J., Theory and applications of adaptive control — a survey, *Automatica*, 19, 471–486, 1983.
- Åström, K.J., Adaptive feedback control, *Proc. IEEE*, 75, 185–218, 1987.
- Åström, K.J. and Wittenmark, B., *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.
- Cook, P.A., Application of model reference adaptive control to a benchmark problem, *Automatica*, 30, 585–588, 1994.
- Goodwin, C.G., Can we identify adaptive control, *Proc. European Contr. Conf.*, 1714–1725, 1991.

- Goodwin, C.G. and Sin, K.S., *Adaptive Filtering, Prediction, and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Guo, L. and Chen, H., The Åström–Wittenmark self-tuning regulator revisited and ELS-based adaptive trackers, *IEEE Trans. Autom. Contr.*, AC-36, 802–812, 1991.
- Ioannou, P. and Kokotovic, P., *Adaptive Systems with Reduced Models*. Springer-Verlag, New York, 1983.
- Jagannathan, S. and Lewis, F.L., Robust implicit self tuning regulator: convergence and stability, *Automatica*, 32, 1629–1644, 1996.
- Kalkkuhl, J.C. and Hunt, K.J., Discrete-time neural model structures for continuous-time nonlinear systems, *Neural Adaptive Control Technology*, Zbikowski, R. and Hunt, K.J., Eds., Chapter 1, World Scientific, Singapore, 1996.
- Kanellakopoulos, I., A discrete-time adaptive nonlinear system, *IEEE Trans. Autom. Contr.*, AC-39, 2362–2364, 1994.
- Kumar, P.R., Convergence of adaptive control schemes using least-squares parameter estimates, *IEEE Trans. Autom. Contr.*, AC-35, 416–424, 1990.
- Landau, Y.D., *Adaptive Control: The Model Reference Approach*, Marcel Dekker, Inc., Basel, 1979.
- Landau, I.D., Evolution of adaptive control, *ASME J. Dynan. Syst. Meas. Contr.*, 115, 381–391, 1993.
- Lewis, F.L. and Syrmos, V.L., *Optimal Control*, 2nd ed., John Wiley & Sons, New York, 1995.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.
- Lewis, F.L., Jagannathan, S., and Yesiderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, London, 1999.
- Lin, Y. and Narendra, K.S., A new error model for adaptive systems, *IEEE Trans. Autom. Contr.*, AC-25, 1980.
- Luenberger, D.G., *Introduction to Dynamic Systems*, John Wiley & Sons, New York, 1979.
- Narendra, K.S. and Annaswamy, A.M., A new adaptive law for robust adaptation without persistent excitation, *IEEE Trans. Autom. Contr.*, AC-32, 134–145, 1987.
- Narendra, K.S. and Annaswamy, A.M., *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Ortega, R., Praly, L., and Landau, I.D., Robustness of discrete-time direct adaptive controllers, *IEEE Trans. Autom. Contr.*, AC-30, 1179–1187, 1985.
- Ren, W. and Kumar, P.R., Stochastic adaptive prediction and model reference control, *IEEE Trans. Autom. Contr.*, AC-39, 2047–2060, 1994.
- Sastry, S. and Bodson, M., *Adaptive Control*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Slotine, J.-J.E. and Li, W., *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- Von Bertalanffy, L., *General System Theory*, Braziller, New York, 1998.
- Whitehead, A.N., *Science and the Modern World*, Lowell Lectures (1925), Macmillan, New York, 1953.

PROBLEMS

SECTION 2.1

2.1-1: *Simulation of compound interest system.* Simulate the system of Example 2.1.3 and plot the state with respect to time.

2.1-2: *Genetics.* Many congenital diseases can be explained as a result of both genes at a single location being the same recessive gene (Luenberger 1979). Under some assumptions, the frequency of the recessive gene at generation k is given by the recursion

$$x(k+1) = \frac{x(k)}{1+x(k)}$$

Simulate in MATLAB using $x(0) = 75$. Observe that $x(k)$ converges to zero, but very slowly. This explains why deadly genetic diseases can remain active for a very long time. Simulate the system starting for a small negative value of $x(0)$ and observe that it tends away from zero.

2.1-3: *Discrete-time system.* Simulate the system

$$\begin{aligned} x_1(k+1) &= \frac{x_2(k)}{1+x_2(k)} \\ x_2(k+1) &= \frac{x_1(k)}{1+x_2(k)} \end{aligned}$$

using MATLAB. Plot the phase-plane plot.

SECTION 2.3

2.3-1: *Passivity definition.* Show (2.33) is same as (2.34). To show this, sum both sides of the former equation and use the boundedness assumption on $L(x(k))$.

SECTION 2.4

2.4-1: *Lyapunov stability analysis.* Using the Lyapunov stability analysis examine stability for the following systems. Plot time histories to substantiate your claims.

$$(a) \quad x_1(k+1) = x_1(k)\sqrt{(x_1^2(k) + x_2^2(k))}$$

$$x_2(k+1) = x_2(k)\sqrt{(x_1^2(k) + x_2^2(k))}$$

$$(b) \quad x(k+1) = -x^2(k) - 10x(k) \sin x(k)$$

design an adaptive controller by selecting a suitable parameter update when the parameters are considered unknown except $b_0 = 5$. Select the following $a_0 = 0.98$, $a_1 = 1.27$, and $a_2 = 1.97$ and a sinusoid as the desired trajectory for the simulations.

APPENDIX 2.A

Proof of Theorem 2.5.2

Algorithm (a): Define the Lyapunov function candidate

$$J = r^T(k)r(k) + \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k)\tilde{\theta}(k)] \quad (2.A.1)$$

The first difference is

$$\begin{aligned} \Delta J &= \Delta J_1 + \Delta J_2 = r^T(k+1)r(k+1) - r^T(k)r(k) \\ &\quad + \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k+1)\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\tilde{\theta}(k)] \end{aligned} \quad (2.A.2)$$

Considering the first term ΔJ_1 from (2.A.2) and substituting (2.101) results in

$$\begin{aligned} \Delta J_1 &= r^T(k+1)r(k+1) - r^T(k)r(k) \\ &= [k_v r(k) + \bar{e}_i(k) + \varepsilon(k) + d(k)]^T \\ &\quad \times [k_v r(k) + \bar{e}_i(k) + \varepsilon(k) + d(k)] - r^T(k)r(k) \\ &= r^T(k)k_v^T k_v r(k) + 2[k_v r(k)]^T \bar{e}_i(k) + \bar{e}_i^T(k) \bar{e}_i(k) \\ &\quad + [\varepsilon(k) + d(k)]^T [\varepsilon(k) + d(k)] + 2[k_v r(k)]^T [\varepsilon(k) + d(k)] \\ &\quad + 2[\varepsilon(k) + d(k)]^T e_i(k) - r^T(k)r(k) \end{aligned} \quad (2.A.3)$$

Now, taking the second term in (2.A.2), rewriting the functional augmented error in (2.104) as

$$\bar{f} = \tilde{\theta}^T(k)\phi(k) + \varepsilon(k) + d(k) \quad (2.A.4)$$

And using this equation with (2.104) and (2.104) yields

$$\begin{aligned} \Delta J_2 &= \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k+1)\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\tilde{\theta}(k)] \\ &= \frac{1}{\alpha} \text{tr}\{\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T \times [I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) - \tilde{\theta}^T(k)\tilde{\theta}(k) \\ &\quad - 2\alpha\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T \phi(k)[\varepsilon(k) + d(k)]^T \\ &\quad + \alpha^2[\varepsilon(k) + d(k)]\phi^T(k)\phi(k)[\varepsilon(k) + d(k)]^T\} \end{aligned} \quad (2.A.5)$$

Given a vector $x \in \Re^n$, $\text{tr}(xx^T) = x^T x$, using this in the above equation, one may obtain

$$\begin{aligned} \Delta J_2 = & -[2 - \alpha\phi^T(k)\phi(k)]\bar{e}_i^T(k)\bar{e}_i(k) - 2[1 - \alpha\phi(k)\phi^T(k)]\bar{e}_i^T(k) \\ & \times [\varepsilon(k) + d(k)]\alpha\phi(k)\phi^T(k)[\varepsilon(k) + d(k)]^T[\varepsilon(k) + d(k)] \end{aligned} \quad (2.A.6)$$

Substituting (2.A.3) and (2.A.6) into (2.A.2), collecting terms together, and completing the square for $\bar{e}_i(k)$ yields

$$\begin{aligned} \Delta J = & -[1 - \alpha\phi^T(k)\phi(k)] \left(\bar{e}_i - \frac{k_v r(k) + \alpha\phi^T(k)\phi(k)[\varepsilon(k) + d(k)]}{1 - \alpha\phi^T(k)\phi(k)} \right)^T \\ & \times \left(\bar{e}_i(k) - \frac{k_v r(k) + \alpha\phi^T(k)\phi(k)[\varepsilon(k) + d(k)]}{1 - \alpha\phi^T(k)\phi(k)} \right) \\ & - r^T(k)(I - \eta k_v^T k_v)r(k) + 2\eta[k_v r(k)]^T[\varepsilon(k) + d(k)] \\ & + \frac{1}{1 - \alpha\phi^T(k)\phi(k)} \times [\varepsilon(k) + d(k)]^T[\varepsilon(k) + d(k)] \\ \leq & -(1 - \eta k_{v \max}^2) \left(\|r(k)\|^2 - 2\frac{(\eta - 1)k_{v \max}}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M) \|r(k)\| \right. \\ & \left. - \frac{(\eta - 1)}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M)^2 \right) - [1 - \alpha\|\phi(k)\|^2]\bar{e}_i(k) \\ & - \left\| \frac{k_v r(k) + \alpha\|\phi(k)\|^2[\varepsilon(k) + d(k)]}{1 - \alpha\|\phi(k)\|^2} \right\|^2 \end{aligned} \quad (2.A.7)$$

where η is given in (2.110), with $k_{v \max}$ the maximum singular value of k_v . Since $\varepsilon_N + d_M$ is constant, $\Delta J \leq 0$ as long as

$$\|r(k)\| > \frac{\eta - 1}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M) \left(k_{v \max} + \sqrt{\frac{1 - k_{v \max}^2}{\eta - 1}} \right) \quad (2.A.8)$$

$|\sum_{k=k_0}^{\infty} \Delta J(k)| = |J(\infty) - J(0)| < \infty$, since $\Delta J \leq 0$ in a compact set U , as long as (2.109), (2.110), and (2.A.8) hold. According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the tracking error $r(k)$ is bounded for all $k \geq 0$, and it remains to show that the weight estimates $\hat{\theta}(k)$, or equivalently $\tilde{\theta}(k)$, are bounded.

The dynamics relative to the errors in weight estimates are given by

$$\tilde{\theta}(k+1) = [I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) - \alpha\phi(k)[\varepsilon(k) + d(k)]^T \quad (2.A.9)$$

where the functional reconstruction error $\varepsilon(k)$ and the disturbance $d(k)$ are considered to be bounded. Using the PE condition (2.103) and Lemma 2.5.1, the boundedness of $\tilde{\theta}(k)$ in (2.A.9) and hence $\tilde{\theta}(k)$ are assured.

Algorithm (b). Define a Lyapunov function candidate as in (2.A.1) whose first difference is given by (2.A.2). The first term in (2.A.2) can be obtained from (2.101) and is given in (2.A.3). On the other hand, (2.108) is used to compute the second term in (2.A.2), which is given by

$$\begin{aligned}\Delta J_2 &= \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k+1)\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\tilde{\theta}(k)] \\ &= -2[k_v r(k)]^T \bar{e}_i(k) - 2\bar{e}_i^T(k)\bar{e}_i(k) + \alpha\phi(k)\phi^T(k)[k_v r(k) + \bar{e}_i(k)]^T \\ &\quad \times [k_v r(k) + \bar{e}_i(k)] - 2[1 - \alpha\phi(k)\phi^T(k)][\bar{e}_i(k)]^T [\varepsilon(k) + d(k)] \\ &\quad + 2\alpha\phi(k)\phi^T(k)[k_v r(k)]^T [\varepsilon(k) + d(k)] + \alpha\phi(k)\phi^T(k) \\ &\quad \times [\varepsilon(k) + d(k)]^T [\varepsilon(k) + d(k)]\end{aligned}\tag{2.A.10}$$

Substituting (2.A.3) and (2.A.10) into (2.A.2) and collecting terms together, one obtains

$$\begin{aligned}\Delta J &\leq -r^T(k)(I - k_v^T k_v)r(k) + 2[1 + \alpha \|\phi(k)\|^2][k_v r(k)]^T [\varepsilon(k) + d(k)] \\ &\quad - [1 - \alpha \|\phi(k)\|^2] \left(\bar{e}_i^T(k)\bar{e}_i(k) - 2\frac{\alpha \|\phi(k)\|^2}{1 - \alpha \|\phi(k)\|^2}[k_v r(k)]^T \bar{e}_i(k) \right) \\ &\quad + 2\alpha \|\phi(k)\|^2 [k_v r(k)]^T [k_v r(k)] + [1 + \alpha \|\phi(k)\|^2] \\ &\quad \times [\varepsilon(k) + d(k)]^T [\varepsilon(k) + d(k)] \\ &\leq -(1 - \eta k_{v \max}^2)^2 \left(\|r(k)\|^2 - 2\frac{\eta k_{v \max}}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M)\|r(k)\| \right. \\ &\quad \left. - \frac{\eta}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M)^2 \right) - [1 - \alpha \|\phi(k)\|^2] \\ &\quad \times \left\| \bar{e}_i(k) - \frac{\alpha \|\phi(k)\|^2}{1 - \alpha \|\phi(k)\|^2}[k_v r(k) + \varepsilon(k) + d(k)] \right\|^2\end{aligned}\tag{2.A.11}$$

with η given by (2.111). $\Delta J \leq 0$ as long as (2.109) holds and

$$\|r(k)\| > \frac{1}{1 - \eta k_{v \max}^2}(\varepsilon_N + d_M)(\eta k_{v \max} + \sqrt{\eta})\tag{2.A.12}$$

This demonstrates that ΔJ is negative outside a compact set U . According to a standard Lyapunov extension theorem (Lewis et al. 1993), the tracking

error $r(k)$ is bounded for all $k \geq 0$, and it remains to show that the parameter estimates $\hat{\theta}(k)$, or equivalently $\tilde{\theta}(k)$, are bounded.

The dynamics relative to errors in weight estimates are given by

$$\tilde{\theta}(k+1) = [I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) - \alpha\phi(k)[k_v r(k) + \varepsilon(k) + d(k)]^T \quad (2.A.13)$$

where the filtered tracking error $r(k)$ in (2.A.12), the functional reconstruction error $\varepsilon(k)$, and the disturbance $d(k)$ are considered to be bounded. Using the PE condition (2.103) and Lemma 2.5.1, the boundedness of $\hat{\theta}(k)$ in (2.A.13) and hence, $\tilde{\theta}(k)$ are assured.

Proof of Theorem 2.5.3

Algorithm (a). Select the Lyapunov function candidate as in (2.A.1) whose first difference is given by (2.A.2). The first term in (2.A.2) can be obtained from (2.101) and is given in (2.A.3). The second term in (2.A.2) is obtained as

$$\Delta J_2 = \frac{1}{\alpha} \text{tr}[\tilde{\theta}^T(k+1)\tilde{\theta}(k+1) - \tilde{\theta}^T(k)\tilde{\theta}(k)] \quad (2.A.14)$$

Using (2.126) in the above equation, one obtains

$$\begin{aligned} \Delta J_2 &= \frac{1}{\alpha} \text{tr}\{\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T[I - \alpha\phi(k)\phi^T(k)]\tilde{\theta}(k) + \alpha^2\phi(k)\phi^T(k) \\ &\quad \times [\varepsilon(k) + d(k)][\varepsilon(k) + d(k)]^T - 2\alpha\Gamma[\varepsilon(k) + d(k)]\phi^T(k) \\ &\quad \times \|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}^T(k) + \Gamma^2\|I - \alpha\phi(k)\phi^T(k)\|^2\hat{\theta}^T(k)\hat{\theta}(k) \\ &\quad + 2\Gamma\tilde{\theta}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T\|I - \alpha\phi(k)\phi^T(k)\|\hat{\theta}(k) - \hat{\theta}^T(k)\tilde{\theta}(k)\} \end{aligned} \quad (2.A.15)$$

Combining the above equation and (2.A.3) to obtain (2.A.2), rewriting with adding and subtracting

$$\alpha^{-1} \text{tr}[\|I - \alpha\phi(k)\phi^T(k)\|^2\tilde{\theta}^T(k)\tilde{\theta}(k)] \quad (2.A.16)$$

and completing the squares for $\bar{e}_i(k)$, one obtains

$$\begin{aligned} \Delta J &\leq -r^T(k) \left[I - k_v^T k_v \left(1 + \frac{1}{1 - \alpha\phi^T(k)\phi(k)} \right) \right] r(k) - [1 - \alpha\phi^T(k)\phi(k)] \\ &\quad \times \left\| \bar{e}_i(k) - \frac{1}{1 - \alpha\phi^T(k)\phi(k)} k_v r(k) \right. \\ &\quad \left. + [\alpha\phi^T(k)\phi(k) + \Gamma\|I - \alpha\phi(k)\phi^T(k)\|][\varepsilon(k) + d(k)] \right\|^2 \\ &\quad + [1 + \alpha\phi^T(k)\phi(k)][\varepsilon(k) + d(k)]^T[\varepsilon(k) + d(k)] \end{aligned}$$

$$\begin{aligned}
& + 2 \left\{ 1 + \frac{1}{1 - \alpha \phi^T(k) \phi(k)} [\Gamma \|I - \alpha \phi(k) \phi^T(k)\| + \alpha \phi^T(k) \phi(k)] \right\} \\
& \times [k_v r(k)]^T [\varepsilon(k) + d(k)] + \frac{1}{1 - \alpha \phi^T(k) \phi(k)} \\
& \times [\Gamma \|I - \alpha \phi(k) \phi^T(k)\| + \alpha \phi(k) \phi^T(k)]^2 [\varepsilon(k) + d(k)]^T \\
& \times [\varepsilon(k) + d(k)] + 2\Gamma \|I - \alpha \phi(k) \phi^T(k)\| \|\phi(k)\| \theta_{\max} (\varepsilon_N + d_M) \\
& - \frac{1}{\alpha} \|I - \alpha \phi(k) \phi^T(k)\|^2 [\Gamma(2 - \Gamma) \|\tilde{\theta}(k)\|^2 \\
& - 2\Gamma(1 - \Gamma) \theta_{\max} \|\tilde{\theta}(k)\| - \Gamma^2 \theta_{\max}^2] \tag{2.A.17}
\end{aligned}$$

Completing the squares for $\|\tilde{\theta}(k)\|$ using (2.A.16) to get

$$\begin{aligned}
\Delta J & \leq -(1 - \eta k_{v \max}^2) \left(\|r(k)\|^2 - \frac{2\kappa k_{v \max}}{1 - \eta k_{v \max}^2} (\varepsilon_N + d_M) - \frac{\rho}{1 - \eta k_{v \max}^2} \right) \\
& - [1 - \alpha \phi^T(k) \phi(k)] \left\| \bar{e}_i(k) - \frac{1}{1 - \alpha \phi^T(k) \phi(k)} k_v r(k) \right. \\
& \left. + [\alpha \phi^T(k) \phi(k) + \Gamma \|I - \alpha \phi(k) \phi^T(k)\|] [\varepsilon(k) + d(k)] \right\|^2 \\
& - \frac{1}{\alpha} \|I - \alpha \phi(k) \phi^T(k)\|^2 \Gamma(2 - \Gamma) \left(\|\tilde{\theta}(k)\| - \frac{1 - \Gamma}{2 - \Gamma} \theta_{\max} \right)^2 \tag{2.A.18}
\end{aligned}$$

where

$$\kappa = 1 + \frac{1}{1 - \alpha \phi_{\max}^2} [\Gamma(1 - \alpha \phi_{\max}^2) + \alpha \phi_{\max}^2] \tag{2.A.19}$$

$$\begin{aligned}
\rho & = \left(1 + \alpha \phi_{\max}^2 + \frac{1}{1 - \alpha \phi_{\max}^2} [\Gamma(1 - \alpha \phi_{\max}^2) + \alpha \phi_{\max}^2] \right)^2 (\varepsilon_N + d_M)^2 \\
& + 2\Gamma(1 - \alpha \phi_{\max}^2) \phi_{\max} \theta_{\max} (\varepsilon_N + d_M) + \frac{1}{\alpha} \frac{\Gamma}{2 - \Gamma} (1 - \alpha \phi_{\max}^2)^2 \theta_{\max}^2 \tag{2.A.20}
\end{aligned}$$

Then $\Delta J \leq 0$ as long as (2.132) through (2.134) hold and the quadratic term for $r(k)$ in (2.A.18) is positive, which is guaranteed when

$$\begin{aligned}
\|r(k)\| & > \frac{1}{1 - \eta k_{v \max}^2} \left[\kappa k_{v \max} (\varepsilon_N + d_M) \right. \\
& \left. + \sqrt{\kappa^2 k_{v \max}^2 (\varepsilon_N + d_M)^2 + \rho (1 - \eta k_{v \max}^2)} \right] \tag{2.A.21}
\end{aligned}$$

Similarly, completing the squares for $\|r(k)\|$ using (2.A.16) yields

$$\begin{aligned} \Delta J \leq & -(1 - \eta k_{v\max}^2) \left(\|r(k)\|^2 - \frac{\kappa k_{v\max}}{1 - \eta k_{v\max}^2} (\varepsilon_N + d_M) \right)^2 \\ & - [1 - \alpha \phi^T(x(k)) \phi(x(k))] \left\| \bar{e}_i(k) - \frac{1}{1 - \alpha \phi^T(x(k)) \phi(x(k))} k_v r(k) \right. \\ & \left. + [\alpha \phi^T(k) \phi(k) + \Gamma[\|I - \alpha \phi(k) \phi^T(k)\|][\varepsilon(k) + d(k)]] \right\|^2 \\ & - \frac{1}{\alpha} \|I - \alpha \phi(k) \phi^T(k)\|^2 [\Gamma(2 - \Gamma) \|\tilde{\theta}(k)\|^2 \\ & - 2\Gamma(1 - \Gamma) \theta_{\max} \|\tilde{\theta}(k)\| - \rho] \end{aligned} \quad (2.A.22)$$

where κ is given by (2.A.19) and

$$\begin{aligned} \rho = & \left[\frac{k_{v\max}^2 \kappa^2}{1 - \eta k_{v\max}^2} (\varepsilon_N + d_M)^2 + 2\Gamma(1 - \alpha \phi_{\max}^2) \phi_{\max} \theta_{\max} (\varepsilon_N + d_M) \right. \\ & + \left(1 + \alpha \phi_{\max}^2 + \frac{1}{1 - \alpha \phi_{\max}^2} [\Gamma(1 - \alpha \phi_{\max}^2) + \alpha \phi_{\max}^2] \right)^2 \\ & \times (\varepsilon_N + d_M)^2 \left. \right] \frac{\alpha}{(1 - \alpha \phi_{\max}^2)^2} + \Gamma^2 \theta_{\max}^2 \end{aligned} \quad (2.A.23)$$

Then $\Delta J \leq 0$ as long as (2.132) through (2.134) hold and the quadratic term for $\tilde{\theta}(k)$ in (2.A.23) is positive, which is guaranteed when

$$\|\tilde{\theta}(k)\| > \frac{\Gamma(1 - \Gamma) \theta_{\max} + \sqrt{\Gamma^2(1 - \Gamma)^2 \theta_{\max}^2 + \Gamma(2 - \Gamma)\rho}}{\Gamma(2 - \Gamma)} \quad (2.A.24)$$

From (2.A.21) or (2.A.24), ΔJ is negative outside a compact set U . According to a standard Lyapunov extension theorem (Lewis et al. 1993), the tracking error $r(k)$ is bounded for all $k \geq 0$, and it remains to show that the parameter estimates $\hat{\theta}(k)$, or equivalently $\bar{\theta}(k)$, are UUB.

Algorithm (b). Select the Lyapunov function candidate as in (2.A.1) whose first difference is given by (2.A.2). The first term ΔJ_1 is presented in (2.A.3). Use the tuning mechanism (2.125) and combining (2.A.3) and (2.101) and proceeding similarly to algorithm (a), we obtain

$$\begin{aligned} \Delta J \leq & -r^T(k)(I - k_v^T k_v)r(k) + 2[k_v r(k)]^T \bar{e}_i(k) + 2[k_v r(k)]^T [\varepsilon(k) + d(k)] \\ & + \bar{e}_i^T(k) \bar{e}_i(k) + 2[\varepsilon(k) + d(k)]^T \bar{e}_i(k) + [\varepsilon(k) + d(k)]^T [\varepsilon(k) + d(k)] \\ & - 2[1 - \alpha \phi(k) \phi^T(k)][k_v r(k)]^T \bar{e}_i(k) - [2 - \alpha \phi(k) \phi^T(k)] \bar{e}_i^T(k) \bar{e}_i(k) \end{aligned}$$

$$\begin{aligned}
& -2[1 - \alpha\phi(k)\phi^T(k)]\bar{e}_i^T(k)[\varepsilon(k) + d(k)] + \alpha\phi(k)\phi T(k)\{r^T(k)k_v^T k_v r(k) \\
& + 2[k_v r(k)]^T[\varepsilon(k) + d(k)] + [\varepsilon(k) + d(k)]^T[\varepsilon(k) + d(k)]\} \\
& - \frac{1}{\alpha}\|I - \alpha\phi(k)\phi^T(k)\|^2[\Gamma(2 - \Gamma)\|\tilde{W}(k)\|^2 2\Gamma(1 - \Gamma)\|\tilde{\theta}(k)\| \\
& \times \theta_{\max} - \Gamma^2\theta_{\max}^2] + 2\Gamma\|I - \alpha\phi(k)\phi^T(k)\|[k_v r(k)]^T\bar{e}_i(k) \\
& + 2\Gamma\|I - \alpha\phi(k)\phi^T(k)\|\bar{e}_i^T(k)[\varepsilon(k) + d(k)] \\
& + 2\Gamma k_{v\max}\|I - \alpha\phi(k)\phi^T(k)\|\|\phi(k)\|\theta_{\max}\|r(k)\| \\
& + 2\Gamma\|I - \alpha\phi(k)\phi^T(k)\|(\varepsilon_N + d_M)\|\phi(k)\|\theta_{\max} \\
& \leq -(1 - \bar{\sigma})k_{v\max}^2\|r(k)\|^2 - [1 - \alpha\phi(k)\phi^T(k)] \\
& \times \left\| \bar{e}_i(k) - \frac{1}{1 - \alpha\phi^T(k)\phi(k)}[\alpha\phi^T(k)\phi(k) + 2\Gamma\|I - \alpha\phi(k)\phi^T(k)\|] \right. \\
& \times [k_v r(k) + \varepsilon(k) + d(k)]^2 2\gamma k_{v\max}\|r(k)\| + \rho \\
& - \frac{1}{\alpha}\|I - \alpha\phi(k)\phi^T(k)\|^2 \\
& \times [\Gamma(2 - \Gamma)\|\tilde{\theta}(k)\|^2 - 2\Gamma(1 - \Gamma)\|\tilde{\theta}(k)\|\theta_{\max} - \Gamma^2\theta_{\max}^2] \quad (2.A.25)
\end{aligned}$$

where

$$\gamma = \eta(\varepsilon_N + d_M) + \Gamma(1 - \alpha\phi_{\max}^2)\phi_{\max}\theta_{\max} \quad (2.A.26)$$

$$\rho = \eta(\varepsilon_N + d_M)^2 + 2\Gamma(1 - \alpha\phi_{\max}^2)\phi_{\max}\theta_{\max}(\varepsilon_N + d_M) \quad (2.A.27)$$

Following steps similar to Algorithm (a), one can show boundedness of tracking error and parameter updates.

3 Neural Network Control of Nonlinear Systems and Feedback Linearization

In Chapter 2, standard adaptive controller development in discrete-time, normally referred as self tuning regulator (STR), was covered in detail. In contrast with the available controllers, the STR discussed in the previous chapter guarantees analytically the performance of the controller without persistency of excitation (PE) condition and certainty equivalence (CE) principle. The suite of nonlinear design tools includes adaptive controllers. However, most commercially available systems use proportional, integral, and derivative (PID) control algorithms. PID control allows accuracy acceptable for many applications at a set of via points specified by a human user but it does not allow accurate dynamic trajectory following between via points. As performance requirements on speed and accuracy of motion increase in today's micro- and nanoscale manufacturing environments, PID controllers lag further behind in providing adequate system performance. Since most commercial controllers do not use any sort of adaptation or learning capabilities, control accuracy is lost when certain nonlinearities change. In this chapter we show how to use biologically inspired control techniques to remedy these problems while further relaxing linear in the unknown parameter (LIP) assumption, which is required in STRs.

Adaptive controllers in general are designed when the dynamic systems have certain unknown parameters. However, an adaptive controller requires that the system under consideration can be expressed as an LIP, which is very difficult for a complex nonlinear industrial process (or system) to satisfy. This is an assumption that restricts the sorts of systems amenable to control. Actuator nonlinearities, for instance friction, do not satisfy the LIP assumption. Moreover, this LIP assumption requires one to determine the regression matrix for the system; this can involve tedious

computations and a new regression matrix must be computed for each system under consideration. Hyperstability and certain model-reference adaptive control (MRAC) techniques (Landau 1979) do not require LIP though they have not been applied in a rigorous manner to control complex industrial systems.

In Chapter 1 we saw that neural networks (NN) possess some very important properties, including a universal approximation property (Cybenko 1989) where, for every smooth function, $f(x)$, there exists an NN such that

$$f(x) = W^T \phi(V^T x) + \varepsilon(k) \quad (3.1)$$

for some weights W and V . This approximation holds for all x in a compact set S , and the functional estimation error $\varepsilon(x)$ is bounded so that

$$\|\varepsilon\| \leq \varepsilon_N \quad (3.2)$$

where ε_N is a known bound dependent on S . The approximating weights may be unknown, but the NN approximation property guarantees that they exist. In contrast with the adaptive control LIP requirement, which is an assumption that restricts the sorts of systems one can deal with, the result (3.1) is a property that holds for all smooth functions $f(x)$. An n -layer NN required for approximation is shown in Figure 1.24.

Advantage of NN control over standard adaptive control. The contrast between NN function approximation property (3.1) and LIP assumption of standard adaptive control (2.69) should be clearly understood. Both are linear in the tunable parameters, but the former is linear in the tunable NN weights, whereas the latter is linear in the unknown system parameters. The NN approximation property holds for all functions $f(x(k))$ in $C^m(S)$ whereas the LIP assumption holds for a specific function $f(x(k))$. In the NN approximation property the same basis set $\phi(x(k))$ suffices for all $f(x(k))$ in $C^m(S)$, while in the LIP assumption the regression matrix depends upon $f(x(k))$ and must be recomputed for each system under consideration. Therefore, the one-layer NN controller is significantly more powerful than adaptive standard LIP-based adaptive controllers; it provides a universal controller for a class of nonlinear systems. It is important to note the convergence of the error in both standard adaptive and NN controllers depend upon the initial conditions.

In this chapter, we propose to use a filtered-error-based approach in discrete-time, employing an NN to approximate unknown nonlinear functions in the complex industrial process dynamics, thereby overcoming some limitations of adaptive control (Jagannathan and Lewis 1996a, 1996b). The main results of this chapter are the controllers presented in Section 3.1 through Section 3.5. Instead of requiring knowledge of the system structure, as in adaptive control

(Jagannathan and Lewis 1996c), NN controllers use certain structural properties of the system, including passivity, to guarantee system performance. The NN will be designed to adapt its weights online to learn unknown dynamics. The study will be for a class of nonlinear systems, which includes rigid-link robot arms and other class of nonlinear systems. Initially, we assume that the states of the system are available through measurement. If only some states are measurable corresponding to the case of output feedback, then an additional dynamical or recurrent NN is required to estimate the unmeasured states (He and Jagannathan 2004).

Overcoming requirements for linearity in the tunable parameters has been a major obstacle to continued development of adaptive control techniques. In this chapter, we overcome this problem, providing tuning rules for a set of NN weights, some of which appear in a nonlinear fashion. In fact, the two-layer NN required in (3.1) is nonlinear in the first-layer weights.

Though one-layer NN can approximate a nonlinear function, one has to select suitable basis functions in order to achieve the desired approximation. The basis selection can be relaxed by using a two-layer NN, which is nonlinear in the first-layer weights V as given in Chapter 1. This nonlinear dependence of the two-layer NN creates some difficulties in designing an NN controller that adapts its weights online. Therefore, in Section 3.1.2 we first design a controller based on a simplified one-layer NN for a class of nonlinear system. In this section, the two-layer NN controller is also covered. The NN controllers have important passivity properties that make them robust to disturbances and unmodeled dynamics; these are detailed in Section 3.1.3.

In open-loop NN applications such as system identification, classification, and prediction, a bound on the NN weights alone implies the overall stability of the system, since the open-loop system is assumed stable. Therefore, gradient-based weight tuning (e.g., backpropagation) yielding nonincreasing weight energy functions are applied to these types of systems. On the contrary, in closed-loop feedback control applications, boundedness of the weights alone demonstrates very little. Therefore, standard open-loop weight-tuning schemes do not suffice in closed-loop control systems. There, both the tracking error and the NN weights must be guaranteed to be bounded while ensuring that the internal states remain bounded.

As mentioned in the previous chapter, in continuous-time systems, the estimation and control are combined in the design of direct adaptive control (Slotine and Li 1989) and in NN control (Yesilderek and Lewis 1994; Lewis et al. 1999), and Lyapunov proofs are available to guarantee stability of the tracking error as well as boundedness of the parameter estimates. By contrast, in the discrete-time case the Lyapunov proofs are so intractable that simultaneous demonstration of stable tracking and bounded estimates is not available (Jagannathan 1994). Therefore, in the next section are given the main results of

this chapter that are taken from (Jagannathan and Lewis 1996a, 1996b), where a one-layer NN controller is utilized first to adaptively control a class of nonlinear system. This work set the stage for more results in the area of discrete-time adaptive and NN control.

A family of novel learning schemes from Jagannathan (1994) is presented here that do not require preliminary off-line training. The traditional problems with discrete-time adaptive control are overcome by using a single Lyapunov function containing both the parameter identification errors and the control errors. This guarantees at once both stable identification and stable tracking. However, it leads to complex proofs where it is necessary to complete the square with respect to several different variables. The use of a single Lyapunov function for tracking and estimation avoids the need for the CE assumption. Along the way various other standard assumptions in discrete-time adaptive control are also overcome, including PE, linearity in the parameters (LIP) and the need for tedious computation of a regression matrix. Then, these results are extended to a more general class of affine nonlinear discrete-time systems.

3.1 NN CONTROL WITH DISCRETE-TIME TUNING

A foundation for NN in control has been provided in seminal results by Narendra and Parthasarathy (1990), Werbos (1974, 1989), and in the *Handbook of Intelligent Control* (White and Sofge 1992), *Neural Networks for Control* (Miller et al. 1991) and more recently in Lewis et al. (1999). Papers employing NN for control are too numerous to mention, but in the early years most of the works omit stability proofs and rely on ad hoc design and simulation studies. Several researchers have studied NN control and managed to prove stability (Polycarpou and Ioannou 1991; Sanner and Slotine 1991; Chen and Khalil 1992, 1994; Sadegh 1993; Rovithakis and Christodoulou 1994). Many of the results are developed for continuous-time systems except in Chen and Khalil (1994) where the results are provided for discrete-time systems.

To confront all these issues head on, in this section, a Lyapunov-based stability approach is formulated for an NN in order to control discrete-time nonlinear systems. Specifically, direct adaptive NN control is attempted and the stability of the closed-loop system is presented using the Lyapunov technique, since little was discussed in the literature about the application of NN in direct closed-loop applications that yield guaranteed performance until the work of Jagannathan (1994). By guaranteed we mean that both the tracking errors and the parameter estimates are bounded. This approach will indeed overcome the sector-bound restriction that is common in the discrete-time control literature. In addition, note that in the continuous-time case, the Lyapunov function is chosen so that its derivative is linear in the parameter error (provided that the system is linear in the parameters) and in the derivative of the parameter estimates (Kanellakopoulos

1994). This crucial property is not present in the difference of a discrete-time Lyapunov function which is a major problem. However, in this section, this problem is overcome by appropriately combining the terms and completing the squares in the first difference of the Lyapunov function. Finally, this section will set the stage for the more advanced NN-based adaptive controllers.

The adaptive scheme is composed of an NN incorporated into a dynamical system, where the structure comes from tracking error and passivity notions. It is shown that the delta rule-based tuning algorithm yields a passive NN. This, if coupled with the dissipativity of the dynamical system, guarantees the boundedness of all the signals in the closed-loop system under a PE condition (Section 3.1.4.2). However, PE is difficult to guarantee in an adaptive system for robust performance. Unfortunately, if PE does not hold, the delta rule-based gradient tuning schemes generally do not guarantee tracking and boundedness of the NN weights. Moreover, it is found here that the maximum permissible tuning rate for delta rule-based schemes decreases with an increase in the number of hidden-layer neurons; this is a major drawback. A projection algorithm is shown to easily correct the problem. New modified NN tuning algorithms are introduced to avoid the need for PE by making the NN controller robust, that is, state strict passive (Section 3.1.4.3).

3.1.1 DYNAMICS OF THE m TH ORDER MULTI-INPUT AND MULTI-OUTPUT DISCRETE-TIME NONLINEAR SYSTEM

Consider an m th order multi-input and multi-output (MIMO) discrete-time nonlinear system to be controlled, given by

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= f(x(k)) + \beta_0 u(k) + d(k) \end{aligned} \tag{3.3}$$

where $x(k) = [x_1(k) \dots x_n(k)]^T$ with $x_i(k) \in \mathbb{R}^m$, $i = 1, \dots, n$, $\beta_0 \in \mathbb{R}^{m \times m}$, $u(k) \in \mathbb{R}^{m \times 1}$, and $d(k) \in \mathbb{R}^m$ denotes a disturbance vector acting on the system at the instant k , with $|d(k)| \leq d_M$ being a known constant. The nonlinear function $f(\cdot)$ is assumed unknown. In this section, we consider the class of systems where the control input $u(k)$ directly enters the last equation in (3.3). In Section 3.3 we consider the more complex case where $x_n(k+1)$ depends upon $g(x(k))u(k)$ with the control influence function $g(\cdot)$ unknown.

Many system dynamics are naturally modeled in continuous-time. Unfortunately, the exact discretization of the continuous-time Brunovsky form does not yield the discrete-time Brunovsky form, but a more general discrete-time system of the form $x(k+1) = F(x(k), u(k))$, $y(k) = H(x(k), u(k))$. Under

certain reachability and involutivity conditions, this may be converted to the discrete-time. A suitable transformation is required to accomplish this which is presented for continuous-time systems in Zhang et al. (1998). This remains to be done in discrete-time systems.

Given a desired trajectory $x_{nd}(k)$ and its delayed values, define the tracking error as

$$e_n(k) = x_n(k) - x_{nd}(k) \quad (3.4)$$

It is typical in robotics to define a so-called filtered tracking error as $r(k) \in \Re^m$ and given by

$$r(k) = e_n(k) + \lambda_1 e_{n-1}(k) + \cdots + \lambda_{n-1} e_1(k) \quad (3.5)$$

where $e_{n-1}(k), \dots, e_1(k)$, are the delayed values of the error $e_n(k)$, and $\lambda_1, \dots, \lambda_{n-1}$, are constant matrices selected so that $|z^{n-1} + \lambda_1 z^{n-2} + \cdots + \lambda_{n-1}|$ is stable. Equation 3.5 can be further expressed as

$$r(k+1) = e_n(k+1) + \lambda_1 e_{n-1}(k-1) + \cdots + \lambda_{n-1} e_1(k+1) \quad (3.6)$$

Using (3.3) in (3.6), the dynamics of the MIMO system can be written in terms of the tracking error as

$$\begin{aligned} r(k+1) = & f(x(k)) - x_{nd}(k+1) + \lambda_1 e_n(k) + \cdots + \lambda_{n-1} e_2(k) \\ & + \beta_0 u(k) + d(k) \end{aligned} \quad (3.7)$$

Define the control input $u(k)$ in (3.7) as

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{f}(x(k)) + k_v r(k) - \lambda_1 e_n(k) - \cdots - \lambda_{n-1} e_2(k)] \quad (3.8)$$

with the closed-loop gain matrix k_v and $\hat{f}(x(k))$ an estimate of $f(x(k))$. Then the closed-loop error system becomes

$$r(k+1) = k_v r(k) + \tilde{f}(x(k)) + d(k) \quad (3.9)$$

where the functional estimation error is given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)) \quad (3.10)$$

This is an error system in which the filtered tracking error is driven by the functional estimation error and the unknown disturbances.

In the remainder of this section, (3.9) is used to focus on selecting NN tuning algorithms that guarantee the stability of the output tracking error $r(k)$ in (3.5). Then, since (3.5), where the input is considered as $r(k)$ and the output as $e(k)$, describes a stable system by using the notion of operator gain (Slotine and Li 1991) one can guarantee that $e(k)$ exhibits stable behavior. In fact, (3.5) can be rewritten as

$$\bar{x}(k+1) = A\bar{x}(k) + Br(k) \quad (3.11)$$

where

$$\bar{x}(k) = [e_1(k), \dots, e_{n-1}(k)]^T$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \cdot & \cdot & \cdot \\ -\lambda_{n-1} & \cdot & -\lambda_1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

One may show using the notion of operator gain that

$$\|e_1(k)\| \leq \frac{\|r(k)\|}{\Lambda_{\min}(A)}, \dots, \|e_n(k)\| \leq \frac{\|r(k)\|}{\Lambda_{\min}(A)} \quad (3.12)$$

with $\Lambda_{\min}(A)$ the minimum singular value of matrix A .

3.1.2 ONE-LAYER NN CONTROLLER DESIGN

In this section, the one-layer NN is considered as a first step to bridging the gap between the discrete-time adaptive control presented in the previous chapter and NN control. In the next section, we cover the multilayer discrete-time NN for control and present our results. In the one-layer case, the tunable weights enter in a linear fashion. The one-layer case for discrete-time tuning is covered in Sira-Ramirez and Zak (1991) and in Sadegh (1993). Even though discrete-time controller development is presented in Sira-Ramirez and Zak (1991), Lyapunov stability analysis is not discussed.

In this section, stability analysis by Lyapunov's direct method is carried out for a family of weight-tuning algorithms for NN controller design developed based on the delta rule. These tuning paradigms yield a passive NN, yet PE is

generally needed for suitable performance. Unfortunately PE cannot generally be tested for or guaranteed in the inputs, so that these delta-rule-based weight-tuning algorithms are generally doomed to failure. Modified tuning paradigms are proposed to make the NN robust so that PE is not needed. Finally, for guaranteed stability, the delta-rule-based NN weight-tuning algorithms must slow down with an increase in the number of hidden-layer neurons. By employing a projection algorithm, it is shown that the tuning rate can be made independent of the size of the NN.

In order to formulate the discrete-time controller, the following stability notations are needed. Consider the linear discrete time-varying system given by

$$\begin{aligned} x(k+1) &= A(k)x(k) + B(k)u(k) \\ y(k) &= C(k)x(k) \end{aligned} \tag{3.13}$$

where $A(k)$, $B(k)$, and $C(k)$ are appropriately dimensioned matrices.

Lemma 3.1.1: Define $\psi(k_1, k_0)$ as the state transition matrix corresponding to $A(k)$ for the system (3.13), that is, $\psi(k_1, k_0) = \prod_{k=k_0}^{k_1-1} A(k)$. Then if $\|\psi(k_1, k_0)\| < 1$, $\forall k_1, k_0 \geq 0$, the system (3.13) is exponentially stable.

Proof: See Ioannou and Kokotovic (1983).

3.1.2.1 NN Controller Design

Assume that there exist constant target weights W for one-layer NN so that the nonlinear function in (3.3) can be written as

$$f(x(k)) = W^T \phi(x(k)) + \varepsilon(k) \tag{3.14}$$

where $\phi(x(k))$ provides a suitable basis and $\|\varepsilon(k)\| < \varepsilon_N$, with the bounding constant ε_N known. Unless the network is “minimal,” the target weights may not be unique (Sontag 1992; Sussmann 1992). The best weights may then be defined as those which minimize the supremum norm over S of $\varepsilon(k)$. This issue is not a major concern here as only the existence of such target weights is important; their actual values are not required. This assumption is similar to Erzberger’s assumptions in the linear-in-the-parameters adaptive control. The major difference is that, while the Erzberger’s assumptions often do not hold, the approximation properties of NN guarantee that the target weights always exist if $f(x)$ is continuous over a compact set.

For suitable approximation properties, it is necessary to select a large enough number of hidden-layer neurons. It is not known how to compute this

number for a general fully connected NN; however, for cerebellar model articulation controller (CMAC) NN the required number of hidden-layer neurons for approximation to a desired degree of accuracy is given in Commuri and Lewis (1996). This scenario allows one to select a simple NN structure and thereafter compensate for the increased magnitude of ε_N by using the gain term k_v , as will be seen.

3.1.2.2 Structure of the NN and Error System Dynamics

Define the NN functional estimate in the controller (3.8) by

$$\hat{f}(x(k)) = \hat{W}^T(k)\phi(k) \quad (3.15)$$

where $\hat{W}(k)$ is the current value of the weights. This yields the controller structure shown in Figure 3.1. Processing the output of the plant through a series of delays provides the past values of the output. By feeding these as inputs to the NN the nonlinear function in (3.3) can be suitably approximated. Thus the NN controller derived in a straightforward manner using filtered error

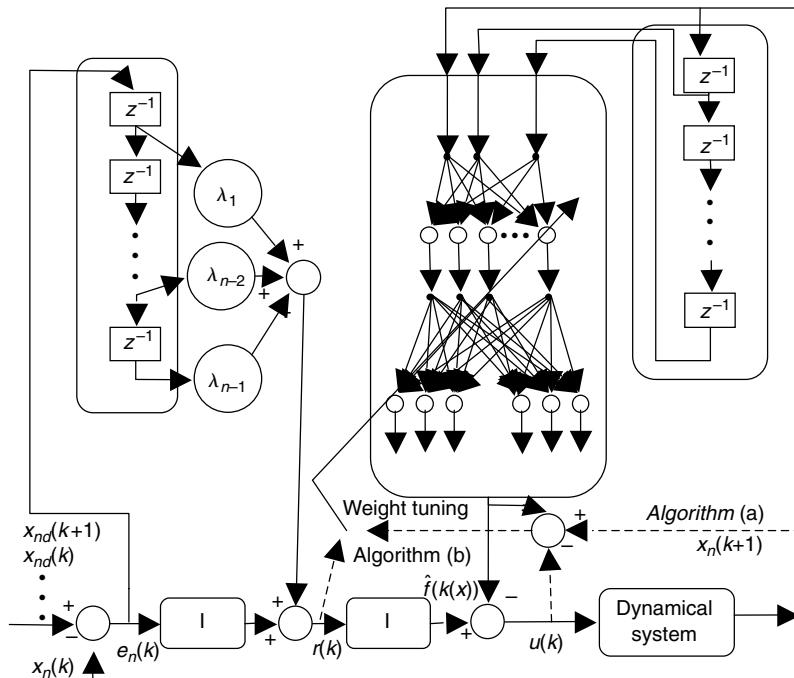


FIGURE 3.1 One-layer NN controller structure.

notions naturally provides a dynamical NN structure. Note that neither input $u(k)$ or its past values are needed by the NN. The next step is to determine the weight updates so that the tracking performance of the closed-loop filtered error dynamics is guaranteed.

Let W be the unknown constant weights required for the approximation to hold in (3.7) and assume that they are bounded by known values so that

$$\|W\| \leq W_{\max} \quad (3.16)$$

Then, the error in the weights during estimation, also called the weight estimation error, is given by

$$\tilde{W}(k) = W - \hat{W}(k) \quad (3.17)$$

Fact 3.1.1: The activation functions are bounded by known positive values so that

$$\|\phi(x(k))\| \leq \phi_{\max} \quad \text{and} \quad \|\tilde{\phi}(x(k))\| \leq \tilde{\phi}_{\max}$$

Select the control input $u(k)$ for the system (3.3) to be

$$u(k) = \beta_0^{-1}[x_{nd}(k+1) - \hat{W}^T(k)\phi(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) + k_v r(k)] \quad (3.18)$$

Then the closed-loop filtered error dynamics become

$$r(k+1) = k_v r(k) + \bar{e}_i(k) + \varepsilon(k) + d(k) \quad (3.19)$$

where the identification error denoted in (3.19) is given by

$$\bar{e}_i(k) = \tilde{W}^T(k)\phi(k) \quad (3.20)$$

The next step is to determine the weight-update laws for the error system derived in (3.19) so that the tracking performance of the closed-loop error dynamics is guaranteed.

3.1.2.3 Weight Updates of the NN for Guaranteed Tracking Performance

A family of NN-tuning paradigms including the delta rule that guarantee the stability of the closed-loop system (3.19) is presented in this section. It is required to demonstrate that the tracking error $r(k)$ is suitably small and that the NN weights $\hat{W}(k)$ remain bounded, for then the control $u(k)$ is bounded. In order to proceed further the following definitions are needed.

Lemma 3.1.2: If $A(k) = I - \alpha\phi(x(k))\phi^T(x(k))$ in (3.13), where $0 < \alpha < 2$ and $\phi(x(k))$ is the vector of basis functions, then $\|\psi(k_1, k_0)\| < 1$ is guaranteed if there is an $L > 0$ such that $\sum_{k=k_0}^{k_1+L-1} \phi(x(k))\phi^T(x(k)) > 0$ for all k . Then Lemma 3.1.2 guarantees the exponential stability of the system (3.13).

Proof: See Sadegh (1993).

Definition 3.1.1: An input sequence $x(k)$ is said to be persistently exciting (Ioannou and Kokotovic 1983) if there are $\lambda > 0$ and an integer $k_1 \geq 1$ such that

$$\lambda_{\min} \left[\sum_{k=k_0}^{k_1+k-1} \phi(x(k))\phi^T(x(k)) \right] > \lambda \quad \forall k_0 \geq 0 \quad (3.21)$$

where $\lambda_{\min}(P)$ represents the smallest eigenvalue of P .

Note that PE is exactly the stability condition needed in Lemma 3.1.2.

In the following, it is first taken that the NN reconstruction error-bound ε_N and the disturbance bound d_M are nonzero. Theorem 3.1.1 and in Table 3.1 gives two alternative weight-tuning algorithms, one based on a modified functional estimation error and the other based on the tracking error, showing that both the tracking error and the error in the weight estimates are bounded if a PE condition holds. This PE requirement is relaxed in Theorem 3.1.3.

Theorem 3.1.1 (One-layer Discrete-Time NN Controller Requiring PE Condition): Let the desired trajectory $x_{nd}(k)$ for (3.3) and the initial conditions be bounded in a compact set U . Let the NN functional reconstruction error and the disturbance bounds ε_N and d_M respectively be known constants. Consider the weight tuning provided by either

$$(a) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))\bar{f}^T(k) \quad (3.22)$$

where $\bar{f}(k)$, is defined as the functional augmented error and it is computed using

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}^T(k)\phi(x(k)) \quad (3.23)$$

or

$$(b) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))r^T(k+1) \quad (3.24)$$

TABLE 3.1
Discrete-Time Controller Using One-Layer NN: PE Required

The control input $u(k)$ is

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{W}^T(k)\phi(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) + k_v r(k)]$$

The weight tuning is provided by either

$$(a) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))\bar{f}^T(k),$$

where $\bar{f}(k)$, is defined as the functional augmented error computed using

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}^T(k)\phi(x(k))$$

or

$$(b) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))r^T(k+1)$$

where $\alpha > 0$ denoting constant learning rate parameter or adaptation gain.

where $\alpha > 0$ denotes the constant learning rate parameter or adaptation gain. Let the hidden-layer output vector, $\phi(x(k))$ be persistently exciting. Then the filtered tracking error $r(k)$ in (3.5) and the errors in weight estimates $\tilde{W}(k)$ are uniformly ultimately bounded (UUB) provided the following conditions hold:

$$\begin{aligned} \alpha \|\phi(x(k))\|^2 &< 1 \\ k_{v \max} &< \frac{1}{\sqrt{\eta}} \end{aligned} \tag{3.25}$$

where η is given for Algorithm (a) as

$$\eta = 1 + \frac{1}{1 - \alpha \|\phi(x(k))\|^2} \tag{3.26}$$

and for algorithm (b) as

$$\eta = \frac{1}{1 - \alpha \|\phi(x(k))\|^2} \tag{3.27}$$

Proof: See Jagannathan and Lewis (1996a).

Outline of proof. In the proof, it is first demonstrated by using a Lyapunov function, the tracking error dynamics (3.19) and the weight updates (3.22) for

Algorithm (a), and (3.24) for Algorithm (b), that the filtered tracking error $r(k)$ is bounded. In addition, it is necessary to show that the weight estimates are also bounded. In order to prove the boundedness of the error in the weight estimates, the PE condition, the bound on the tracking error, and the dynamics in the error in weight estimates are considered. Using these conditions it is shown that the weight estimates are bounded.

Note from the tracking and weight estimation error bounds (Jagannathan and Lewis 1996a) that the tracking error increases with the NN reconstruction error-bound ε_N and the disturbance-bound, d_M , yet small tracking errors may be achieved by selecting small gains k_v . In other words, placing the closed-loop error poles closer to the origin inside the unit circle forces smaller tracking errors. Selecting $k_{v\max} = 0$ results in a deadbeat controller, but this should be avoided since it is not robust.

Remarks:

1. It is important to note that unlike other standard works in discrete-time adaptive control (Astrom and Wittenmark 1989) in this theorem there is no CE assumption for the controller. In the former, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function used in the proof (Jagannathan and Lewis 1996a) is of this form

$$J = r^T(k)r(k) + \frac{1}{\alpha} \text{tr}[\tilde{W}^T(k)\tilde{W}(k)]$$

which weights the tracking errors $r(k)$ and the weight estimation errors for the controller, $\tilde{W}(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

2. The weight-updating rules (3.22) and (3.24) are nonstandard schemes that were derived from Lyapunov analysis and do not include an extra term normally used to provide robustness due to the coupling in the proof between the tracking and weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required if the PE condition is applied.

3. Condition (3.25) can be checked easily. The NN learning rate and the maximum singular value of the controller gain $k_{v\max}$ have to be satisfied (3.25) in order for the closed-loop system to be stable. This relationship between the controller gain and the NN learning rate is unique. This condition states that for faster tuning of NN weights the closed-loop poles should be inside the unit disc. By contrast, such a relationship does not exist between controller design and NN learning rate parameters in continuous-time. Therefore, the design parameters for the adaptive NN controllers in continuous-time are normally selected arbitrarily.
4. *Initial condition requirement.* The approximation accuracy of the NN determines the allowed magnitude of the initial tracking error $r(0)$. For a larger NN with more hidden-layer units, ε_N is small for a set, S_r , with a large radius. Thus the allowed initial condition set is larger. Likewise a more active desired trajectory containing higher-frequency components results in a larger acceleration $x_{nd}(k+1)$ and yields a larger bound on the initial allowable tracking errors, thereby decreasing the set S_r . Since the proposed controller includes a conventional controller along with an NN, it is important to note the dependence of the set of initial allowable tracking errors on the controller gains. Though the initial condition requirement may seem to be cast in terms of complex quantities, it merely indicates that the NN should be large enough in terms of the number L of hidden-layer neurons for all practical purposes. Therefore, in design, one would select a suitable value of L based on experience, run a simulation to test the controller then repeat with a large value of L . A suitable value of L is selected for the proposed NN controller implementation based on experience and it is tested to ensure that there is no appreciable increase in performance is noted if this number is increased.
5. *Weight initialization and online tuning.* In the NN control scheme derived in the book there is no off-line learning phase for the NN in general except for the work presented in Chapter 9. The weights are simply initialized at zero, for then the NN controller structure illustrated in Figure 3.1 shows that the controller is just a standard industrial controller. Gains for the available industrial controllers are normally selected such that the overall system is stable in a limited operating region. Therefore, the closed-loop system remains stable if the allowable initial conditions are selected within this operating region and until the NN begins to learn. The weights of the NN are tuned online in real-time as the system tracks the desired trajectory. As the NN learns the unknown nonlinear dynamics, $f(x(k))$, the

tracking performance improves. This is a significant improvement over other NN control techniques where one must find some initial stabilizing weights, generally an impossible feat for complex nonlinear system. Moreover, the proposed approach is modular in the sense that the NN inner loop can be added to the existing industrial controllers without a significant change in the design. By adding the NN inner loop, the performance of the tracking performance is improved as the NN learns.

Example 3.1.1 (NN Control of Continuous-Time Nonlinear System): Consider a continuous-time nonlinear system, the objective being to control a MIMO system by using a one-layer NN controller in discrete-time. It is important to note that it is generally very difficult to discretize a nonlinear system and therefore to offer proofs of stability. Moreover, the NN controller that is derived herein requires no a priori knowledge of the system dynamics unlike conventional adaptive control, nor is any initial or off-line learning phase needed.

Consider the nonlinear system described by

$$\begin{aligned}\dot{X}_1 &= X_2 \\ X_2 &= F(X_1, X_2) + U\end{aligned}\tag{3.28}$$

where $X_1 = [x_1, x_2]^T$, $X_2 = [x_3, x_4]^T$, $U = [u_1, u_2]^T$, and the nonlinear function in (3.28) is described by

$$F(X_1, X_2) = [M(X_1)]^{-1}G(X_1, X_2)$$

with

$$M(X_1) = \begin{bmatrix} (b_1 + b_2)a_1^2 + b_2a_2^2 + 2b_2a_1a_2 \cos(x_2) & b_2a_2^2 + b_2a_1a_2 \cos(x_2) \\ b_2a_2^2 + b_2a_1a_2 \cos(x_2) & b_2a_2^2 \end{bmatrix}\tag{3.29}$$

and

$$G(X_1, X_2) = \begin{bmatrix} -b_2a_1a_2(2x_3x_4 + x_4^2) \sin(x_2) + 9.8(b_1 + b_2)a_1 \cos(x_1) + 9.8b_2a_2 \cos(x_1 + x_2) \\ b_2a_1a_2x_1^2 \sin(x_2) + 9.8b_2a_2 \cos(x_1 + x_2) \end{bmatrix}\tag{3.30}$$

The parameters for the nonlinear system were selected as $a_1 = a_2 = 1$, $b_1 = b_2 = 1$. Desired sinusoidal, $\sin(2\pi t)/25$ and cosine inputs, $\cos(2\pi t)/25$

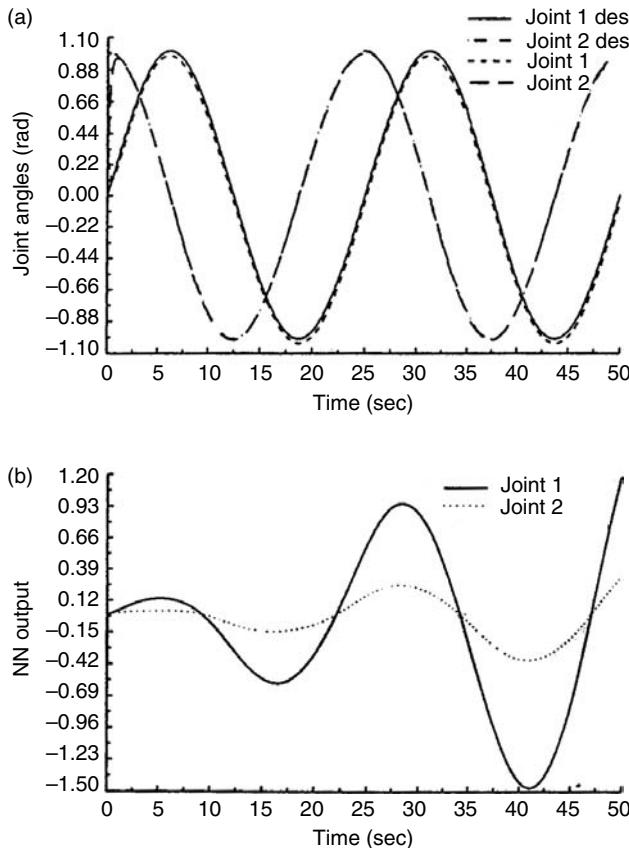


FIGURE 3.2 Response of the NN controller with delta-rule weight tuning and small α .
(a) Actual and desired joint angles. (b) NN outputs.

were preselected for the axis 1 and 2, respectively. The gains of the PD controller in continuous-time were chosen as $k_v = \text{diag}(20, 20)$ with $\Lambda = \text{diag}(5, 5)$ and a sampling interval of 10 msec was considered. A one-layer NN was selected with 25 hidden-layer neurons. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for X_1 were chosen to be $[0.5, 0.1]^T$ and the weights were initialized to zero. No off-line learning is performed initially to train the networks. Figure 3.2 presents the tracking response of the NN controller with delta-rule weight tuning (3.1.22) and small adaptation gain $\alpha = 0.1$. From the figure it can be seen that the delta-rule-based weight tuning performs impressively.

3.1.2.4 Projection Algorithm

The adaptation gain $\alpha > 0$ is a constant parameter in the update laws presented in (3.22) and (3.24). These update laws correspond to the gradient rule (Åström and Wittenmark 1989). The theorem reveals that update tuning mechanisms employing the gradient rule have a major drawback. In fact, using (3.25) an upper bound on the adaptation gain can be obtained as

$$\alpha < \frac{1}{\|\phi(x(k))\|^2} \quad (3.31)$$

Here $\phi(x(k)) \in \Re^{N_2}$, with N_2 being the number of hidden-layer neurons. It is evident that the upper bound on the adaptation gain depends upon the number of hidden-layer neurons. Specifically, if there are N_2 hidden-layer neurons and if the maximum value of each hidden-node output is taken as unity (as for the sigmoid) then in order to assure stability of the closed-loop system, the bound on the adaptation gain is given by

$$0 < \alpha < \frac{1}{N_2} \quad (3.32)$$

In other words, the upper bound on the adaptation gain for the case of delta-rule-based tuning decreases with an increase in number of hidden-layer nodes, so that the adaptation slows down due to the addition of more hidden-layer neurons for guaranteed performance. The phenomenon of large NN requiring very slow learning rates has often been encountered in the practical NN literature (Rumelhart et al. 1990; Mpitsos and Burton 1992) but never explained adequately.

This major drawback can be easily overcome by modifying the update law in order to obtain a projection algorithm (Goodwin and Sin, 1984; Åström and Wittenmark 1989). Replace the constant adaptation gain by

$$\alpha = \frac{\xi}{\zeta + \|\phi(x(k))\|^2} \quad (3.33)$$

where $0 < \xi < 1$ and $\zeta > 0$ are constants. Note that ξ is now the new adaptation gain, and it is always true that

$$\frac{\xi}{\zeta + \|\phi(x(k))\|^2} \|\phi(x(k))\|^2 < 1 \quad (3.34)$$

hence guaranteeing (3.34) for every N_2 .

Finally, it is very interesting to note that for identification purposes alone the upper bound on the adaptation gain in (3.25) is well known to be less than two (Goodwin and Sin 1984; Åström and Wittenmark 1989), whereas, for the purpose of control the upper bound on the adaptation gain is further reduced to one owing to the constraints imposed by the tracking error system.

Example 3.1.2 (Effect on Learning Rate on NN Size): The objective here is to demonstrate for Example 3.1.1 that the learning rate for the delta rule employed at each layer in fact decreases with an increase in the number of hidden-layer neurons in that layer. In addition, it is demonstrated that the problem can be avoided by using a projection algorithm.

The adaptation gains for the weight tuning are increased from 0.0005 to 0.1 for the case of delta rule (3.24), and $\xi = 0.5$ with $\zeta = 0.001$ for the case of the delta rule with projection algorithm with (3.24) and (3.33). Figure 3.3 and Figure 3.4 present the tracking responses of the controllers with the delta rule and projection algorithm, respectively. It is clear that the controller using the delta rule at each layer performs equally well with the projection algorithm when the value of the adaptation gain is small so that (3.25) is satisfied. However, initially large values of the weights were needed not only for the delta rule with small (shown in Figure 3.2) but also for the projection algorithm with large adaptation gain (Figure 3.3).

Note that from Figure 3.3 the tracking performance is extremely impressive. Figure 3.4 illustrates the response of the NN controller when the delta rule is employed with the adaptation gain increased from 0.1 to 0.51. From Figure 3.4 it is evident that weight tuning using the delta rule becomes unstable. Note that the representative weight estimates, as illustrated in Figure 3.4b, of the NN are unbounded in this case. This demonstrates that the adaptation gain in the case of the delta rule must decrease with an increase in the hidden-layer neurons.

3.1.2.5 Ideal Case: No Disturbances and No NN Reconstruction Errors

The next result discusses the behavior of the closed-loop system in the idealized case of no NN functional reconstruction errors without any unmodeled disturbances in the dynamics of the system. In this case, PE is not needed. Note that it guarantees asymptotic stability (AS) and not UUB.

Theorem 3.1.2 (Ideal Case with No NN Reconstruction Errors and Bounded Disturbances): Let the desired trajectory $x_{nd}(k)$ for (3.6) be bounded and the NN functional reconstruction error bound ε_N and the disturbance-bound d_M be

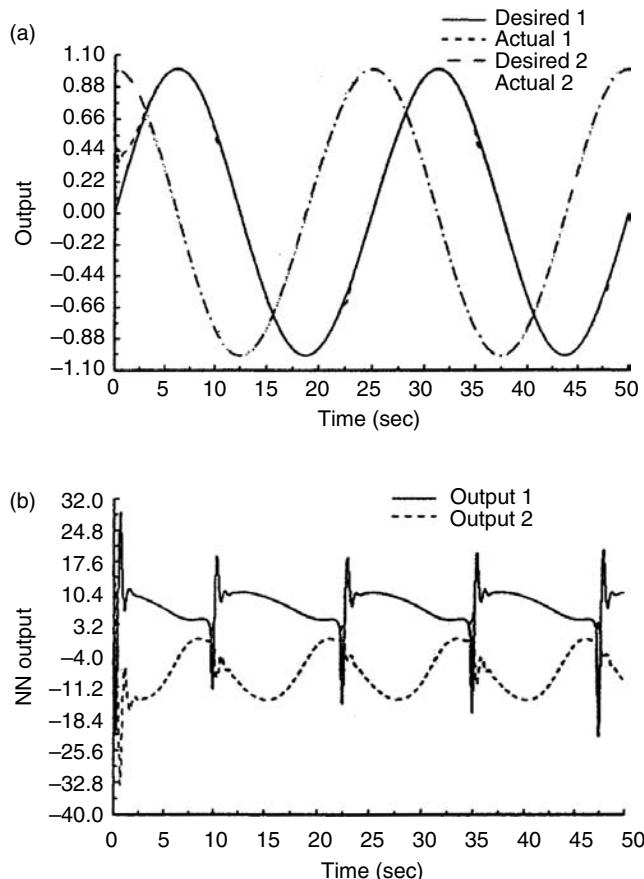


FIGURE 3.3 Response of the NN controller with delta rule weight-tuning and projection algorithm. (a) Actual and desired joint angles. (b) NN outputs.

equal to zero. Let the parameter tuning be provided by either

$$(a) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha \phi(x(k)) \bar{f}^T(k) \quad (3.35)$$

or

$$(b) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha \phi(x(k)) r^T(k+1) \quad (3.36)$$

where $\alpha > 0$ is a constant adaptation.

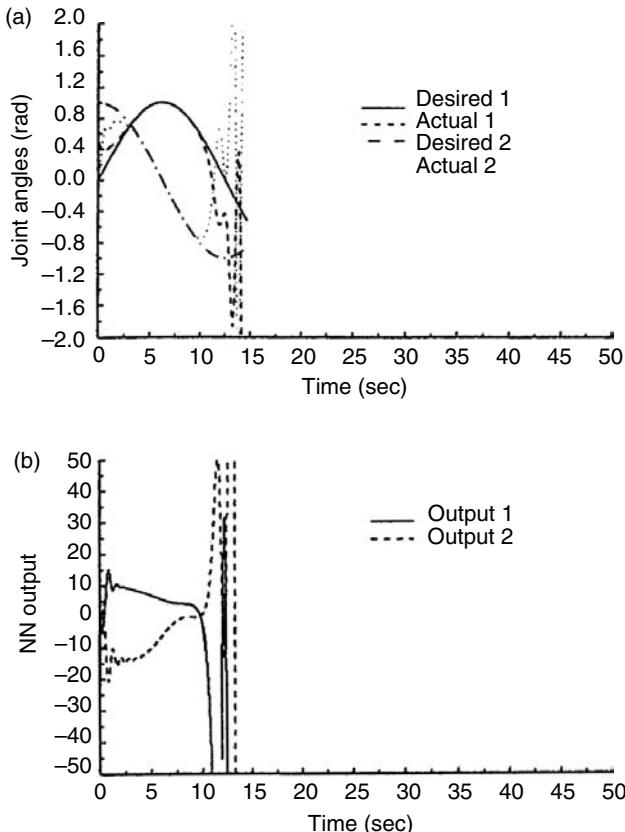


FIGURE 3.4 Response of the NN controller with delta rule weight-tuning and large α .
 (a) Actual and desired joint angles. (b) NN outputs.

Then the tracking errors $r(k)$ in (3.5) asymptotically approach zero and the NN weight estimates are bounded provided the condition (3.25) of Theorem 3.1.1 holds, with η as given for the Algorithms (a) and (b) by (3.26) and (3.27), respectively.

Proof: Since the functional reconstruction error and the disturbances are all zero, these new assumptions yield the error systems

$$r(k+1) = k_v r(k) + \bar{e}_i(k) \quad (3.37)$$

Algorithm (a): Selecting the Lyapunov function candidate given in the outline of the proof of the previous theorem with the new assumptions and the update-tuning mechanism in (3.35) results in

$$\begin{aligned}
\Delta J &= -r^T(k)(I - k_v^T k_v)r(k) + \bar{e}_i^T(k)\bar{e}_i(k) + 2[k_v r(k)]^T \bar{e}_i(k) \\
&\quad - [2 - \alpha \phi^T(x(k))\phi(x(k))] \bar{e}_i^T(k)\bar{e}_i(k) \\
&\leq -r^T(k) \left[I - \left(1 + \frac{1}{1 - \alpha \|\phi(x(k))\|^2} \right) k_v^T k_v \right] \\
&\quad \times r(k) - [1 - \alpha \|\phi(x(k))\|^2][\bar{e}_i(k) - k_v r(k)]^T [\bar{e}_i(k) - k_v r(k)] \\
&\leq -(1 - \eta k_{v \max}^2) \|r(k)\|^2 - [1 - \alpha \|\phi(k)\|^2] \|\bar{e}_i(k) - k_v r(k)\|^2 \quad (3.38)
\end{aligned}$$

where η is given by (3.26). Since $J > 0$ and $\Delta J \leq 0$, stability in the sense of Lyapunov is seen, provided the conditions (3.25) hold, so that $r(k)$ and $\tilde{W}(k)$ (and hence $\hat{\theta}(k)$) are bounded if $r(k_0)$ and $\tilde{W}(k_0)$ are bounded in the compact set U . In addition, on summing both sides of (3.38), one notes that as $k \rightarrow \infty$ in $e(k+1) = k_v e(k) + \bar{e}_i(k)$, the tracking error $\|r(k)\| \rightarrow 0$ (Lin and Narendra 1980).

Algorithm (b): For the case of the weight-tuning mechanism given in (3.36), select the Lyapunov function candidate given in the outline of the proof of the previous theorem, and use the new assumptions as well as the update law to obtain

$$\begin{aligned}
\Delta J &= -r^T(k)\{I - [1 + \alpha \phi^T(x(k))\phi(x(k))]k_v^T k_v\}r(k) + 2\alpha \phi^T(x(k))\phi(x(k)) \\
&\quad \times [k_v r(k)]^T \bar{e}_i(k) - [1 - \alpha \phi^T(x(k))\phi(x(k))] \bar{e}_i^T(k)\bar{e}_i(k) \\
&= -r^T(k) \left\{ I - [1 - \alpha \phi^T(x(k))\phi(x(k))] + \frac{[\alpha \phi^T(x(k))\phi(x(k))]^2}{[1 - \alpha \phi^T(x(k))\phi(x(k))]} \right. \\
&\quad \left. \times k_v^T k_v \right\} r(k) - [1 - \alpha \phi^T(x(k))\phi(x(k))] \\
&\quad \times \left[\bar{e}_i(k) - \frac{[\alpha \phi^T(x(k))\phi(x(k))]^2}{[1 - \alpha \phi^T(x(k))\phi(x(k))]} k_v r(k) \right]^T \\
&\quad \times \left[\bar{e}_i(k) - \frac{[\alpha \phi^T(x(k))\phi(x(k))]^2}{[1 - \alpha \phi^T(x(k))\phi(x(k))]} k_v r(k) \right] \\
&\leq -(1 - \eta k_{v \max}^2) \|r(k)\|^2 \\
&\quad - [1 - \alpha \|\phi(x(k))\|^2] \times \left\| \bar{e}_i(k) - \frac{\alpha \|\phi(x(k))\|^2}{1 - \alpha \|\phi(x(k))\|^2} k_v r(k) \right\|^2 \quad (3.39)
\end{aligned}$$

where η is given in (3.27). Since $J > 0$ and $\Delta J \leq 0$, this shows stability in the sense of Lyapunov, provided the conditions (3.25) holds, so that $r(k)$ and $\hat{W}(k)$ (and hence $\tilde{W}(k)$) are bounded if $r(k_0)$ and $\tilde{W}(k_0)$ are bounded in the compact set U . In addition, on summing both sides of (3.39), one notes that as $k \rightarrow \infty$ in $e(k+1) = k_v e(k) + \bar{e}_i(k)$, the tracking error $\|r(k)\| \rightarrow 0$ (Lin and Narendra 1980).

Note that now for guaranteed closed-loop stability, it is not necessary that the basis function vector be PE. Equation 3.35 and Equation 3.36 are nothing but the delta-rule-based weight-tuning algorithms. Theorem 3.1.2 indicates that delta-rule-based weight updates suffice when the NN functional reconstruction error $\varepsilon(k)$ and disturbance $d(k)$ are zero. However, Theorem 3.1.1 reveals the failure of standard delta-rule-based weight tuning in the presence of NN reconstruction errors and bounded disturbances. Therefore, delta-rule-based tuning updates used in an NN that cannot exactly reconstruct certain unknown parameters because of the presence of nonlinearities $f(\cdot)$ or uncertainties in the estimation process with bounded unmodeled disturbances cannot be guaranteed to yield bounded estimates. Then the PE condition is required to guarantee boundedness of the weight estimates. However, it is very difficult to guarantee or verify the PE of the hidden-layer output functions $\phi(x(k))$ and this problem is compounded by the presence of hidden-layers in the case of multilayer NN. This possible unboundedness of the weight estimates (cf. parameter estimates in adaptive control) when PE fails to hold is known as parameter drift (Åström and Wittenmark 1989; Narendra and Annaswamy 1989). In the next section, improved weight tuning paradigms for the NN are presented so that PE is not required.

3.1.2.6 Parameter Tuning Modification for Relaxation of PE Condition

Approaches such as σ -modification (Ioannou and Kokotovic 1983) or ε -modification (Narendra and Annaswamy 1987) are available for the robust adaptive control of continuous systems, for which the PE condition is not needed. The property of robustness in the update laws is needed or arises from the persistent excitation due to the disturbances or changes in the reference signal. A one-layer NN with continuous weight-update laws and ε -modification was developed (Lewis et al. 1995) and the UUB of both the tracking error and error in weight estimates was demonstrated. Finally, schemes in discrete-time are guaranteed to perform well by empirical studies only with no convergence or stability proofs, whereas the continuous-time counterparts are guaranteed to perform successfully both analytically and by simulation studies. Therefore, modification to the standard weight-tuning mechanisms in discrete-time to avoid the necessity of PE is investigated in Jagannathan and Lewis (1996a,

TABLE 3.2**Discrete-Time Controller Using One-Layer NN: PE Not Required**

The control input $u(k)$ is

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{W}^T(k)\phi(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) + k_v r(k)]$$

The weight tuning is given either by

$$(a) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))\bar{f}^T(k) - \Gamma \|I - \alpha\phi(x(k))\phi^T(x(k))\| \hat{W}(k)$$

where $\bar{f}(k)$, is defined as the functional augmented error given by

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}^T(k)\phi(x(k))$$

or

$$(b) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))r^T(k+1) - \Gamma \|I - \alpha\phi(x(k))\phi^T(x(k))\| \hat{W}(k)$$

with $\alpha = \xi/(\zeta + \|\phi(x(k))\|)$, where $\zeta > 0$ and $0 < \xi < 1$ denote learning rate parameters or adaptation gain and $\Gamma > 0$ a design parameter.

1996c) for the first time. Since then, numerous researchers have started demonstrating stability results in discrete-time.

In Jagannathan and Lewis (1996a, 1996c), an approach similar to ε -modification was derived for discrete-time NN control. The following theorem from that paper shows two tuning schemes that overcome the need for PE. Table 3.2 also presents the NN Controller.

Theorem 3.1.3 (One-Layer Discrete-Time NN Controller with No PE Condition): Assume the hypotheses presented in Theorem 3.1.1 and consider the modified tuning algorithms provided by either

$$(a) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))\bar{f}^T(k) - \Gamma \|I - \alpha\phi(x(k))\phi^T(x(k))\| \hat{W}(k) \quad (3.40)$$

or

$$(b) \quad \hat{W}(k+1) = \hat{W}(k) + \alpha\phi(x(k))r^T(k+1) - \Gamma \|I - \alpha\phi(x(k))\phi^T(x(k))\| \hat{W}(k) \quad (3.41)$$

with $\Gamma > 0$ a design parameter. Then the filtered tracking error $r(k)$ and the NN weight estimates $\hat{W}(k)$ are UUB and the practical bounds for $r(k)$ and $\hat{W}(k)$,

denoted here by b_t and b_W , respectively, are given by

$$\begin{aligned} b_t = & \frac{1}{(1 - \eta k_{v \max}^2)} + \left[\kappa k_{v \max} (\varepsilon_N + d_M) \right. \\ & \left. + \sqrt{\kappa^2 k_{v \max}^2 (\varepsilon_N + d_M)^2 + \rho (1 - \eta k_{v \max}^2)} \right] \end{aligned} \quad (3.42)$$

$$b_W = \frac{\Gamma(1 - \Gamma)W_{\max} + \sqrt{\Gamma^2(1 - \Gamma)^2 W_{\max}^2 + \Gamma(2 - \Gamma)\rho}}{\Gamma(2 - \Gamma)} \quad (3.43)$$

for algorithm (a), and

$$b_t = \frac{1}{1 - \bar{\sigma} k_{v \max}^2} \left[\gamma k_{v \max} + \sqrt{\rho_1 (1 - \bar{\sigma} k_{v \max}^2)} \right] \quad (3.44)$$

$$b_W = \frac{\Gamma(1 - \Gamma)W_{\max} + \sqrt{\Gamma^2(1 - \Gamma)^2 W_{\max}^2 + \Gamma(2 - \Gamma)\bar{\theta}}}{\Gamma(2 - \Gamma)} \quad (3.45)$$

for algorithm (b), provided the following conditions hold:

$$(a) \quad \alpha \|\phi(x(k))\|^2 < 1 \quad (3.46)$$

$$(b) \quad 0 < \Gamma < 1 \quad (3.47)$$

$$(c) \quad k_{v \max} < \frac{1}{\sqrt{\bar{\eta}}} \quad (3.48)$$

for algorithm (a),

$$k_{v \max} < \frac{1}{\sqrt{\bar{\sigma}}} \quad (3.49)$$

for algorithm (b), where η is given in (3.26) for algorithm (a) and $\bar{\sigma}$ for algorithm (b) is given by

$$\begin{aligned} \bar{\sigma} = & \eta + \frac{1}{(1 - \alpha \|\phi(x(k))\|^2)} [\Gamma^2(1 - \alpha \|\phi(x(k))\|^2)^2 \\ & + 2\alpha\Gamma \|\phi(x(k))\|^2 (1 - \alpha \|\phi(x(k))\|^2)] \end{aligned} \quad (3.50)$$

Note: The parameters $\alpha, \eta, \bar{\sigma}, \kappa$, and ρ are dependent upon the trajectory.

Proof: See Jagannathan and Lewis (1996a).

Outline of the proof: It is first demonstrated by using a Lyapunov function and the parameter updates (3.40) and (3.41) along with the tracking error dynamics (3.19), that the tracking error is bounded. Then it can be shown by simple manipulation that the errors in the weight estimates are also bounded without a PE condition. The parameter updates in Theorem 3.1.3 include an additional term similar to the ε -modification approach for the continuous-time systems. It is found that an exact replica of the ε -modification term used for the continuous-time case does not yield a bound on the parameters for the discrete-time case. However, using (3.40) and (3.41), it can be shown that both the tracking and NN weight estimation errors are bounded.

Note that the NN reconstruction error ε_N and the bounded disturbances d_M determine the bounds on $\|r(k)\|$ and $\|\tilde{W}(k)\|$. Small tracking error bounds may be achieved by placing the closed-loop poles inside the unit circle and near the origin through the selection of the largest gain eigenvalue k_{vmax} . On the other hand, the NN weight estimation error estimates are fundamentally bounded by W_{max} , the known bound on the ideal parameter W . The parameter Γ offers a design trade-off between the relative eventual magnitudes of $\|r(k)\|$ and $\|\tilde{W}(k)\|$; a smaller Γ yields a smaller $\|r(k)\|$ and a larger $\|\tilde{W}(k)\|$ and vice versa.

Remarks:

1. It is important to note that in this theorem also there is no CE assumption for the adaptive NN controller, in contrast to standard work in discrete-time adaptive control (Åström and Wittenmark 1989). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption and another Lyapunov function is selected that weighs only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in this proof, the Lyapunov function is selected similar to the one in Theorem 3.1.2 which weights the tracking errors, $r(k)$ and the weight estimation errors for the controller, $\tilde{W}(k)$. The proof is exceedingly complex due to the presence of several different variables and an extra term used for ensuring robustness. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

2. The NN weight-updating rules (3.40) and (3.41) are nonstandard schemes that were derived from Lyapunov analysis and do include an extra term referred to as discrete-time ε -modification (Jagannathan and Lewis 1996a), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and parameter estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required if the PE condition is applied.
3. Conditions (3.46) through (3.49) can be checked easily. The maximum singular value of the controller gain $k_{v \max}$ and the NN learning rate parameter have to satisfy (3.48) in order for the closed-loop system to be stable. This is a unique relationship between the controller gain and the parameter adaptation matrix. This condition states that for faster tuning of NN weights, the closed-loop poles should be far inside the unit disc. On the other hand, such constraints do not exist for controller design parameters and NN adaptation gains in continuous time. Therefore, the design parameters for the adaptive and NN controllers in continuous time are selected arbitrarily.
4. *Initial condition requirement.* In this theorem also the proposed controller includes a conventional controller along with an NN. Therefore it is important to note the dependence of the set of initial allowable tracking errors on the controller gains. Though the initial condition requirement may seem to be cast in terms of complex quantities, it merely indicates that the NN should be large enough in terms of the number L of hidden-layer neurons for all practical purposes. Therefore, in design, one would select a suitable value of L based on experience, run a simulation to test the controller then repeat with a large value of L . A suitable value of L is selected for the proposed NN controller implementation based on experience and it is tested to ensure that no appreciable increase in performance is noted if this number is increased.
5. *Weight initialization and online tuning.* In the NN control scheme derived in the book there is no off-line learning phase for the NN. The weights are simply initialized at zero, for then the NN controller structure shows that the controller is just a standard industrial controller. Gains for the available industrial controllers are normally selected such that the overall system is stable in a limited operating region. Therefore, the closed-loop system remains stable if the allowable initial conditions are selected within this operating region and until the NN begins to learn. The weights of the NN are tuned online in real-time as the system tracks the desired trajectory. As the

NN learns the unknown nonlinear dynamics, $f(x(k))$, the tracking performance improves.

Example 3.1.3 (NN Controller with Improved Weight Tuning): For Example 3.1.1, the response of the NN controller with the improved weight tuning (3.41) and projection algorithm (3.33) is presented in Figure 3.5. The design parameter Γ is selected as 0.01. Note that with improved weight tuning, the output of the NN remains bounded because the weights are guaranteed to remain bounded without the necessity of the PE condition. To study the contribution of the NN, Figure 3.6 shows the response of the PD controller

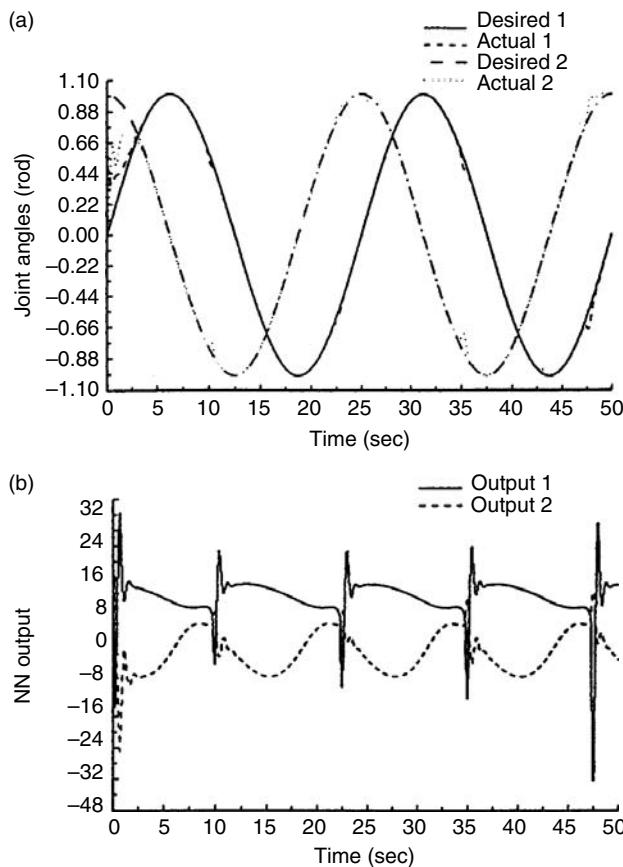


FIGURE 3.5 Response of the NN controller with improved weight-tuning and projection algorithm. (a) Actual and desired joint angles. (b) NN outputs.

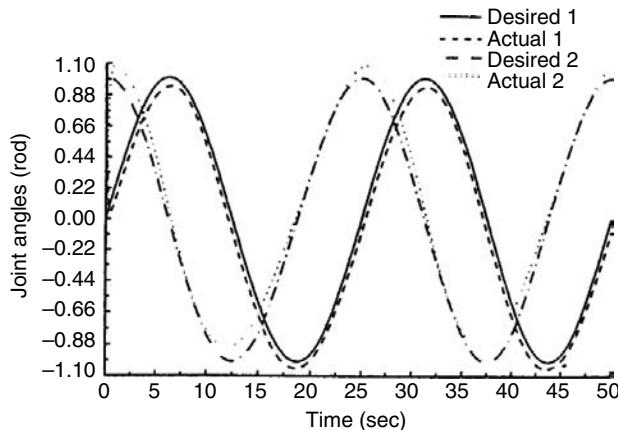


FIGURE 3.6 Response of the PD controller.

without the NN. From the figure, it is clear that the addition of the NN makes a significant improvement in the tracking performance.

Example 3.1.4 (NN Control of Discrete-Time Nonlinear System): Consider the first-order MIMO discrete-time nonlinear system described by

$$X(k+1) = F(X) + U(k) \quad (3.51)$$

where

$$X(k) = [x_1(k), x_2(k)]^T$$

$$F(X) = \begin{bmatrix} \frac{x_2(k)}{1+x_2^2(k)} \\ \frac{x_1(k)}{1+x_2^2(k)} \end{bmatrix}$$

and

$$U(k) = [u_1(k), u_2(k)]^T$$

The objective is to track a periodic step input of magnitude two units with a period of 30 sec. The elements of the diagonal matrix were chosen as $k_v = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ and a sampling interval of 10 msec was considered. A one-layer NN was selected with 12 hidden-layer nodes. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for the plant were chosen to be $[1, -1]^T$. The weights were initialized to zero with an initial threshold value of 3.0. The design parameter Γ is selected to be 0.01.

No learning is performed initially to train the networks. The design parameters for the projection algorithm (3.33) were selected to be $\xi = 0.5$ with $\zeta = 0.001$.

It is also found (not shown) in this example that the delta rule weight tuning performs very well when the learning rate is small. In addition, it was also observed during simulation studies that the learning rate for delta-rule-based weight tuning should decrease with an increase in the number of hidden-layer neurons. As expected, however, this problem is solved by using a projection algorithm. In this example, only results using the improved weight tuning are presented. The response of the controller with the improved weight tuning (3.41) with (3.33) is shown in Figure 3.7. Note from Figure 3.7, as expected, the performance of the controller is extremely impressive.

Let us consider the case when a bounded disturbance given by

$$w(k) = \begin{cases} 0.0, & 0 \leq kT_m < 12 \\ 0.1, & kT_m \geq 12 \end{cases} \quad (3.52)$$

is acting on the plant at the time instant k . Figure 3.8 presents the tracking response of the NN controllers with the improved weight tuning and projection algorithm. The magnitude of the disturbance can be increased but the value should be bounded. The value shown in (3.52) is employed for simulation purposes only. It can be seen from the figure that the bounded disturbance induces bounded tracking errors at the output of the plant. From the results, it can be inferred that the bounds presented and the theoretical claims were justified through simulation studies both in continuous- and discrete-time.

3.1.3 MULTILAYER NN CONTROLLER DESIGN

In the previous section, a one-layer NN is used for the controller design. In this section, a three-layer NN is considered and stability analysis is carried out for the closed-loop system (3.19). Thereafter, the stability analysis presented for a three-layer NN is extended for a multilayer NN having an arbitrary number of hidden layers. In this section, stability analysis by Lyapunov's direct method is performed for a family of multilayer NN weight tuning algorithms using a delta rule in each layer. These weight-tuning algorithms yield a passive net, yet PE condition is generally needed for suitable performance similar to the case of the one-layer NN. Unfortunately PE cannot be tested for or guaranteed in a multilayer NN. In addition, for guaranteed stability, the weight tuning using the delta rule at each layer must slow down as the NN becomes larger. This is a problem often noted in the literature (Mpitssos and Burton 1992).

By employing a projection algorithm, it is shown that the tuning rate can be made independent of the NN size. Modified tuning paradigms are finally proposed to make the NN robust so that PE is not needed.

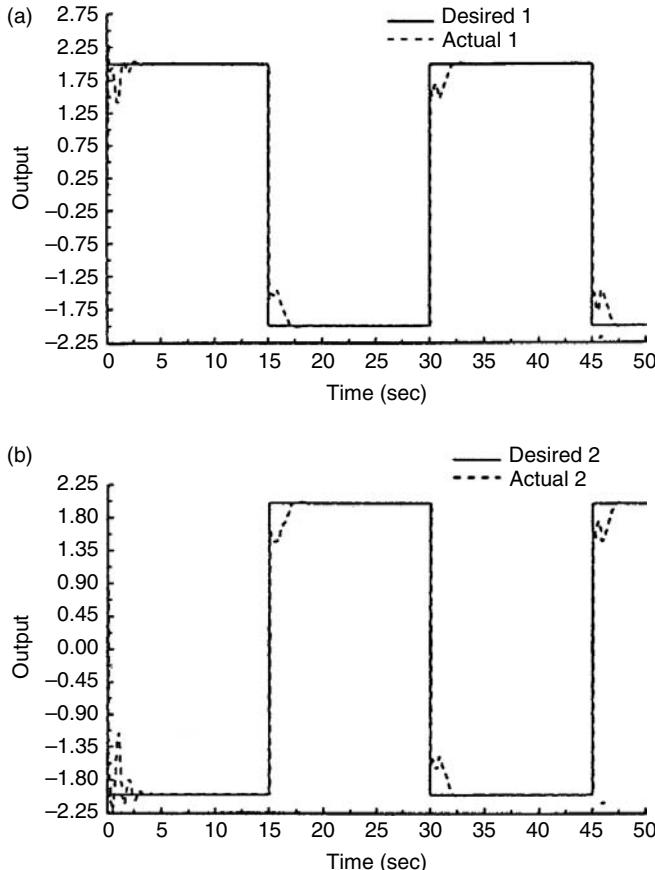


FIGURE 3.7 Response of the NN controller with improved weight-tuning and projection algorithm. (a) Desired and actual state 1. (b) Desired and actual state 2.

Assume that there exist some constant target weights W_1 , W_2 , and W_3 for a three-layer NN so that the nonlinear function in (3.3) can be written as

$$f(x) = W_3^T \varphi_3(W_2^T \varphi_2(W_1^T \varphi_1(x(k)))) + \varepsilon(k) \quad (3.53)$$

where $\|\varepsilon(k)\| < \varepsilon_N$, with the bounding constant known. Unless the network is minimal, the target weights may not be unique (Sontag 1992; Sussmann 1992). The best weights may then be defined as those which minimize the supremum norm over S of $\varepsilon(k)$. This issue is not a major concern here, as only the existence of such target (ideal) weights is required; their actual values

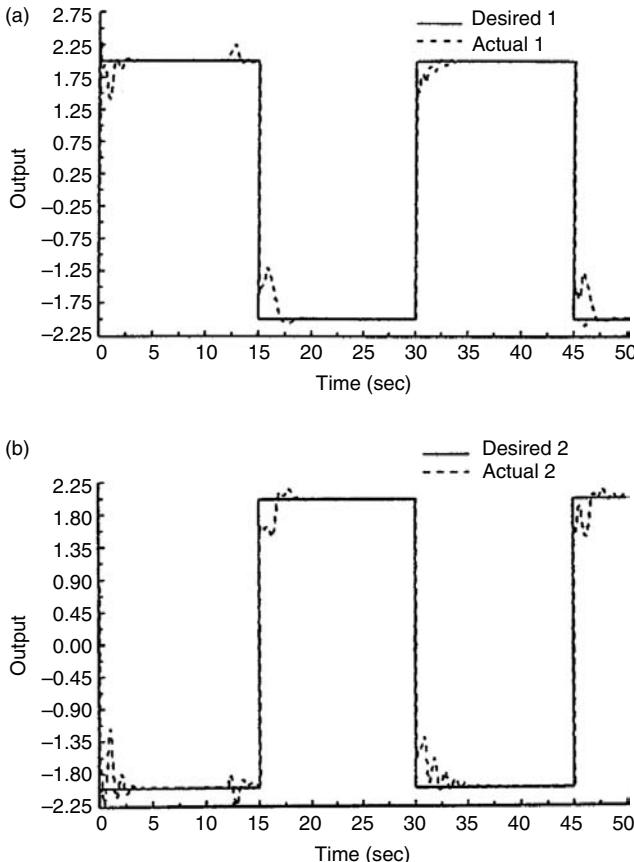


FIGURE 3.8 Response of the NN controller with improved weight-tuning in the presence of bounded disturbances. (a) Desired and actual state 1. (b) Desired and actual state 2.

are not required. This assumption is similar to Erzberger's assumptions in the linear-in-the-parameters adaptive control. The major difference is that, while the Erzberger's assumptions often do not hold, the approximation properties of NN guarantee that the target weights always exist if $f(x)$ is continuous over a compact set.

For notational convenience define the matrix of all the target weights as $Z = \text{diag}\{W_1, W_2, W_3\}$. Then bounding assumptions can be stated.

Assumption 3.1.1: The ideal weights are bounded by known positive values so that $\|W_1\| \leq W_{1\max}$, $\|W_2\| \leq W_{2\max}$, and $\|W_3\| \leq W_{3\max}$, or $\|Z\| \leq Z_{\max}$.

3.1.3.1 Error Dynamics and NN Controller Structure

Define the NN functional estimate by

$$\hat{f}(x) = \hat{W}_3^T \varphi_3(\hat{W}_2^T \varphi_2(\hat{W}_1^T \varphi_1(x(k)))) \quad (3.54)$$

where \hat{W}_1 , \hat{W}_2 , and \hat{W}_3 are the current value of the weights. The vector of input layer activation functions is given by $\hat{\phi}_1(k) = \phi_1(k) = \phi_1(x(k))$. Then the vector of activation functions of the hidden and output layer with the actual weights at the instant k is denoted by

$$\hat{\phi}_{m+1}(k) = \phi(\hat{W}_m^T \hat{\phi}_m(k)) \quad \forall m = 1, \dots, n-1 \quad (3.55)$$

Fact 3.1.2: For a given trajectory, the activation functions are bounded by known positive values so that

$$\|\hat{\phi}_1(k)\| \leq \phi_{1\max} \quad \|\hat{\phi}_2(k)\| \leq \phi_{2\max} \quad \|\hat{\phi}_3(k)\| \leq \phi_{3\max} \quad (3.56)$$

The error in the weights or weight estimation errors are given by

$$\begin{aligned} \tilde{W}_3(k) &= W_3 - \hat{W}_3(k) & \tilde{W}_2(k) &= W_2 - \hat{W}_2(k) \\ \tilde{W}_1(k) &= W_1 - \hat{W}_1(k) & \tilde{Z}(k) &= Z - \hat{Z}(k) \end{aligned} \quad (3.57)$$

where $\hat{Z}(k) = \text{diag}\{\hat{W}_1(k), \hat{W}_2(k), \hat{W}_3(k)\}$ and the hidden-layer output errors are defined as

$$\tilde{\phi}_2(k) = \phi_2 - \hat{\phi}_2(k) \quad \tilde{\phi}_3(k) = \phi_3 - \hat{\phi}_3(k) \quad (3.58)$$

Select the control input $u(k)$ by

$$\begin{aligned} u(k) &= \beta_0^{-1} [x_{nd}(k+1) - \hat{W}_3^T(k) \hat{\phi}_3(k) - \lambda_1 e_n(k) - \dots \\ &\quad - \lambda_{n-1} e_2(k) + k_v r(k)] \end{aligned} \quad (3.59)$$

where the functional estimate (3.54) is provided by a three-layer NN and denoted in (3.59) by $\hat{W}_3^T(k) \hat{\phi}_3(k)$. Then closed-loop filtered tracking error dynamics become

$$r(k+1) = k_v r(k) + \bar{e}_i(k) + W_3^T \tilde{\phi}_3(k) + \varepsilon(k) + d(k) \quad (3.60)$$

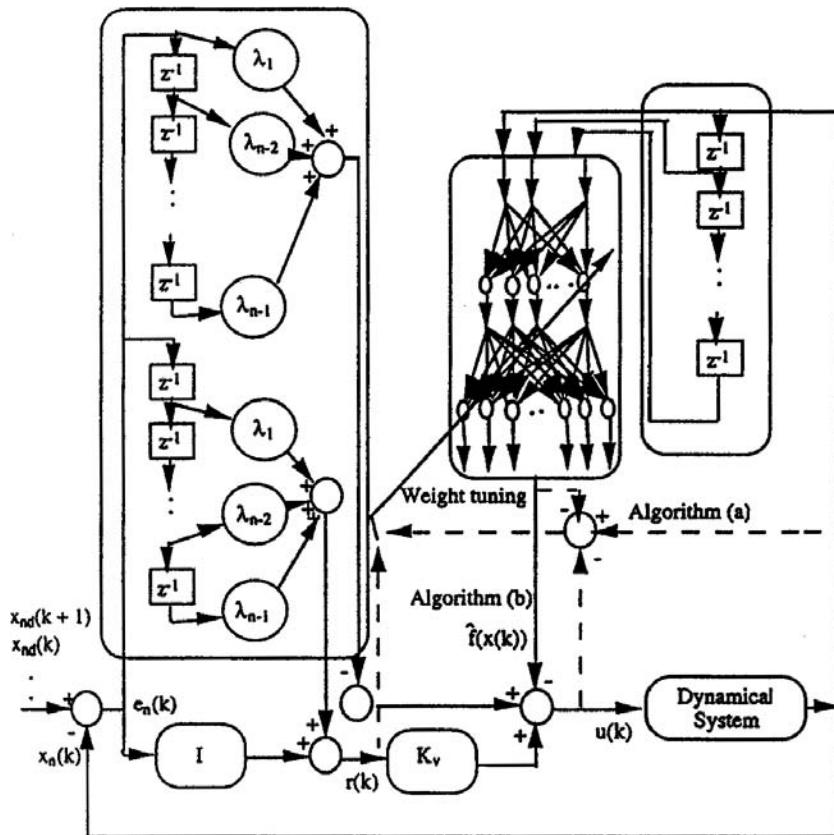


FIGURE 3.9 Multilayer NN controller structure.

where the identification error for a three-layer NN is defined by

$$\bar{e}_i(k) = \tilde{W}_3^T \hat{\phi}_3(k) \quad (3.61)$$

The proposed controller structure is shown in Figure 3.9. The output of the plant is processed through a series of delays to obtain the past values of the output and fed as inputs to the NN so that the nonlinear function in (3.3) can be suitably approximated. Note that neither the input $u(k)$ nor its past values are still needed by the NN. The next step is to determine the weight tuning schemes so that the tracking error of the closed-loop filtered error dynamics is guaranteed.

3.1.3.2 Multilayer NN Weight Updates

A family of NN weight-tuning paradigms that guarantee the stability of the closed-loop system (3.60) is presented in this section. It is required to demonstrate that the tracking error $r(k)$ is suitably small and that the NN weights \hat{W}_1 , \hat{W}_2 , and \hat{W}_3 remain bounded, for then the control $u(k)$ is bounded.

Here we consider the three-layer NN case. The two tuning algorithms in Table 3.2 are derived in the next theorem. One algorithm is based on a modified functional estimation error and the other based on the filtered tracking error. These algorithms require PE, which is defined for a multilayer NN during the proof (Jagannathan and Lewis 1996b).

Theorem 3.1.4 (Three-Layer NN Controller Requiring PE): Let the desired trajectory $x_{nd}(k)$ for (3.3) and the initial conditions be bounded in a compact set U . Let the NN functional reconstruction error and the disturbance bounds ε_N and d_M respectively be known constants. Consider the weight tuning provided for the input and hidden layers as

$$\hat{W}_1(k+1) = \hat{W}_1(k) - \alpha_1 \hat{\phi}_1(k)[\hat{y}_1(k) + B_1 k_v r(k)]^T \quad (3.62)$$

$$\hat{W}_2(k+1) = \hat{W}_2(k) - \alpha_2 \hat{\phi}_2(k)[\hat{y}_1(k) + B_2 k_v r(k)]^T \quad (3.63)$$

where $\hat{y}_i(k) = \hat{W}_i^T(k) \hat{\phi}_i(k)$, and

$$\|B_i\| \leq \kappa_i \quad i = 1, 2 \quad (3.64)$$

Take the weight tuning for the output layer as either

$$(a) \quad \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k) \bar{f}^T(k) \quad (3.65)$$

where $\bar{f}(k)$, is defined as the functional augmented error computed using

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}_3^T(k) \hat{\phi}_3(k) \quad (3.66)$$

or as

$$(b) \quad \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k) r^T(k+1) \quad (3.67)$$

where $\alpha_i > 0, \forall i = 1, 2, 3$ denoting constant learning rate parameter or adaptation gains.

Let the output vectors of the input, hidden, and output layers, $\hat{\phi}_1(k)$, $\hat{\phi}_2(k)$, and $\hat{\phi}_3(k)$ be persistently exciting. Then the filtered tracking

error $r(k)$ in (3.5) and the errors in weight estimates $\tilde{W}_1(k)$, $\tilde{W}_2(k)$, and $\tilde{W}_3(k)$ are UUB and the practical bounds given by (3.77) and (3.78) through (3.80) for the case of algorithm (a) and (3.77) and (3.82) through (3.84) for the case of algorithm (b), provided the following conditions hold:

$$(1) \quad \alpha_i \|\hat{\phi}_i(x(k))\|^2 < \begin{cases} 2, & \forall i = 1, 2 \\ 1, & \forall i = 3 \end{cases} \quad (3.68)$$

$$(2) \quad k_{v \max} < \frac{1}{\sqrt{\eta}}$$

where η is given for algorithm (a) as

$$\eta = 1 + \frac{1}{(1 - \alpha_3 \|\hat{\phi}_3(k)\|^2)} + \sum_{i=1}^2 \frac{\kappa_i^2}{(1 - \alpha_3 \|\hat{\phi}_i(k)\|^2)} \quad (3.69)$$

and for algorithm (b) as

$$\eta = \frac{1}{(1 - \alpha_3 \|\hat{\phi}_3(k)\|^2)} + \sum_{i=1}^2 \frac{\kappa_i^2}{(1 - \alpha_3 \|\hat{\phi}_i(k)\|^2)} \quad (3.70)$$

Proof: Algorithm (a): Define the Lyapunov function candidate

$$\begin{aligned} J = & r^T(k)r(k) + \frac{1}{\alpha_1} \text{tr}(\tilde{W}_1^T(k)\tilde{W}_1(k)) + \frac{1}{\alpha_2} \text{tr}(\tilde{W}_2^T(k)\tilde{W}_2(k)) \\ & + \frac{1}{\alpha_3} \text{tr}(\tilde{W}_3^T(k)\tilde{W}_3(k)) \end{aligned} \quad (3.71)$$

The first difference is given by

$$\begin{aligned} \Delta J = & r^T(k+1)r(k+1) - r^T(k)r(k) + \frac{1}{\alpha_1} \text{tr}(\tilde{W}_1^T(k+1)\tilde{W}_1(k+1) \\ & - \tilde{W}_1^T(k)\tilde{W}_1(k)) + \frac{1}{\alpha_2} \text{tr}(\tilde{W}_2^T(k+1)\tilde{W}_2(k+1) - \tilde{W}_2^T(k)\tilde{W}_2(k)) \\ & + \frac{1}{\alpha_3} \text{tr}(\tilde{W}_3^T(k+1)\tilde{W}_3(k+1) - \tilde{W}_3^T(k)\tilde{W}_3(k)) \end{aligned} \quad (3.72)$$

Substituting (3.60) and (3.62) to (3.65) in (3.72), collecting terms together and completing the square yields

$$\begin{aligned}
\Delta J \leq & r^T(k)[I - k_v^T k_v]r(k) + 2(k_v r(k))^T [W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) \\
& + d(k)] + (1 + \alpha_3 \hat{\varphi}_3^T \hat{\varphi}_3)[W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k)] + \frac{W_{1 \max}^2 \|\hat{\varphi}_1(k)\|^2}{(2 - \alpha_1 \|\hat{\varphi}_1(k)\|^2)} \\
& + \frac{W_{2 \max}^2 \|\hat{\varphi}_2(k)\|^2}{(2 - \alpha_2 \|\hat{\varphi}_2(k)\|^2)} - [1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k)] \\
& \times \left[\bar{e}_i(k) - \frac{k_v r(k) + \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k) (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k))}{(1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k))} \right]^T \\
& \times \left[\bar{e}_i(k) - \frac{k_v r(k) + \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k) (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k))}{(1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k))} \right] \\
& + \frac{1}{(1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k))} [k_v r(k) + \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k) (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) \\
& + d(k))]^T [k_v r(k) + \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k) (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k))] \\
& - (2 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k)) \left\| \hat{W}_1^T \hat{\varphi}_1(k) - \frac{(1 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k))}{(2 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k))} \right. \\
& \times (W_1^T \hat{\varphi}_1(k) + k_v r(k)) \left. \right\|^2 - (2 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k)) \\
& \times \left\| \hat{W}_2^T \hat{\varphi}_2(k) - \frac{(1 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k))}{(2 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k))} (W_2^T \hat{\varphi}_2(k) + k_v r(k)) \right\|^2 \\
& + 2k_{v \max} \|r(k)\| \sum_{i=1}^2 \frac{\kappa_i \|\varphi_i(k)\| W_{i \max}}{(2 - \alpha_i \hat{\varphi}_i^T(k) \varphi_i(k))} + k_{v \max}^2 \|r(k)\|^2 \\
& \times \sum_{i=1}^2 \frac{\kappa_i^2}{(2 - \alpha_i \hat{\varphi}_i^T(k) \varphi_i(k))} \tag{3.73}
\end{aligned}$$

Completing the squares for the last term in (3.73) results in

$$\begin{aligned}
\Delta J \leq & -(1 - \eta k_{v \max}^2) \left[\|r(k)\|^2 - 2 \frac{\gamma k_{v \max}}{(1 - \eta k_{v \max}^2)} \right. \\
& \left. - \frac{\rho}{(1 - \eta k_{v \max}^2)} \right] - [1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k)]
\end{aligned}$$

$$\begin{aligned}
& \times \left\| \bar{e}_i(k) - \frac{k_v r(k) + \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k) (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k))}{(1 - \alpha_3 \hat{\varphi}_3^T(k) \hat{\varphi}_3(k))} \right\|^2 \\
& - (2 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k)) \left\| \hat{W}_1^T \hat{\varphi}_1(k) - \frac{(1 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k))}{(2 - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k))} \right. \\
& \quad \times (W_1^T \hat{\varphi}_1(k) + k_v r(k)) \left. \right\|^2 - (2 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k)) \\
& \quad \times \left\| \hat{W}_2^T \hat{\varphi}_2(k) - \frac{(1 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k))}{(2 - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k))} (W_2^T \hat{\varphi}_2(k) + k_v r(k)) \right\|^2 \quad (3.74)
\end{aligned}$$

where η is given in (3.69) with $k_{v \max}$ the maximum singular value of k_v and

$$\gamma = \frac{1}{(1 - \alpha_3 \|\hat{\varphi}_3(k)\|^2)} [W_{3 \max} \tilde{\varphi}_{3 \max} + \varepsilon_N + d_M] + \sum_{i=1}^2 \frac{\kappa_i \|\hat{\varphi}_i(k)\| W_{i \max}}{(2 - \alpha_i \|\hat{\varphi}_i(k)\|^2)} \quad (3.75)$$

and

$$\rho = \left[\frac{(W_{3 \max} \|\tilde{\varphi}_3(k)\| + \varepsilon_N + d_M)^2}{(1 - \alpha_3 \|\hat{\varphi}_3(k)\|^2)} + \sum_{i=1}^2 \frac{\|\hat{\varphi}_i(k)\|^2 W_{i \max}^2}{(2 - \alpha_i \|\hat{\varphi}_i(k)\|^2)} \right] \quad (3.76)$$

Since $\varepsilon_N + d_M$ is a constant, $\Delta J \leq 0$ as long as

$$\|r(k)\| > \frac{1}{(1 - \eta k_{v \max}^2)} \left[\gamma k_{v \max} + \sqrt{\gamma^2 k_{v \max}^2 + \rho(1 - \eta k_{v \max}^2)} \right] \quad (3.77)$$

Note that $|\sum_{k=0}^{\infty} \Delta J(k)| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (3.68) holds. This demonstrates that the tracking error $r(k)$ is bounded for all $k \geq 0$ and it remains to show that the weight estimates $\hat{W}_1(k)$, $\hat{W}_2(k)$, and $\tilde{W}_3(k)$ are bounded.

The dynamics relative to error in weight estimates using (3.62), (3.63), and (3.65) are given by

$$\begin{aligned}\tilde{W}_1(k+1) &= [I - \alpha_1 \hat{\phi}_1^T(k) \varphi_1(k)] \tilde{W}_1(k) + \alpha_1 \hat{\phi}_1(k) \\ &\quad \times [W_1^T \hat{\phi}_1(k) + B_1 k_v r(k)]^T\end{aligned}\quad (3.78)$$

$$\begin{aligned}\tilde{W}_2(k+1) &= [I - \alpha_2 \hat{\phi}_2^T(k) \varphi_2(k)] \tilde{W}_2(k) + \alpha_2 \hat{\phi}_2(k) \\ &\quad \times [W_2^T \hat{\phi}_2(k) + B_2 k_v r(k)]^T\end{aligned}\quad (3.79)$$

$$\begin{aligned}\tilde{W}_3(k+1) &= [I - \alpha_3 \hat{\phi}_3^T(k) \varphi_3(k)] \tilde{W}_3(k) + \alpha_3 \hat{\phi}_3(k) \\ &\quad \times [W_3^T \hat{\phi}_3(k) + \varepsilon(k) + d(k)]^T\end{aligned}\quad (3.80)$$

where the functional reconstruction error $\varepsilon(k)$ and the disturbance $d(k)$ are considered to be bounded. Applying the PE condition (3.21) and using the tracking error bound (3.77) and Lemma 3.1.2 for the cases $\varphi(k) = \hat{\phi}_i(k); \forall i = 1, \dots, 3$, the boundedness of $\tilde{W}_1(k)$, $\tilde{W}_2(k)$, and $\tilde{W}_3(k)$ in (3.62) to (3.65) respectively, and hence $\hat{W}_1(k)$, $\hat{W}_2(k)$, and $\hat{W}_3(k)$ are assured.

Algorithm (b): Define a Lyapunov function candidate as in (3.70). Substituting (3.60), (3.62) to (3.63), and (3.67) in (3.72), collecting terms together and completing the squares yields

$$\begin{aligned}\Delta J &\leq -(1 - \eta k_{v \max}^2) \left[\|r(k)\|^2 - 2 \frac{\gamma k_{v \max}}{(1 - \eta k_{v \max}^2)} - \frac{\rho}{(1 - \eta k_{v \max}^2)} \right] \\ &\quad - [1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k)] \\ &\quad \times \left\| \bar{e}_i(k) - \frac{\alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k) (k_v r(k) + (W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k)))}{(1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k))} \right\|^2 \\ &\quad - (2 - \alpha_1 \hat{\phi}_1^T(k) \varphi_1(k)) \left\| \hat{W}_1^T \hat{\phi}_1(k) - \frac{(1 - \alpha_1 \hat{\phi}_1^T(k) \varphi_1(k))}{(2 - \alpha_1 \hat{\phi}_1^T(k) \varphi_1(k))} \right. \\ &\quad \left. \times W_1^T \hat{\phi}_1(k) + k_v r(k) \right\|^2 - (2 - \alpha_2 \hat{\phi}_2^T(k) \varphi_2(k)) \\ &\quad \times \left\| \hat{W}_2^T \hat{\phi}_2(k) - \frac{(1 - \alpha_2 \hat{\phi}_2^T(k) \varphi_2(k))}{(2 - \alpha_2 \hat{\phi}_2^T(k) \varphi_2(k))} W_2^T \hat{\phi}_2(k) + k_v r(k) \right\|^2\end{aligned}\quad (3.81)$$

where η is given in (3.70) and ρ is given in (3.76). $\Delta J \leq 0$ as long as (3.68) holds and this results in (3.77).

The dynamics relative to error in weight estimates using (3.62), (3.63), and (3.67) are given by

$$\tilde{W}_1(k+1) = [I - \alpha_1 \hat{\varphi}_1^T(k) \varphi_1(k)] \tilde{W}_1(k) + \alpha_1 \hat{\varphi}_1(k) [W_1^T \hat{\varphi}_1(k) + B_1 k_v r(k)]^T \quad (3.82)$$

$$\tilde{W}_2(k+1) = [I - \alpha_2 \hat{\varphi}_2^T(k) \varphi_2(k)] \tilde{W}_2(k) + \alpha_2 \hat{\varphi}_2(k) [W_2^T \hat{\varphi}_2(k) + B_2 k_v r(k)]^T \quad (3.83)$$

$$\begin{aligned} \tilde{W}_3(k+1) = & [I - \alpha_3 \hat{\varphi}_3^T(k) \varphi_3(k)] \tilde{W}_3(k) + \alpha_3 \hat{\varphi}_3(k) \\ & \times [k_v r(k) + W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k)]^T \end{aligned} \quad (3.84)$$

where the tracking error $r(k)$, functional reconstruction error $\varepsilon(k)$, and the disturbance $d(k)$ are considered to be bounded. Applying the PE condition (3.21) and using the tracking error bound (3.77), and Lemma 3.1.2 for the cases $\varphi(k) = \hat{\varphi}_i(k); \forall i = 1, \dots, 3$, the boundedness of $\tilde{W}_1(k)$, $\tilde{W}_2(k)$, and $\tilde{W}_3(k)$ in (3.62) through (3.67), respectively, and hence $\hat{W}_1(k)$, $\hat{W}_2(k)$, and $\hat{W}_3(k)$ are assured.

The right-hand sides of Equation 3.77 and Equations 3.78 to 3.80 for the case of algorithm (a) or (3.77) and (3.82) to (3.84) for the case of algorithm (b) may be taken as practical bounds, both on the norms of the error $r(k)$ and the weight errors $\tilde{W}_1(k)$, $\tilde{W}_2(k)$, and $\tilde{W}_3(k)$. Since the target values are bounded, it follows that the NN weights, $\hat{W}_1(k)$, $\hat{W}_2(k)$, and $\hat{W}_3(k)$ provided by the tuning algorithms are bounded. Hence the control input is bounded.

One of the drawbacks of the available methodologies that guarantee the tracking and bounded weights (Chen and Khalil 1995; Lewis et al. 1995) is the lack of generalization of stability analysis to NN having an arbitrary number of hidden layers. The reason is partly due to the problem of defining and verifying the PE for a multilayered NN. For instance, in the case of a three-layered continuous-time NN (Lewis et al. 1995), the PE conditions are not easy to derive as one is faced with the observability properties of a bilinear system. According to the proof presented above, however, the PE for a multilayered NN is defined as the PE (in the sense of definition) of all the hidden-layer inputs $\hat{\varphi}_i(k); \forall i = 1, \dots, n$. The three-layer stability analysis given in Theorem 3.1.3 and Theorem 3.1.4 can be extended to n -layer NN. The n -layer stability analysis is presented in Jagannathan and Lewis (1996b).

Remarks:

1. It is important to note that in this theorem also there is no CE assumption for the NN controller, in contrast to standard work in

discrete-time adaptive control (Astrom and Wittenmark 1989). In the proof, the Lyapunov function shown in Appendix 2.A of this chapter is of the form

$$J = r^T(k)r(k) + \sum_{i=1}^3 \frac{1}{\alpha_i} \text{tr}[\tilde{W}_i^T(k)\tilde{W}_i(k)]$$

which weights the tracking errors, $r(k)$ and the weight estimation errors for the controller $\tilde{W}_i(k); \forall i = 1, 2, 3$. The proof is exceedingly complex due to the presence of several different variables resulting from multiple NN layers. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

2. The weight updating rules (3.62), (3.63), (3.65), and (3.67) are non-standard schemes that were derived from Lyapunov analysis and do not include an extra term which is normally used to provide robustness due to the coupling in the proof between the tracking and weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required if the PE condition is applied.
3. Condition (3.68) can be checked easily. The maximum singular value of the controller gain $k_{v\max}$ and the NN learning rate parameter have to satisfy (3.68) in order for the closed-loop system to be stable. This is a unique relationship between the controller gain and the NN learning rate. This condition states that even for three-layer NNs, for faster tuning of weights, the closed-loop poles should be inside the unit disc. On the other hand, such constraints do not exist for controller design parameters and NN learning rate parameter in continuous-time. Therefore, the design parameters for the adaptive NN controllers in continuous-time are selected arbitrarily.
4. It is important to note that the problem of initializing the network weights (referred to as symmetric breaking (Rumelhart et al. 1990) occurring in other techniques in the literature does not arise even though the NN weights are tuned online with no explicit off-line learning phase. This is because when $\hat{Z}(0)$ is taken as zero the standard PD controller $k_{vr}(k)$ stabilizes the plant on an interim basis as seen in certain restricted classes of nonlinear systems such as robotic systems. Thus the NN controller requires no off-line learning phase.

Example 3.1.5 (Nonlinear System Using Multilayer NN): Consider the nonlinear system described by Example 3.1.1 with the parameters for the nonlinear system are selected as $a_1 = a_2 = 1, b_1 = b_2 = 1$. Desired sinusoidal, $\sin(2\pi t)/25$ and cosine inputs, $\cos(2\pi t)/25$ were preselected for the axis 1 and 2, respectively. The gains of the PD controller in continuous-time were chosen as $k_v = \text{diag}(20, 20)$ with $\Lambda = \text{diag}(5, 5)$ and a sampling interval of 10 msec was considered. A three-layer NN was selected with four input neurons, six hidden and two output neurons. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for X_1 were chosen to be $[0.5, 0.1]^T$ and the weights were initialized to zero. No off-line learning is performed to train the networks.

The elements of the $B_i, \forall i = 1, 2$ are chosen to be 0.1. Figure 3.10 presents the tracking response of the NN controller with $\alpha_1 = 0.1, \alpha_2 = 0.01$, and $\alpha_3 = 0.1$ using (3.62), (3.63), and (3.67). From the figure, it can be seen that the tracking response is impressive.

3.1.3.3 Projection Algorithm

The adaptation gains for a three-layer NN and an n -layer NN $\alpha_i > 0, \forall i = 1, 2, \dots, n$, are constant parameters in the update laws presented in (3.62) through (3.67). These update laws correspond to the delta rule (Rumelhart et al. 1990; Sadegh 1993). The theorem reveals that update tuning mechanisms employing the delta rule have a major drawback. In fact using (3.68), an upper bound on the adaptation gain can be obtained as

$$\begin{aligned} \alpha_i &< \frac{2}{\|\hat{\phi}_i(x(k))\|^2} \quad i = 1, \dots, n-1 \\ &< \frac{1}{\|\hat{\phi}_i(x(k))\|^2} \quad i = n \end{aligned} \quad (3.85)$$

Since $\hat{\phi}_i(x(k)) \in \Re^{N_p}$, with N_p the number of hidden-layer neurons in the i th layer. It is evident that the upper bound on the adaptation gain at each layer depends upon the number of hidden-layer neurons present in that layer. Specifically, if there are N_p hidden-layer neurons and if the maximum value of each hidden-node output at the i th layer is taken as unity (as for the sigmoid), then the bound on the adaptation gain in order to assure stability of the closed-loop

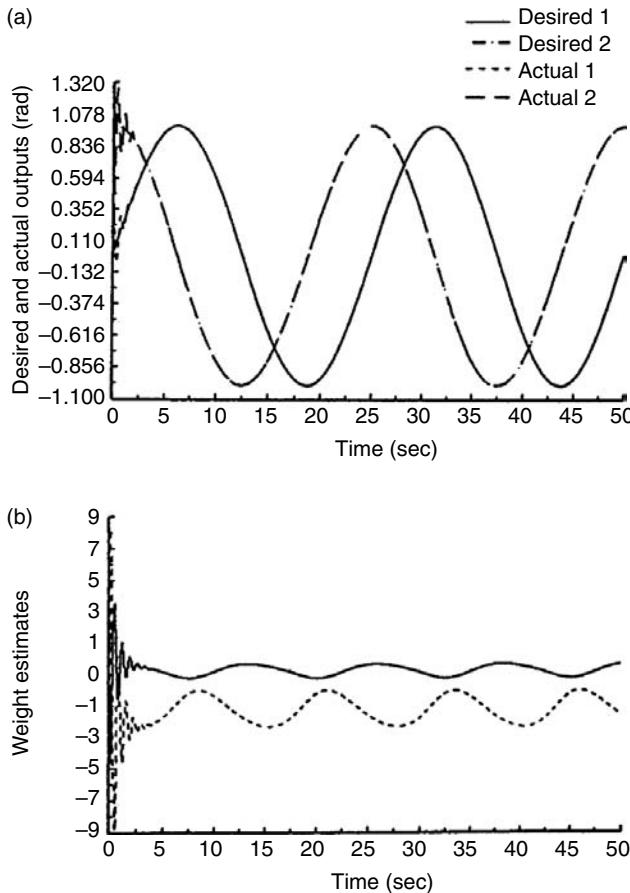


FIGURE 3.10 Response of multilayer NN controller with delta rule weight tuning and small α_3 . (a) Desired and actual trajectory. (b) Representative weight estimates.

system is given by

$$\begin{aligned}
 0 < \alpha_i &< \frac{2}{N_p} \quad \forall i = 1, 2, \dots, n-1 \\
 0 < \alpha_i &< \frac{1}{N_p} \quad i = n
 \end{aligned} \tag{3.86}$$

In other words, the upper bound on the adaptation gain for the case of delta-rule-based tuning decreases with an increase in number of nodes

in that layer, so that adaptation must slow down owing to the addition of more hidden-layer neurons for guaranteed performance similar to the case of one-layer NN.

This major drawback can be easily overcome by modifying the update law in order to obtain a projection algorithm (Goodwin and Sin 1984; Åström and Wittenmark 1989). Replace the constant adaptation gain by

$$\alpha_i = \frac{\xi_i}{\xi_i + \|\hat{\phi}_i(k)\|^2} \quad i = 1, 2, \dots, n \quad (3.87)$$

where

$$\xi_i > 0 \quad i = 1, \dots, n$$

and

$$0 < \xi_i < 2 \quad i = 1, \dots, n - 1$$

$$0 < \xi_i < 1, n$$

are constants. Note that $\xi_i, i = 1, 2, \dots, n$ is now the new adaptation gain at each layer and it is always true that

$$\begin{aligned} \frac{\xi_i}{\xi_i + \|\hat{\phi}_i(k)\|^2} \|\hat{\phi}_i(k)\|^2 &< 2 \quad i = 1, \dots, n - 1 \\ &< 1 \quad i = n \end{aligned} \quad (3.88)$$

hence guaranteeing (3.87) for every N_p at each layer.

Example 3.1.6 (Multiple NN Size and Adaptation Gains): Consider Example 3.1.1 and note that the upper bound on the allowed adaptation gains α_1, α_2 , and α_3 using (3.85) for the case of delta rule at each layer is computed to be 0.5, 0.32, 0.5. The adaptation gains for the multilayer NN weight tuning are selected as $\xi_1 = \xi_2 = 1.0$, and $\xi_3 = 0.7$, and $\zeta_1 = \zeta_2 = \zeta_3 = 0.01$ for the case of projection algorithm with (3.62) through (3.63) with (3.67). Figure 3.11 presents the tracking response of the controller with projection algorithm. It is clear from Figure 3.10 that the controller using the delta rule at each layer performs equally well with the projection algorithm (see Figure 3.10) when the adaptation gain is small so that (3.85) is satisfied.

Large weight values are needed initially not only for the delta rule with small (shown in Figure 3.10b but also for the projection algorithm with large

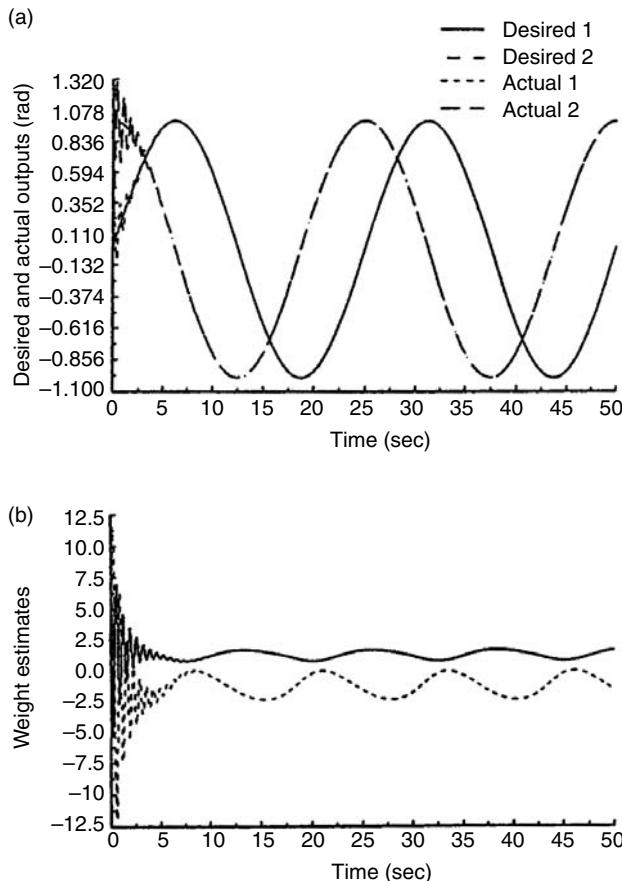


FIGURE 3.11 Response of multilayer NN controller with delta-rule weight-tuning and projection algorithm with large α_3 . (a) Desired and actual trajectory. (b) Representative weight estimates.

adaptation gain (see Figure 3.11b). Note from Figure 3.11 due to the large adaptation gains for the case of the projection algorithm, overshoots and undershoots are observed in the initial stages even though the tracking performance is extremely impressive.

Figure 3.12 illustrates the response of the NN controller when the delta rule is employed with the adaptation gain α_3 in the last layer is changed from 0.1 to 0.51. From Figure 3.12, it is evident that the weight tuning using the delta rule at each layer becomes unstable at $t = kT = 1.08$ sec, where T is the

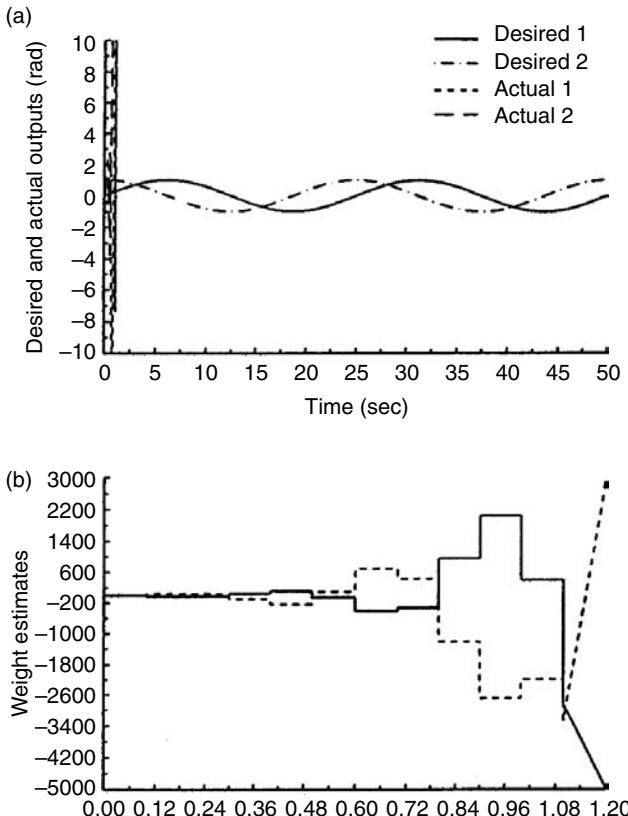


FIGURE 3.12 Response of multilayer NN controller with delta-rule weight-tuning with large α_3 . (a) Desired and actual trajectory. (b) Representative weight estimates.

sampling interval. Note that the representative weight estimates as illustrated in Figure 3.12b of the NN are unbounded in this case. This demonstrates that the adaptation gain in the case of delta rule at each layer must decrease with an increase in the hidden-layer neurons. In fact, the theoretical limit implied by (3.68) in this case for this sampling interval is such that this bound appears to be a tight bound in general.

The performance of the NN controller was investigated while varying the adaptation gains at the output layer for the case of the projection algorithm. Figure 3.13a and b show the tracking response and some NN representative weight estimates of the NN controller with $\xi_3 = 0.1$. As expected, the overshoots and undershoots have been totally eliminated but there appears to

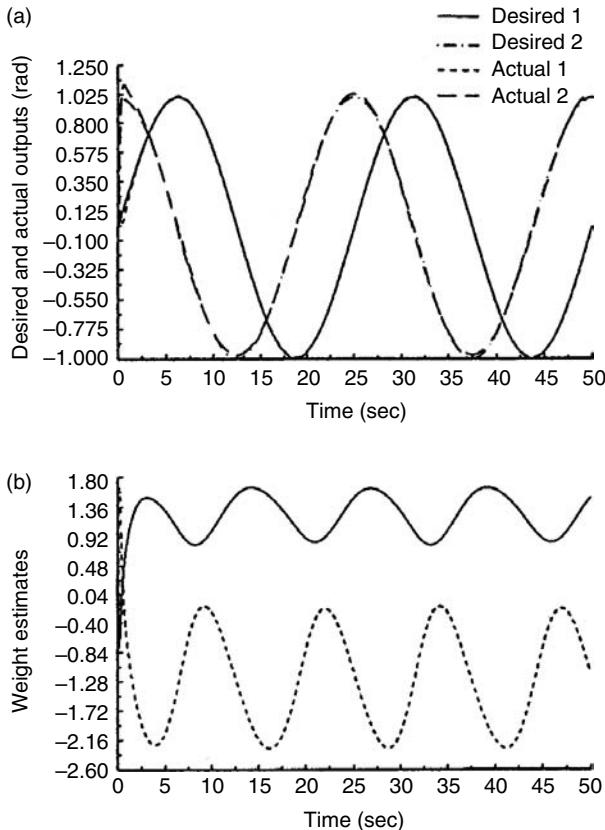


FIGURE 3.13 Response of multilayer NN controller with delta-rule weight-tuning and projection algorithm with small α_3 . (a) Desired and actual trajectory. (b) Representative weight estimates.

be a slight degradation in performance. In other words, at very low adaptation gains, overshoots and undershoots are not seen but there appears a slight degradation in the tracking performance with a slow and smooth convergence. On the other hand, at large adaptation gains, overshoots and undershoots are observed with a good tracking performance. As the adaptation gains are further increased, the oscillatory behavior continues to increase and finally the system becomes unstable. In other words, from the bounds presented in (3.68), as the adaptation gains are increased, the margin of stability continues to decrease and at large adaptation gains (close to one for instance at the last layer) the system becomes unstable. Thus the simulation results conducted corroborate with the bounds presented in the previous sections.

3.1.3.4 Multilayer NN Weight-Tuning Modification for Relaxation of PE Condition

An approach similar to ε -modification is derived for one-layer (i.e., linear) discrete-time NN (Jagannathan and Lewis 1996b). In this section, the modified weight tuning algorithms discussed for a one-layer discrete-time NN in Section 3.1.2.6 are extended to a multilayer discrete-time NN so that the PE condition is not needed. The results of this section present tuning algorithms that overcome the need for PE in the case of multilayered NN. Here a three-layer NN is considered, but in Jagannathan and Lewis (1996b), the three-layer NN case is extended to a multilayer NN with n -layers.

The next theorem proves the stability of the NN controllers in Table 3.4, which have tuning algorithms augmented to avoid PE.

Theorem 3.1.5 (Three-Layer NN Controller Not Requiring PE): Assume the hypothesis presented in Theorem 3.1.4. Consider the weight tuning provided for the input and hidden layers as

$$\begin{aligned}\hat{W}_1(k+1) &= \hat{W}_1(k) - \alpha_1 \hat{\phi}_1(k)[\hat{y}_1(k) + B_1 k_v r(k)]^T \\ &\quad - \Gamma \|I - \alpha_1 \hat{\phi}_1(k) \hat{\phi}_1^T(k)\| \hat{W}_1(k)\end{aligned}\quad (3.89)$$

$$\begin{aligned}\hat{W}_2(k+1) &= \hat{W}_2(k) - \alpha_2 \hat{\phi}_2(k)[\hat{y}_2(k) + B_2 k_v r(k)]^T \\ &\quad - \Gamma \|I - \alpha_2 \hat{\phi}_2(k) \hat{\phi}_2^T(k)\| \hat{W}_2(k)\end{aligned}\quad (3.90)$$

Take the weight tuning for the output layer as either

$$(a) \quad \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k) \bar{f}^T(k) - \Gamma \|I - \alpha_3 \hat{\phi}_3(k) \hat{\phi}_3^T(k)\| \hat{W}_3(k) \quad (3.91)$$

or as

$$(b) \quad \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k) r^T(k+1) \\ - \Gamma \|I - \alpha_3 \hat{\phi}_3(k) \hat{\phi}_3^T(k)\| \hat{W}_3(k) \quad (3.92)$$

with $\Gamma > 0$ a design parameter. Then the filtered tracking error $r(k)$ in (3.5) and the errors in weight estimates $\tilde{W}_1(k)$, $\tilde{W}_2(k)$, and $\tilde{W}_3(k)$ are UUB provided

the following conditions hold:

- $$(1) \quad \alpha_i \|\hat{\phi}_i(x(k))\|^2 < \begin{cases} 2, & \forall i = 1, 2 \\ 1, & \forall i = 3 \end{cases}$$
- $$(2) \quad k_{v \max} < \frac{1}{\sqrt{\bar{\sigma}}}$$
- $$(3) \quad 0 < \Gamma < 1$$
- (3.93)

where $\bar{\sigma}$ is given by

$$\bar{\sigma} = \beta_3 + \sum_{i=1}^2 \beta_i \kappa_i^2$$
(3.94)

with

$$\beta_i = \alpha_i \|\hat{\phi}_i(k)\|^2 + \frac{[(1 - \alpha_i \|\hat{\phi}_i(k)\|^2) - \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|]^2}{(2 - \alpha_i \|\hat{\phi}_i(k)\|^2)}$$
(3.95)

For algorithm (a), β_3 is given by

$$\beta_3 = 1 + \frac{1}{(1 - \alpha_3 \|\hat{\phi}_3(k)\|^2)}$$
(3.96)

whereas for algorithm (b) it is given by

$$\begin{aligned} \beta_3 = 1 + \alpha_3 \|\hat{\phi}_3(k)\|^2 + \frac{1}{(1 - \alpha_3 \|\hat{\phi}_3(k)\|^2)} [\alpha_3 \|\hat{\phi}_3(k)\|^2 \\ + \Gamma (1 - \alpha_3 \|\hat{\phi}_3(k)\|^2)]^2 \end{aligned}$$
(3.97)

Note: The parameters $\beta_i, \alpha_i, \forall i = 1, 2, 3$ and $\bar{\sigma}$ are dependent upon the trajectory.

The proofs and bounds on tracking error and NN weights are similar to those in the other theorems of this chapter.

Proof: See Jagannathan and Lewis (1996b).

Example 3.1.7 (Improved Weight-Tuning and Projection Algorithm): In the case of the projection algorithm (3.62), (3.63), and (3.67), with (3.87), NN weights in Figure 3.12 appear to be bounded, though in general they cannot

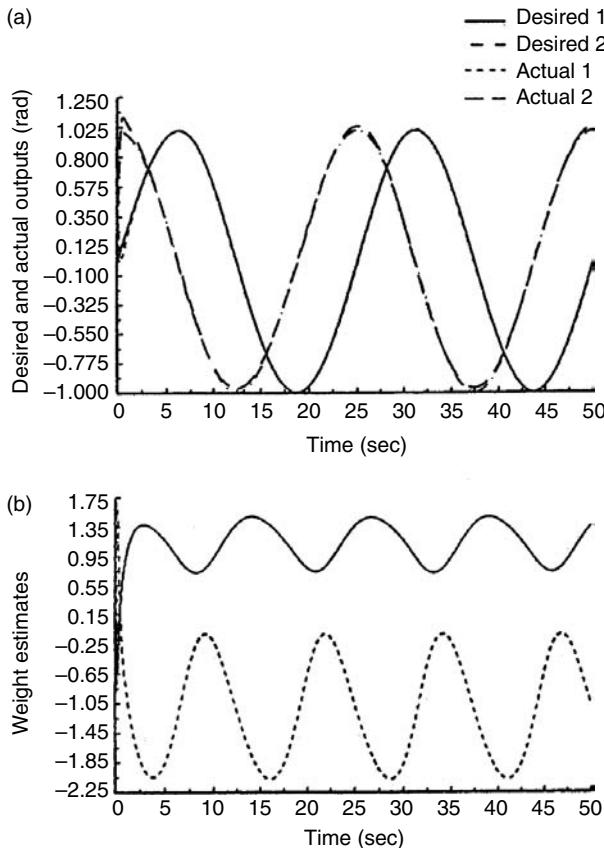


FIGURE 3.14 Response of multilayer NN controller with improved weight-tuning and projection algorithm with small α_3 . (a) Desired and actual trajectory. (b) Representative weight estimates.

be guaranteed without the PE condition. Therefore, the response of the controller with the improved weight tuning (3.89) and (3.90) with (3.87) is shown in Figure 3.14.

The design parameter Γ is selected to be 0.01. Note that with the improved weight tuning, not only is the tracking performance improved, for instance in axis 2, but also the weights remain bounded without the necessity of the PE condition. Finally, in all cases no initial NN training or learning phase was needed. In addition, the dynamics of the nonlinear system were not required to implement the NN controller as opposed to conventional adaptive control.

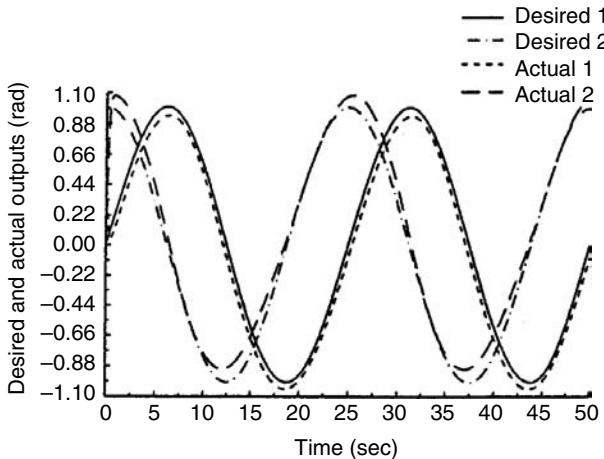


FIGURE 3.15 Response of PD controller.

To study the contribution of the NN, Figure 3.15 shows the response of the PD controller with no neural net. From Figure 3.15, it is clear that the addition of the NN makes a significant improvement in the tracking performance.

Example 3.1.8 (Discrete-Time System Control with Improved Weight Tuning): Consider Example 3.1.4 and the objective is to track a periodic step input of magnitude two units with a period of 30 sec. The elements of the diagonal matrix were chosen as $k_y = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ and a sampling interval of 10 msec was considered. A three-layer NN was selected with two input neurons, six hidden, and two output nodes. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for the plant were chosen to be $[1, -1]^T$. The weights were initialized to zero with an initial threshold value of 3.0. The design parameter Γ is selected to be 0.01. All the elements of the design parameter matrix $B_i; i = 1, 2$, are taken to be 0.1. No learning is performed initially to train the networks. The design parameters for the projection algorithm (3.87) were selected to be $\xi_1 = \xi_2 = 1.0$, and $\xi_3 = 0.7$ with $\zeta_1 = \zeta_2 = \zeta_3 = 0.001$.

It is also found (not shown) in this example as well that the delta-rule-based weight tuning performs very well when the learning rate is small. In addition, it was also observed during simulation studies that the learning rate for

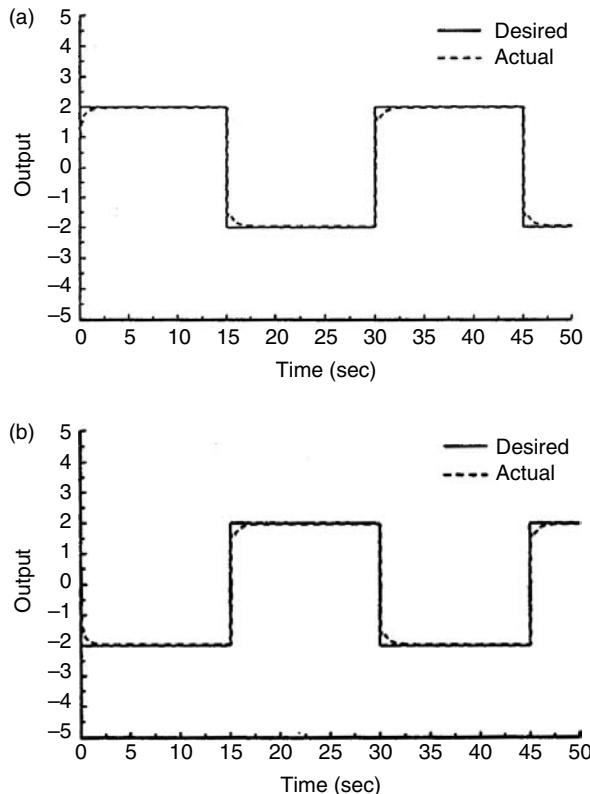


FIGURE 3.16 Response of multilayer NN controller with improved weight-tuning and projection algorithm. (a) Desired and actual state 1. (b) Desired and actual 1.

delta-rule-based weight tuning should decrease with an increase in the number of hidden-layer neurons. As expected, however, this problem is solved by employing a projection algorithm. In this example, only results using the improved weight tuning are presented. The response of the controller with the improved weight tuning (3.89) through (3.90) with (3.92) is shown in Figure 3.16. Note from Figure 3.16, as expected, the performance of the controller is extremely impressive.

Let us consider the case when a bounded disturbance given by

$$w(k) = \begin{cases} 0.0, & 0 \leq kT_m < 12 \\ 0.5, & kT_m \geq 12 \end{cases} \quad (3.98)$$

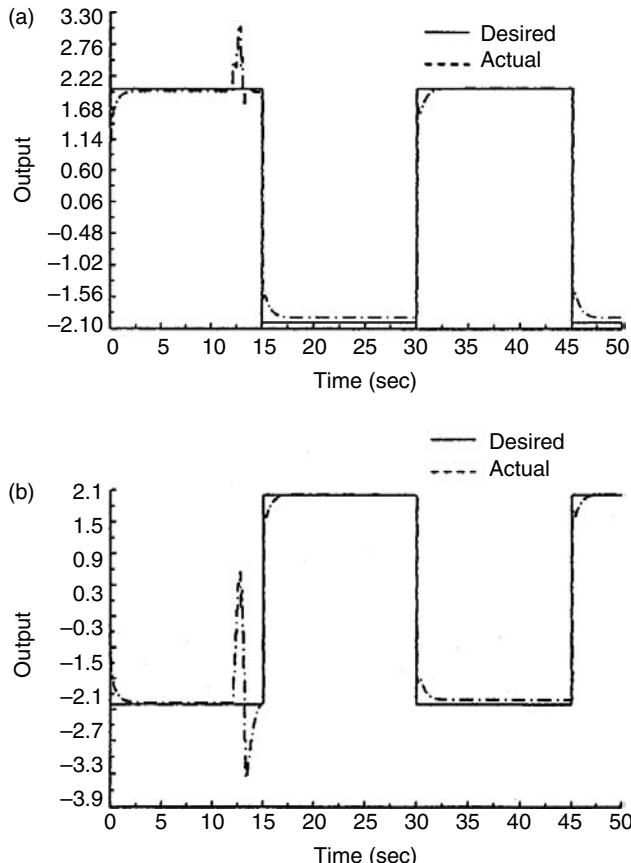


FIGURE 3.17 Response of multilayer NN controller with improved weight-tuning and projection algorithm with large α_3 . (a) Desired and actual state 1. (b) Desired and actual state 2.

is acting on the plant at the time instant k . Figure 3.17 presents the tracking response of the NN controllers with the improved weight-tuning and projection algorithm. The magnitude of the disturbance can be increased but the value should be bounded. The value shown in (3.98) is employed for simulation purposes only. It can be seen from the figure that the bounded disturbance induces bounded tracking errors at the output of the plant. From the results, it can be inferred that the bounds presented and the theoretical claims were justified through simulation studies both in continuous- and discrete-time.

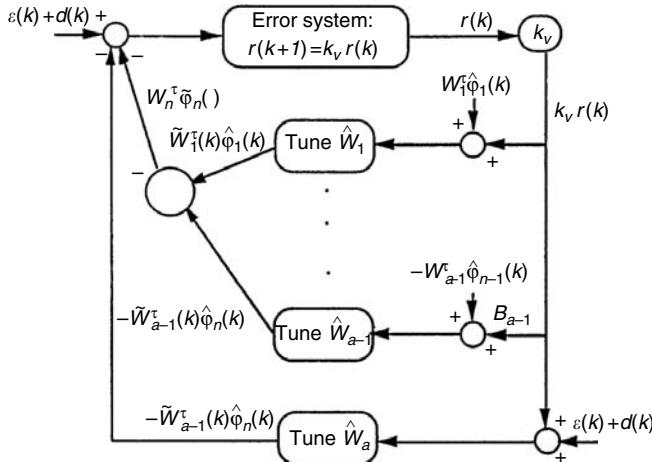


FIGURE 3.18 NN closed-loop system using an n -layer NN.

3.1.4 PASSIVITY OF THE NN

In this section, an interesting property of the NN controller is shown. Namely, the NN controller makes the closed-loop system passive. The practical importance of this is that additional unknown bounded disturbances do not destroy the stability and tracking of the system. Passivity is discussed in Chapter 2 and for discrete-time adaptive control, in Section 2.3.2. Note that the NN used in the controllers in this chapter are feedforward NN with no dynamics. However, tuning them online turns them into dynamical systems so that passivity properties can be defined.

The closed-loop error system (3.19) can be redrawn as shown in Figure 3.18 using an n -layer NN, now in the standard feedback configuration as opposed to the NN in the previous section. Passivity is essential in a closed-loop system since it guarantees the boundedness of signals and hence suitable performance, even in the presence of additional unforeseen bounded disturbances (i.e., NN robustness). Therefore in this section, passivity properties of the one-layer and multilayer NN and consequently of the closed-loop system are guaranteed through the NN weight-tuning algorithms.

3.1.4.1 Passivity Properties of the Tracking Error System

In general, the closed-loop tracking error system (3.19) can also be expressed as

$$r(k+1) = k_v r(k) + \xi_0(k) \quad (3.99)$$

with

$$\xi_0(k) = \tilde{f}(x) + d(k) \quad (3.100)$$

The following dissipative result holds for this system.

Theorem 3.1.6 (Passivity of the Closed-Loop Tracking System): The closed-loop tracking error system (3.99) is state strict passive from $\xi_0(k)$ to $k_v r(k)$ provided that

$$k_v^T k_v < 1 \quad (3.101)$$

Proof: Select a Lyapunov function candidate

$$J = r^T(k) r(k) \quad (3.102)$$

The first difference is given by

$$\Delta J = r^T(k+1) r(k+1) - r^T(k) r(k) \quad (3.103)$$

Substituting (3.99) into (3.103) yields

$$\Delta J = -r^T(k)(I - k_v^T k_v)r(k) + 2r^T(k)k_v\xi_0 + \xi_0^T(k)\xi_0(k) \quad (3.104)$$

Note that (3.104) is in power form defined in (2.33) with $g(k)$ a quadratic function of the state $r(k)$. Hence (3.99) is a state strict passive system.

Even though the closed-loop error system (3.99) is state strict passive, the overall system is not passive unless the weight-update laws guarantee the passivity of the lower block. It is usually difficult to demonstrate that the errors in weight updates are passive. However, in the following theorem, it is shown that the delta-rule-based tuning algorithms (3.22) and (3.24) for a one-layer NN yield a passive network.

3.1.4.2 Passivity Properties of One-Layer NN

It is shown here that the one-layer NN tuning algorithms in Theorem 3.1.1, where PE is required make the NN passive, but the tuning algorithms in

TABLE 3.3
Discrete-Time Controller Using Three-Layer NN: PE Required

Select the control input $u(k)$

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{W}_3^T(k)\hat{\phi}_3(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) + k_v r(k)]$$

Consider the weight tuning provided for the

Input and hidden layers:

$$\hat{W}_1(k+1) = \hat{W}_1(k) - \alpha_1 \hat{\phi}_1(k)[\hat{y}_1(k) + B_1 k_v r(k)]^T$$

$$\hat{W}_2(k+1) = \hat{W}_2(k) - \alpha_2 \hat{\phi}_2(k)[\hat{y}_1(k) + B_2 k_v r(k)]^T$$

where $\hat{y}_i(k) = \hat{W}_i^T(k)\hat{\phi}_i(k)$, and

$$\|B_i\| \leq \kappa_i, \quad i = 1, 2$$

Output layer: Tuning for the output layer is provided by either

$$(a) \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k)\bar{f}^T(k)$$

where $\bar{f}(k)$, is defined as the functional augmented error computed using

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}_3^T(k)\hat{\phi}_3(k)$$

or as

$$(b) \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k)r^T(k+1)$$

where $\alpha_i > 0, \forall i = 1, 2, 3$ denoting constant learning rate parameters or adaptation gains.

Theorem 3.1.3, where PE is not required make the NN state strict passive (SSP). The implications for the closed-loop passivity using the NN controller in Table 3.2 and Table 3.3 are then discussed.

Theorem 3.1.7 (One-Layer NN Passivity of Tuning Algorithms with PE): The weight tuning algorithms (3.22) to (3.24) make the map from $\varepsilon(k+d(k))$ for algorithm (a) and $k_v r(k) + \varepsilon(k) + d(k)$ for the case of algorithm (b) to $-\tilde{W}^T(k)\phi(k)$ a passive map.

Proof: Algorithm (a). Define a Lyapunov function candidate

$$J = \frac{1}{\alpha} \text{tr}[\tilde{W}^T(k)\tilde{W}(k)] \quad (3.105)$$

whose first difference is given by

$$\Delta J = \frac{1}{\alpha} \text{tr}[\tilde{W}^T(k+1)\tilde{W}(k+1) - \tilde{W}^T(k)\tilde{W}(k)] \quad (3.106)$$

Substituting the weight-update law (3.22) into (3.106) yields

$$\begin{aligned} \Delta J &= \frac{1}{\alpha} \text{tr}\{\tilde{W}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T[I - \alpha\phi(k)\phi^T(k)]\tilde{W}(k) - \tilde{W}^T(k)\tilde{W}(k) \\ &\quad - 2\alpha\tilde{W}^T(k)[I - \alpha\phi(k)\phi^T(k)]^T\phi(k)[\varepsilon(k) + d(k)]^T \\ &\quad + \alpha^2[\varepsilon(k) + d(k)]\phi^T(k)\phi(k) \times [\varepsilon(k) + d(k)]^T\} \\ &= \frac{1}{\alpha} \text{tr}\{-\alpha[2 - \alpha\phi^T(k)\phi(k)] \times [-\tilde{W}^T(k)\phi(k)][-\tilde{W}^T(k)\phi(k)]^T \\ &\quad + 2\alpha[1 - \alpha\phi(k)\phi^T(k)][-\tilde{W}^T(k)\phi(x(k))] \times [\varepsilon(k) + d(k)]^T \\ &\quad + \alpha^2[\phi^T(x(k))\phi(x(k))][\varepsilon(k) + d(k)][\varepsilon(k) + d(k)]^T\} \end{aligned} \quad (3.107)$$

Note that (3.107) is in power form (2.33) as long as the condition (3.25) holds. This in turn guarantees the passivity of the weight-tuning mechanism (3.22).

Algorithm (b). Select the Lyapunov function (3.105). Use (3.24) in (3.106) to obtain

$$\begin{aligned} \Delta J &= \{-[2 - \alpha\phi^T(k)\phi(k)][-\tilde{W}^T(k)\phi(k)][-\tilde{W}^T(k)\phi(k)]^T \\ &\quad + 2[1 - \alpha\phi(k)\phi^T(k)][-\tilde{W}^T(k)\phi(k)]^T[k_v r(k) + \varepsilon(k) + d(k)] \\ &\quad + \alpha\phi^T(k)\phi(k)[k_v r(k) + \varepsilon(k) + d(k)]^T[k_v r(k) + \varepsilon(k) + d(k)]\} \end{aligned} \quad (3.108)$$

which is in power form (2.33) for discrete-time systems as long as the condition (3.25) holds. Here, only the passivity of the error system (3.22) is given.

Thus, the parameter error block is passive and the closed-loop tracking error system (3.19) is dissipative; this guarantees the dissipativity of the closed-loop system (Landau 1979). By employing the passivity theorem (Landau 1979), one can conclude that the input/output signals of each block are bounded as long as the external inputs are bounded. Though dissipative, the closed-loop system is not state strict passive, so this does not yield boundedness of the internal states of the lower block (i.e., $\tilde{W}(k)$) unless

PE is not needed with the modified update algorithms (3.40) and (3.41) of Theorem 3.1.3.

Theorem 3.1.8 (One-Layer NN Passivity of Tuning Algorithms without PE): The weight-tuning mechanisms (3.40) and (3.41) make the map from $\varepsilon(k + d(k))$ for algorithm (a) and $k_v r(k) + \varepsilon(k) + d(k)$ for algorithm (b) to $-\tilde{W}^T(k)\phi(k)$ a state strict passive map.

Proof: Algorithm (a). The revised dynamics relative to $\tilde{W}(k)$ using (3.40) are given by

$$\begin{aligned}\tilde{W}(k+1) = & [I - \alpha\phi(k)\phi^T(k)]\tilde{W}(k) - \alpha\phi(x(k))[\varepsilon(k) + d(k)]^T \\ & + \Gamma\|I - \alpha\phi(k)\phi^T(k)\|\hat{W}(k)\end{aligned}\quad (3.109)$$

Select the Lyapunov function candidate (3.105) and use (3.109) into (3.106) to obtain

$$\begin{aligned}\Delta J = & -[2 - \alpha\phi^T(k)\phi(k)][-\tilde{W}^T(k)\phi(k)] \times [-\tilde{W}^T(k)\phi(k)]^T \\ & + 2[1 - \alpha\phi(k)\phi^T(k) - \Gamma\|I - \alpha\phi(k)\phi^T(k)\|] \\ & \times [-\tilde{W}^T(k)\phi(k)]^T[\varepsilon(k) + d(k)] \\ & + \alpha\phi^T(k)\phi(k)[\varepsilon(k) + d(k)]^T \times [\varepsilon(k) + d(k)] - \chi + \gamma_0^2\end{aligned}\quad (3.110)$$

where χ is a constant given in Jagannathan and Lewis (1996a) and

$$\gamma_0^2 = \frac{\Gamma^2}{\alpha}(1 - \alpha\phi_{\max}^2)^2 W_{\max}^2\quad (3.111)$$

Note (3.110) is in power form for discrete-time systems given by (2.33). Similarly, it can be shown using (3.41) that the parameter update law is SSP.

3.1.4.3 Passivity of the Closed-Loop System

It should be noted that SSP of both the filtered tracking error system (3.99) in Figure 3.18 is SSP while the weight error block is passive using the tuning rules in Table 3.2. Thus using standard results (Landau 1979), it can be concluded that the closed-loop system is passive. Therefore, according to the passivity theorem one can conclude that the input/output signals of each block are bounded as

long as the disturbances are bounded. Though passive, however, the closed-loop system is not SSP so this does not yield boundedness of the internal states of the lower block ($\tilde{W}(k)$).

On the other hand, the enhanced tuning rules of Table 3.3 yield a SSP weight-tuning block in the figure, so that the closed-loop system is SSP. Thus the internal states of the lower block (e.g., $\tilde{W}(k)$) guarantee SSP of the closed-loop system, so that the norms of the internal states are bounded in terms of the power delivered to each block. Then boundedness of input/output signals assures state boundedness even without PE.

3.1.4.4 Passivity of the Multilayer NN

In this section the results of the previous subsection are extended for controllers using multilayer NN. It is shown that the three layer tuning algorithms in Theorem 3.1.4, where PE is required make the NN passive, whereas the tuning algorithms in Theorem 3.1.5, where PE is not required make the NN state strict passive. Similar results can be shown for the general case of multiple hidden layers. The implications for closed-loop passivity are detailed.

Theorem 3.1.9 (Three-Layer NN Passivity of Tuning Algorithms with PE): The weight-tuning algorithms (3.62) and (3.63) make the map from $W_i^T \hat{\varphi}_i(k) + B_i k_v r(k)$ to $\tilde{W}_i^T(k) \hat{\varphi}_i(k); i = 1, 2$, a passive map.

Thus, the weight error block is passive and the closed-loop filtered tracking error system (3.99) in Figure 3.18 is dissipative; this guarantees dissipativity of the closed-loop system (Landau 1979). By employing the passivity theorem (Landau 1979), we can conclude that the inputs/output signals of each block are bounded as long as the external inputs are bounded. Although dissipative, the closed-loop system is not state strict passive so this does not yield boundedness of the internal states of the lower block ($\tilde{W}_i(k); \forall i = 1, \dots, n$) unless PE holds.

The next proof shows why PE is not needed with the modified update algorithms.

Theorem 3.1.10 (Three-Layer NN Passivity without PE): The modified weight-tuning algorithms (3.89) and (3.90) make the map from $W_i^T \hat{\varphi}_i(k) + B_i k_v r(k)$ to $\tilde{W}_i^T(k) \hat{\varphi}_i(k); i = 1, 2$, a state strict passive map. Also the weight-tuning mechanisms (3.91) and (3.92) for a three-layer NN make the map from $W_3^T \tilde{\varphi}_3(k) + \varepsilon(k + d(k))$ for algorithm (a) and $(k_v r(k) + W_3^T \tilde{\varphi}_3(k) + \varepsilon(k) + d(k))$ for algorithm (b) to $-\tilde{W}_3^T(k) \varphi_3(x(k))$ a state strict passive map.

Thus the modified tuning algorithms guarantee SSP of the weight-tuning blocks, so that the closed-loop system is SSP. Therefore, internal stability can be guaranteed even in the absence of PE.

3.2 FEEDBACK LINEARIZATION

The previous section presented the NN controller design for a class of nonlinear discrete-time systems. To address the controller design for a broader class of nonlinear systems such as affine systems, one has to apply feedback linearization. For linear time-invariant (LTI) systems there are a wide variety of controller design techniques that achieve a range of performance objectives including state regulation, tracking of desired trajectories and so on. Design techniques include the linear quadratic regulator (LQR), H-infinity, other robust control techniques, adaptive control, classical approaches such as root locus and Bode design, and so on. Generally, as long as the system is controllable it is possible to design a controller using full-state feedback that gives good closed-loop performance. Some problems occur with nonminimum phase systems, but several techniques are now available for confronting these. If only the outputs can be measured, then good performance can be achieved by using a dynamic regulator as long as the system is both controllable and observable.

Unfortunately, for nonlinear systems control design is very complex. There are no universal techniques that apply for nonlinear systems; each nonlinear system must generally be considered as a separate design problem. Though there are techniques available such as Lyapunov, passivity, hyperstability, and variable structure (e.g., sliding mode) approaches, considerable design insight is still required. Feedback linearization techniques offer a widely applicable set of design tools that are useful for broader classes of nonlinear systems. They function basically by converting the nonlinear problem into a related linear controls design problem. They are more powerful, where they apply, than standard classical linearization techniques such as Lyapunov's indirect method where the nonlinear system is linearized using Jacobian techniques. However, it is very important to note that these techniques are available for continuous-time systems whereas very few techniques are available for discrete-time systems. References for this section include Astrom and Bjornmark (1989); Slotine and Li (1991).

3.2.1 INPUT-OUTPUT FEEDBACK LINEARIZATION CONTROLLERS

There are basically two feedback linearization techniques — input-state feedback linearization and input-output (i/o) feedback linearization. The former

requires a complex set of mathematical tools including Frobenius' theorem and Lie algebra. The control laws derived are often complex due to the need to determine nonlinear state-space transformations and their inverses. On the other hand, i/o feedback linearization is direct to apply and represents more of an engineering approach to control systems design. It is very useful for large classes of nonlinear controls problems including those treated in this book, which encompass robot manipulators, mechanical systems, and other Lagrangian systems.

3.2.1.1 Error Dynamics

This discussion involves i/o feedback linearization controller design of discrete-time systems in the general form

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= h(x(k)) \end{aligned} \tag{3.112}$$

Several approaches can be utilized to obtain the discrete-time Brunovsky form

$$\begin{aligned} z_1(k+1) &= z_2(k) \\ z_2(k+1) &= z_3(k) \\ &\vdots \\ z_n(k+1) &= f(x(k)) + g(x(k))u(k) \\ y(k) &= z_1(k) \end{aligned} \tag{3.113}$$

One can advance the prescribed output $y(k) = h(x(k))$ repeatedly, defining new states as $z_1(k) \equiv y(k)$, $z_2(k) \equiv y(k+1)$, $z_3(k) \equiv y(k+2)$, ..., until $u(k)$ appears. The result is the chained form with some additional internal dynamics.

Alternatively, one may take differences $y(k)$ until $u(k)$ appears — the first difference and second difference are defined as

$$\begin{aligned} \Delta y(k) &\equiv y(k+1) - y(k) \\ \Delta^2 y(k) &\equiv \Delta y(k+1) - \Delta y(k) = y(k+2) - 2y(k+1) + y(k) \end{aligned} \tag{3.114}$$

The new states are defined as $z_1(k) \equiv y(k)$, $z_2(k) \equiv \Delta y(k)$, $z_3(k) \equiv \Delta^2 y(k)$, ... This results in considerably complex computations, but is theoretically more appropriate.

3.2.2 CONTROLLER DESIGN

Provided the system has a well-defined relative degree so that $\|g(x(k))\| > 0$ for all k , the obvious choice for a control input that causes $y(k)$ to track the desired trajectory $y_d(k)$ is now

$$\begin{aligned} u(k) = \frac{1}{g(x(k))} &[-f(x(k)) + y_d(k+n) \\ &+ K_n e(k+n-1) + \cdots + K_1 e(k)] \end{aligned} \quad (3.115)$$

where the tracking error is defined as $e(k) = y_d(k) - y(k)$. This yields the closed-loop dynamics

$$e(k+n) + K_n e(k+n-1) + \cdots + K_2 e(k+1) + K_1 e(k) = 0 \quad (3.116)$$

which is stable as long as the gains are appropriately selected. This is a controller with an outer tracking loop and an inner nonlinear feedback linearization loop, as shown in Figure 3.19.

With respect to the defined output $y(k)$, the controller is noncausal. To compute the control input $u(k)$ at time k , one must know the values of $e(k)$ as well as

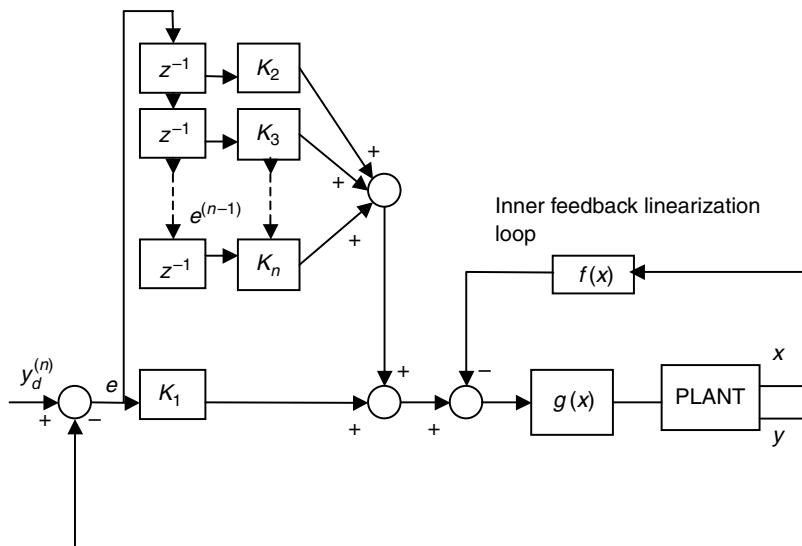


FIGURE 3.19 Feedback linearization controller showing PD outer loop and nonlinear inner loop.

its $n - 1$ future values. However, note that $e(k + j) = y_d(k + j) - y(k + j) = y_d(k + j) - z_{j+1}(k)$, for $0 < j < n$. In this work we shall assume full-state variable feedback, that is, all the states $x(k) = [x_1(k) \ x_2(k) \ \dots \ x_n(k)]^T$ are assumed available at time k for computation of the control input $u(k)$. If only the output $y(k)$ is available at time k , the controller design problem is considerably more complex — then a dynamic regulator containing a state observer must be designed. This can be accomplished by using either dynamic or recurrent NN as shown in He and Jagannathan (2004). Output feedback NN controller designs are treated in Chapter 6 and Chapter 10.

In the next section, NN feedback linearization of uncertain nonlinear discrete-time systems is discussed. If the nonlinearities are known, then (3.115) can be employed to design a controller. If the nonlinearities are unknown, then NNs have to be used. The next section details the NN controller for feedback linearization when the nonlinearities are unknown.

3.3 NN FEEDBACK LINEARIZATION

In Section 3.1, adaptive control of a class of nonlinear systems in discrete-time was presented using NN. Although Lyapunov stability analysis and passivity properties were detailed, the analysis was limited to a specific class of nonlinear systems of the form $x(k + 1) = f(x(k)) + u(k)$, where there are no uncertainties in the coefficient of the control input $u(k)$. However if a system in continuous-time of the form $\dot{x} = f_1(x) + u(k)$ is discretized, the system in discrete-time will be of the form $x(k + 1) = f(x(k)) + g(x(k))u(k)$ for some functions $f(\cdot)$ and $g(\cdot)$. Therefore in this section the results from Section 3.1 are extended to the more general class of nonlinear discrete-time systems in affine form $x(k + 1) = f(x(k)) + g(x(k))u(k)$. Many real-world applications are represented as affine nonlinear discrete-time systems.

It is important to note that for control purposes even if the open-loop system is stable, it must be shown that inputs, outputs, and states will remain bounded when a feedback loop is designed. In addition, if the controller is given in the form $u(k) = N(x)/D(x)$, then $D(x)$ must be nonzero for all time; we call this a well-defined controller. For feedback linearization, this type of controller is usually needed. If any adaptation scheme is implemented to provide an estimate $\hat{D}(x)$ of $D(x)$, then extra precaution is required to guarantee that $D(x) \neq 0$ for all time.

In Jagannathan and Lewis (1996a, 1996b), it has been shown that NN can effectively control in discrete-time a complex nonlinear system without the necessity of a regression matrix. There, the nonlinear systems under consideration are of the form $x(k + 1) = f(x(k)) + u(k)$ with the coefficient of the input

matrix being identity. Even though a multilayer NN controller in discrete-time (Chen and Khalil 1995) is employed to control a nonlinear system of the form $x(k+1) = f(x(k)) + g(x(k))u(k)$, an initial off-line learning phase is needed. There, $g(x(k))$ is reconstructed by an adaptive scheme as $\hat{g}(x(k))$, and a local solution is given by assuming that initial estimates are close to the actual values and they do not leave a feasible invariant set in which $\hat{g}(x(k)) \neq 0$. Unfortunately, even with very good knowledge of the system it is not easy to choose the initial weights to ensure that the NN approximates it. Therefore an off-line learning phase is used to identify the system in order to choose the weight values.

In Yesildirek and Lewis (1995), taking into account all these issues, a multilayer NN controller is designed in continuous-time to perform feedback linearization of systems in Brunovsky form. The motivation of this section (Jagannathan 1996d, 1996e) is to provide like-results in discrete-time. A family of novel learning schemes is presented here that does not require preliminary off-line training. The traditional problems with discrete-time adaptive control are overcome by using a single Lyapunov function containing both the parameter identification and the control errors. This at once guarantees both stable identification and stable tracking. However, it leads to complex proofs where it is necessary to compute the square with respect to several different variables. The use of a single Lyapunov function for tracking and estimation avoids the need for the CE assumption. Along the way various other standard assumptions in discrete-time adaptive control are also overcome, including PE, linearity-in-the-parameters and the need for tedious computation of a regression matrix. The problem of $\hat{g}(x(k)) \neq 0$ is confronted by appropriately selecting the weight updates as well as the control input.

First, we treat the design for one-layer NN (Jagannathan 1996d) where the weights enter linearly. In this case, we discuss the controller structure, various weight-update algorithms and PE definitions. Note that linearity in the NN weights is far milder than the usual adaptive control restriction of linearity in the unknown system parameters, since the universal approximation property of NN means any smooth nonlinear function can be reconstructed. Next, multilayer NN are employed for feedback linearization with rigorous stability analyses presented. Finally, passivity properties of discrete-time NN controllers are covered.

3.3.1 SYSTEM DYNAMICS AND TRACKING PROBLEM

In this section we describe the class of systems to be dealt with in this chapter and study the error dynamics using a specific feedback linearization controller. Consider an $m n$ th order MIMO discrete-time state feedback linearizable minimum phase nonlinear system (Chen and Khalil 1995) to be controlled, given

in the multivariable Brunovsky form as

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) + d(k) \end{aligned} \tag{3.117}$$

with state $x(k) = [x_1^T(k), \dots, x_n^T(k)]^T$ having $x_i(k) \in \Re^m; i = 1, \dots, n$, and control $u(k) \in \Re^m$. The nonlinear functions $f(\cdot)$ and $g(\cdot)$ are assumed to be unknown. The disturbance vector acting on the system at instant k is $d(k) \in \Re^m$ which we assume unknown but bounded so that $\|d(k)\| \leq d_M$ is a known constant. Further, the unknown smooth function satisfies the mild assumption

$$\|g(x(k))\| \geq g > 0 \tag{3.118}$$

with g being a known lower bound. The assumption given above on the smooth function $g(x(k))$ implies that $g(x(k))$ is strictly either positive or negative for all x . From now on, without loss of generality we will assume that $g(x(k))$ is strictly positive. Note that at this point that there is no general approach to analyze this class of unknown nonlinear systems. Adaptive control, for instance, needs an additional linear in the parameters assumption (Goodwin and Sin 1984; Åström and Wittenmark 1989).

Feedback linearization will be used to perform output tracking, whose objective can be described as: given a trajectory in terms of output, $x_{nd}(k)$ and its delayed values, find a control input $u(k)$ so that the system tracks the desired trajectory with an acceptable bounded error in the presence of disturbances while all the states and controls remain bounded. In order to continue, the following assumptions are required.

Assumption 3.3.1 (Bounds for System and Desired Trajectory):

1. The sign of $g(x(k))$ is known.
2. The desired trajectory vector with its delayed values is assumed to be available for measurement and bounded by an upper bound.

Given a desired trajectory $x_{nd}(k)$ and its delayed values, define the tracking error as

$$e_n(k) = x_n(k) - x_{nd}(k) \tag{3.119}$$

and the filtered tracking error, $r(k) \in \Re^m$,

$$r(k) = e_n(k) + \lambda_1 e_{n-1}(k) + \cdots + \lambda_{n-1} e_1(k) \quad (3.120)$$

where $e_{n-1}(k), \dots, e_1(k)$ are the delayed values of the error $e_n(k)$ and $\lambda_1, \dots, \lambda_{n-1}$ are constant matrices selected so that $|z^{n-1} + \lambda_1 z^{n-2} + \cdots + \lambda_{n-1}|$ is stable. Equation 3.120 can be expressed as

$$r(k+1) = e_n(k+1) + \lambda_1 e_{n-1}(k+1) + \cdots + \lambda_{n-1} e_1(k+1) \quad (3.121)$$

Using (3.117) in (3.121), the dynamics of the filtered tracking error system (3.121) can be written in terms of the tracking error as

$$\begin{aligned} r(k+1) = & f(x(k)) - x_{nd}(k+1) + \lambda_1 e_n(k) + \cdots + \lambda_{n-1} e_2(k) \\ & + g(x(k))u(k) + d(k) \end{aligned} \quad (3.122)$$

Equation 3.122 can be expressed as

$$r(k+1) = f(x(k)) + g(x(k))u(k) + d(k) + Y_d \quad (3.123)$$

where

$$Y_d = -x_{nd}(k+1) + \sum_{i=0}^{n-2} \lambda_{i+1} e_{n-i} \quad (3.124)$$

If we knew the functions $f(x(k))$ and $g(x(k))$ and when no disturbances are present, the control input $u(k)$ could be selected as the feedback linearization controller.

$$u(k) = \frac{1}{g(x(k))} (-f(x(k)) + v(k)) \quad (3.125)$$

with $v(k)$ taken as an auxiliary input given by

$$v(k) = k_{vr}(k) - Y_d \quad (3.126)$$

Then the filtered tracking error $r(k)$ goes to zero exponentially if we properly select the gain matrix k_v . Since the system functions are not known a priori, the control input $u(k)$ has to be selected as

$$u(k) = \frac{1}{\hat{g}(x(k))} (-\hat{f}(x(k)) + v(k)) \quad (3.127)$$

with $\hat{f}(x(k))$ and $\hat{g}(x(k))$ being the estimates of $f(x(k))$ and $g(x(k))$, respectively. It is important to note that, even in adaptive control of linear systems, guaranteeing the boundedness of $\hat{g}(x(k))$ away from zero becomes an important issue in this type of controller.

Equation 3.123 can be rewritten as

$$r(k+1) = u(k) - v(k) + f(x(k)) + g(x(k))u(k) + d(k) + Y_d \quad (3.128)$$

Substituting (3.117) and (3.127) for $v(k)$ in (3.128), Equation 3.128 can be rewritten as

$$r(k+1) = k_{vr}(k) + \tilde{f}(x(k)) + \tilde{g}(x(k))u(k) + d(k) \quad (3.129)$$

where the functional estimation errors are given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)) \quad (3.130)$$

and

$$\tilde{g}(x(k)) = g(x(k)) - \hat{g}(x(k)) \quad (3.131)$$

This is an error system wherein the filtered tracking error is driven by the functional estimation errors and unknown disturbances.

In this chapter, discrete-time NN are used to provide the estimates $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$. The error system (3.129) is used to focus on selecting discrete-time NN tuning algorithms that guarantee the stability of the filtered tracking error $r(k)$. Then, since (3.120), with the input considered as $r(k)$ and the output $e(k)$, describes a stable system, using the notion of operator gain (Jagannathan and Lewis 1996a) we can guarantee that $e(k)$ exhibits stable behavior.

3.3.2 NN CONTROLLER DESIGN FOR FEEDBACK LINEARIZATION

In this section we derive the error system dynamics and present the NN controller structure. NN weight-tuning algorithms are given for the one-layer case in Section 3.3.3 and for the multilayer case in Section 3.4.

3.3.2.1 NN Approximation of Unknown Functions

It will be necessary to review the notation given in Chapter 1 for multilayer NN. Assume that there exist some constant ideal weights W_f and W_g for two

one-layer NN and W_{1f} , W_{2f} , W_{3f} and W_{1g} , W_{2g} , W_{3g} for the three-layer NN case so that the nonlinear functions in (3.117) can be written as

$$f(x) = W_f^T \varphi_f(x(k)) + \varepsilon_f(k) \quad (3.132)$$

and

$$g(x) = W_g^T \varphi_g(x(k)) + \varepsilon_g(k) \quad (3.133)$$

For the case of two one-layer NN and

$$f(x) = W_{3f}^T \varphi_{3f}(k) + \varepsilon_f(k) \quad (3.134)$$

and

$$g(x) = W_{3g}^T \varphi_{3g}(k) + \varepsilon_g(k) \quad (3.135)$$

for the case of two three-layer NN. The notation $\varphi_3(k)$ was defined in Section 3.1. We assume that $\|\varepsilon_f(k)\| < \varepsilon_{Nf}$, $\|\varepsilon_g(k)\| < \varepsilon_{Ng}$ where the bounding constants ε_{Nf} and ε_{Ng} are known. The activation functions $\varphi_f(x(k))$ and $\varphi_g(x(k))$ must be selected to provide suitable basis sets for $f(\cdot)$ and $g(\cdot)$, respectively, for the one-layer NN case. Note that the activation functions in the case of multilayer do not need to form a basis, unlike the case of one-layer NN due to the universal approximation property of multilayer NN (Cybenko 1989).

Unless the network is minimal, the target weights may not be unique (Sontag 1992; Sussman 1992). The best weights may then be defined as those which minimize the supremum norm over S of $\varepsilon(k)$. This issue is not a major concern here as only the knowledge of existence of ideal weights is needed; their actual values are not required. Though this assumption is similar to Erzberger's assumptions, the major difference is that, while Erzberger's assumptions often do not hold, the approximation properties of NN guarantee that the ideal weights do always exist, if $f(x(k))$ and $g(x(k))$ are continuous over a compact set.

Let $x \in U$ a compact subset of R^n . Assume that $h(x(k)) \in C^\infty[U]$, that is, a smooth function $U \rightarrow R$, so that the Taylor series expansion of $h(x(k))$ exists. One can derive that $\|x(k)\| \leq d_{01} + d_{11}|r(k)|$. Then using the bound on $x(k)$ and expressing $h(x(k))$ as (3.136) yields an upper bound on $h(x(k))$ as

$$|h(x(k))| = |W_h^T \varphi_h(k) + \varepsilon_h(k)| \leq C_{01} + C_{11} \|r(k)\| \quad (3.136)$$

with C_{01} and C_{11} as computable constants. In addition, the hidden-layer activation functions, such as the radial basis functions (RBFs), sigmoids, and so on

are bounded by a known upper bound

$$\begin{aligned}\|\varphi_f(k)\| &\leq \varphi_{f\max} \\ \|\varphi_g(k)\| &\leq \varphi_{g\max}\end{aligned}\quad (3.137)$$

and

$$\|\varphi_{if}(k)\| \leq \varphi_{if\max} \quad i = 1, 2, 3 \quad (3.138a)$$

$$\|\varphi_{ig}(k)\| \leq \varphi_{ig\max} \quad i = 1, 2, 3 \quad (3.138b)$$

3.3.2.2 Error System Dynamics

Defining the NN functional estimates, employed to select the control input presented in (3.127) as

$$\hat{f}(x(k)) = \hat{W}_f^T(k)\varphi_f(x(k)) \quad (3.139)$$

and

$$\hat{g}(x(k)) = \hat{W}_g^T(k)\varphi_g(x(k)) \quad (3.140)$$

with $\hat{W}_f(k)$ and $\hat{W}_g(k)$ the current value of the weights. Similarly for the case of three-layer NN,

$$\hat{f}(x(k)) = \hat{W}_{3f}^T(k)\phi_{3f}(\hat{w}_{2f}^T\phi_{2f}(\hat{w}_{1f}^T\phi_{1f}(x(k)))) \quad (3.141)$$

and

$$\hat{g}(x(k)) = \hat{W}_{3g}^T(k)\phi_{3g}(\hat{w}_{2g}^T\phi_{2g}(\hat{w}_{1g}^T\phi_{1g}(x(k)))) \quad (3.142)$$

with $\hat{W}_{3f}(k)$, $\hat{W}_{2f}(k)$, $\hat{W}_{1f}(k)$, and $\hat{W}_{3g}(k)$, $\hat{W}_{2g}(k)$, $\hat{W}_{1g}(k)$, the current values of the weights. This yields the controller structure shown in Figure 3.20. The controller structure is very similar for the multilayer case as well. The output of the plant is processed through a series of delays to obtain the past values of the output, and fed as inputs to the NN so that the nonlinear function in (3.117) can be suitably approximated. Thus, the NN controller derived in a straightforward manner using filtered error notions naturally provides a dynamical NN structure. Note that neither the input $u(k)$ nor its past values are needed by the NN since

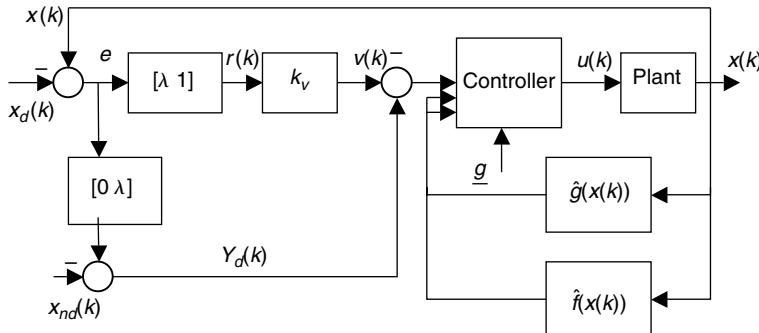


FIGURE 3.20 Discrete-time NN controller structure for feedback linearization.

the nonlinearities approximated by the NN do not require them. The next step is to determine the weight updates so that tracking performance of the closed-loop filtered error dynamics is guaranteed.

Let W_f and W_g be the unknown ideal weights required for the approximation to hold in (3.139) and (3.140) for the case of one-layer NN and W_{1f} , W_{2f} , W_{3f} , W_{1g} , W_{2g} , W_{3g} be the ideal weights for multilayer NN. The weight matrices for the case of multilayer NN are rewritten as

$$Z_f = \text{diag}(Z_{1f}, Z_{2f}, Z_{3f}) \quad (3.143)$$

and

$$Z_g = \text{diag}(Z_{1g}, Z_{2g}, Z_{3g}) \quad (3.144)$$

Assume they are bounded by known values so that

$$\|W_f(k)\| \leq W_{f\max} \quad (3.145)$$

and

$$\|W_g(k)\| \leq W_{g\max} \quad (3.146)$$

for the case of one-layer NN and

$$\|W_{3f}(k)\| \leq W_{3f\max} \quad (3.147)$$

$$\|W_{2f}(k)\| \leq W_{2f\max} \quad (3.148)$$

$$\|W_{1f}(k)\| \leq W_{1f\max} \quad (3.149)$$

and

$$\|W_{3g}(k)\| \leq W_{3g\max} \quad (3.150)$$

$$\|W_{2g}(k)\| \leq W_{2g\max} \quad (3.151)$$

$$\|W_{1g}(k)\| \leq W_{1g\max} \quad (3.152)$$

for the multilayer case. Similarly, the ideal weights for the case of multilayered NN are bounded by a known bound

$$\|Z_f(k)\| \leq Z_{f\max} \quad (3.153)$$

$$\|Z_g(k)\| \leq Z_{g\max} \quad (3.154)$$

Similarly, one can also define the matrix of current weights for the case of a multilayered NN as

$$\hat{Z}_f(k) = \text{diag}(\hat{Z}_{1f}(k), \hat{Z}_{2f}(k), \hat{Z}_{3f}(k)) \quad (3.155)$$

and

$$\hat{Z}_g(k) = \text{diag}(\hat{Z}_{1g}(k), \hat{Z}_{2g}(k), \hat{Z}_{3g}(k)) \quad (3.156)$$

Then the error in the weights during estimation is given by

$$\tilde{W}_f(k) = W_f - \hat{W}_f(k) \quad (3.157)$$

and

$$\tilde{W}_g(k) = W_g - \hat{W}_g(k) \quad (3.158)$$

for the case of one-layer NN and

$$\tilde{W}_{3f}(k) = W_{3f} - \hat{W}_{3f}(k) \quad (3.159)$$

$$\tilde{W}_{2f}(k) = W_{2f} - \hat{W}_{2f}(k) \quad (3.160)$$

$$\tilde{W}_{1f}(k) = W_{1f} - \hat{W}_{1f}(k) \quad (3.161)$$

$$\tilde{Z}_f(k) = Z_f - \hat{Z}_f(k) \quad (3.162)$$

with

$$\tilde{W}_{3g}(k) = W_{3g} - \hat{W}_{3g}(k) \quad (3.163)$$

$$\tilde{W}_{2g}(k) = W_{2g} - \hat{W}_{2g}(k) \quad (3.164)$$

$$\tilde{W}_{1g}(k) = W_{1g} - \hat{W}_{1g}(k) \quad (3.165)$$

$$\tilde{Z}_g(k) = Z_g - \hat{Z}_g(k) \quad (3.166)$$

for the case of multilayered NN. The error vector in the activation function is given by

$$\begin{aligned} \tilde{\varphi}_{1f}(k) &= \varphi_{1f}(k) - \hat{\varphi}_{1f}(k) \\ \tilde{\varphi}_{2f}(k) &= \varphi_{2f}(k) - \hat{\varphi}_{2f}(k) \\ \tilde{\varphi}_{3f}(k) &= \varphi_{3f}(k) - \hat{\varphi}_{3f}(k) \\ \tilde{\varphi}_{1g}(k) &= \varphi_{1g}(k) - \hat{\varphi}_{1g}(k) \\ \tilde{\varphi}_{2g}(k) &= \varphi_{2g}(k) - \hat{\varphi}_{2g}(k) \\ \tilde{\varphi}_{3g}(k) &= \varphi_{3g}(k) - \hat{\varphi}_{3g}(k) \end{aligned} \quad (3.167)$$

The closed-loop filtered dynamics (3.129) become

$$\begin{aligned} r(k+1) &= k_v(k) + \tilde{W}_f^T(k)\varphi_f(k) + \tilde{W}_g^T(k)\varphi_g(k)u(k) + \varepsilon_f(k) \\ &\quad + \varepsilon_g(k)u(k) + d(k) \end{aligned} \quad (3.168)$$

using one-layer NN and

$$\begin{aligned} r(k+1) &= k_v r(k) + \tilde{W}_{3f}^T(k)\hat{\varphi}_{3f}(k) + \tilde{W}_{3f}^T(k)\tilde{\varphi}_{3f}(k) + \tilde{W}_{3g}^T(k)\hat{\varphi}_{3g}(k)u(k) \\ &\quad + \varepsilon_f(k) + \varepsilon_g(k)u(k) + d(k) + W_{3g}^T(k)\tilde{\varphi}_{3g}(k)u(k) \end{aligned} \quad (3.169)$$

for the case of multilayered NN.

3.3.2.3 Well-Defined Control Problem

In general boundedness of $x(k)$, $\hat{W}_f(k)$, and $\hat{W}_g(k)$ does not indicate the stability of the closed-loop system because the control law (3.127) is not well defined when $\hat{g}(\hat{W}_g, x) = 0$. Therefore, some attention must be given to guaranteeing the boundedness of the controller as well. To overcome the problem, several techniques exist in the literature that assure local or global stability with additional knowledge. First if the bounds of the function $g(x)$ are known, then

$\hat{g}(\hat{W}_g, x)$ may be set to a constant and a robust-adaptive controller bypasses this problem. This is not an accurate approach because the bounds on the function are not known a priori. If $g(x)$ is reconstructed by an adaptive scheme then a local solution can be generated assuming that the initial estimates are close to the actual values and these values do not leave a feasible invariant set in which the $\hat{g}(\hat{W}_g, x)$ is not equal to zero (Liu and Chen 1991), or lie inside a region of attraction of a stable equilibrium point which forms a feasible set (Kanellakopoulos et al. 1991). Unfortunately even with good knowledge of the system, it is not easy to pick initial weight values to ensure that NN approximates it. The most popular way to avoid the problem is to project $\hat{W}_g(k)$ inside an estimated feasible region by properly selecting the weight values (Polycarpou and Ioannou 1991). A shortcoming of this approach is that the actual $\hat{W}_g(k)$ does not necessarily belong to this set, which then renders a suboptimal solution.

3.3.2.4 Controller Design

In order to guarantee the boundedness of $\hat{g}(x)$ away from zero for all well-defined values of $x(k)$, $\hat{W}_f(k)$, and $\hat{W}_g(k)$, the control input in (3.127) is selected in terms of another control input, $u_c(k)$, and a term, $u_r(k)$ used to make the system more robust, as

$$\begin{aligned} u(k) &= u_c(k) + \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\|-s)} \quad I = 0 \\ &= u_r(k) - \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\|-s)} \quad I = 1 \end{aligned} \quad (3.170)$$

where

$$u_c(k) = \hat{g}(x)^{-1}(-\hat{f}(x) + v(k)) \quad (3.171)$$

and

$$u_r(k) = -\mu \frac{\|u_c(k)\|}{g} \operatorname{sgn}(r(k)) \quad (3.172)$$

The indicator I in (3.170) is

$$I = \begin{cases} 1, & \text{if } \|\hat{g}(x)\| \geq g \text{ and } \|u_c(k)\| \leq s \\ 0, & \text{otherwise} \end{cases} \quad (3.173)$$

with $\gamma < \ln 2/s$, $\mu > 0$, and $s > 0$ design parameters. These modifications in the control input are necessary in order to ensure that the functional estimate $\hat{g}(x(k))$ is bounded away from zero.

The intuition behind this controller is as follows: When $\|\hat{g}(x(k))\| \geq g$ and $\|u_c(k)\| \leq s$, then the total control input is set to $u_c(k)$, otherwise the control is smoothly switched to the auxiliary input $u_r(k)$ due to the additional term in (3.170). This in turn results in well-defined control everywhere and the UUB of the closed-loop system can be shown by appropriately selecting the NN weight-tuning algorithms.

3.3.3 ONE-LAYER NN FOR FEEDBACK LINEARIZATION

In this section the one-layer NN is considered as a first step to bridging the gap between discrete-time adaptive control and NN control. As mentioned in the previous chapter, in the one-layer case the tunable NN weights enter in a linear fashion. The one-layer case is treated for RBFs in Sanner and Slotine (1992) using a projection algorithm in Polycarpou and Ioannou (1991). The assumptions used in this chapter are milder than in these works. In the next section the analysis is extended to the case of general multilayer NN with discrete-time tuning.

A family of one-layer NN weight-tuning paradigms that guarantee the stability of the closed-loop system (3.168) is presented in this section. It is required to demonstrate that the tracking error $r(k)$ is suitably small and that the NN weights $\hat{W}_f(k)$ and $\hat{W}_g(k)$ remain bounded. To proceed further, the machinery presented in the Lemma 3.1.2 and definition of the PE condition (see Definition 3.1.1) in Chapter 3 should be reviewed. Recall that for one-layer NN the activation functions must provide a basis. See the discussion on functional-link NN in Chapter 1.

Stability analysis by Lyapunov's direct method is performed using a novel weight-tuning algorithm for a one-layer NN developed based on the delta rule. These weight-tuning paradigms yield a passive neural net, yet PE is generally needed for suitable performance. Specifically, this holds for standard back-propagation as well in continuous-time (see Lewis et al. 1999). Unfortunately, PE cannot generally be tested for or guaranteed in an NN. Therefore, modified tuning paradigms are proposed in subsequent subsections to make the NN robust so that PE is not needed. For guaranteed stability, it is shown for the case of feedback linearization that the delta-rule-based weight-tuning algorithms must slow down as the NN becomes larger. By employing a projection algorithm it is shown that the tuning rate can be made independent of the NN size.

3.3.3.1 Weight Updates Requiring PE

In the following theorem we present a discrete-time weight-tuning algorithm given in Table 3.4, based on the filtered tracking error. The algorithm guarantees

TABLE 3.4
Discrete-Time Controller Using Three-Layer NN: PE Not Required

Select the control input $u(k)$ by

$$u(k) = \beta_0^{-1} [x_{nd}(k+1) - \hat{W}_3^T(k)\hat{\phi}_3(k) - \lambda_1 e_n(k) - \cdots - \lambda_{n-1} e_2(k) + k_v r(k)].$$

Consider the weight tuning provided for the

Input and hidden layers:

$$\hat{W}_1(k+1) = \hat{W}_1(k) - \alpha_1 \hat{\phi}_1(k)[\hat{y}_1(k) + B_1 k_v r(k)]^T - \Gamma \|I - \alpha_1 \hat{\phi}_1(k)\hat{\phi}_1^T(k)\| \hat{W}_1(k),$$

$$\hat{W}_2(k+1) = \hat{W}_2(k) - \alpha_2 \hat{\phi}_2(k)[\hat{y}_2(k) + B_2 k_v r(k)]^T - \Gamma \|I - \alpha_2 \hat{\phi}_2(k)\hat{\phi}_2^T(k)\| \hat{W}_2(k)$$

where $\hat{y}_i(k) = \hat{W}_i^T(k)\hat{\phi}_i(k)$, and

$$\|B_i\| \leq \kappa_i, \quad i = 1, 2.$$

Output layer: Tuning for the output layer is provided by either

$$(a) \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k)\bar{f}^T(k) - \Gamma \|I - \alpha_3 \hat{\phi}_3(k)\hat{\phi}_3^T(k)\| \hat{W}_3(k),$$

where $\bar{f}(k)$, is defined as the *functional augmented error* computed using

$$\bar{f}(k) = x_n(k+1) - u(k) - \hat{W}_3^T(k)\hat{\phi}_3(k),$$

or as

$$(b) \hat{W}_3(k+1) = \hat{W}_3(k) + \alpha_3 \hat{\phi}_3(k)r^T(k+1) - \Gamma \|I - \alpha_3 \hat{\phi}_3(k)\hat{\phi}_3^T(k)\| \hat{W}_3(k)$$

where $\alpha_i > 0$, $\forall i = 1, 2, 3$ denoting constant learning rate parameters or adaptation gains.

that both the tracking error and the error in the weight estimates are bounded if a PE condition holds. (This PE requirement is relaxed in Theorem 3.3.2.)

Theorem 3.3.1 (One-Layer Discrete-Time NN Controller Requiring PE): Let the desired trajectory $x_{nd}(k)$ be bounded and the NN functional reconstruction error-bound ε_{Nf} and ε_{Ng} with the disturbance-bound d_M be known constants. Take the control input for (3.117) as (3.170) with weight tuning provided for $f(x(k))$ by

$$\hat{W}_f(k+1) = \hat{W}_f(k) + \alpha \varphi_f(k)r^T(k+1) \quad (3.174)$$

and the weight tuning for $g(x(k))$ is expressed as

$$\begin{aligned} \hat{W}_g(k+1) &= \hat{W}_g(k) + \beta \varphi_g(k)r^T(k+1) \quad I = 1 \\ &= \hat{W}_g(k) \quad I = 0 \end{aligned} \quad (3.175)$$

with $\alpha > 0$ and $\beta > 0$ denoting constant learning rate parameters or adaptation gains.

Assume that the initial error in weight estimates for both NN are bounded and let the hidden-layer output vectors, $\varphi_f(k)$ and $\varphi_g(k)u_c(k)$ be persistently exciting. Then the filtered tracking error $r(k)$ and the error in weight estimates, $\tilde{W}_f(k)$, and $\tilde{W}_g(k)$ are UUB, with the bounds specifically given by (3.174) and (3.187) with (3.188) or (3.224) and (3.226) provided the following conditions hold.

1. $\beta \|\varphi_g(k)u_c(k)\|^2 = \beta \|\varphi_g(k)\|^2 < 1$
 2. $\alpha \|\varphi_f(k)\|^2 < 1$
 3. $\eta < 1$
 4. $\max(a_4, b_0) < 1$
- (3.176)

where η is given as

$$\eta = \alpha \|\varphi_f(k)\|^2 + \beta \|\varphi_g(k)u_c(k)\|^2 \quad (3.177)$$

for $I = 1$, and for $I = 0$, the parameter η is defined as

$$\eta = \alpha \|\varphi_f(k)\|^2 \quad (3.178)$$

and with a_4, b_0 design parameters chosen using the gain matrix $k_{v\max}$ and the relationship is presented during the proof.

Note: The parameters α, β , and η are dependent upon the trajectory.

Proof: (Note, in the proof $g(x(k))$ is also referred to as $g(x)$. Define the Lyapunov function candidate

$$J = r^T(k)r(k) + \frac{1}{\alpha} \text{tr}(\tilde{W}_f^T(k)\tilde{W}_f(k)) + \frac{1}{\beta} \text{tr}(\tilde{W}_g^T(k)\tilde{W}_g(k)) \quad (3.179)$$

The first difference is given by

$$\begin{aligned} \Delta J = & r^T(k+1)r(k+1) - r^T(k)r(k) + \frac{1}{\alpha} \text{tr}(\tilde{W}_f^T(k+1)\tilde{W}_f(k+1) \\ & - \tilde{W}_f^T(k)\tilde{W}_f(k)) + \frac{1}{\beta} \text{tr}(\tilde{W}_g^T(k+1)\tilde{W}_g(k+1) - \tilde{W}_g^T(k)\tilde{W}_g(k)) \end{aligned} \quad (3.180)$$

Region I: $\|\hat{g}(x)\| \geq g$ and $\|u_c(k)\| \leq s$.

The filtered error dynamics (3.168) can be rewritten as

$$\begin{aligned} r(k+1) = & k_v r(k) + (f(x(k)) - \hat{f}(x(k))) + (g(x(k)) - \hat{g}(x(k)))u_c(k) \\ & + d(k) + g(x(k))u_d(k) \end{aligned} \quad (3.181)$$

where $u_d(k) = u(k) - u_c(k)$. Substituting (3.127) and (3.171) in (3.181), one obtains

$$\begin{aligned} r(k+1) = & k_v r(k) + \tilde{W}_f^T(k)\varphi_f(k) + \tilde{W}_g^T(k)\varphi_g(k)u_c(k) + \varepsilon(k) \\ & + d(k) + g(x(k))u_d(k) \end{aligned} \quad (3.182)$$

where

$$\varepsilon(k) = \varepsilon_f(k) + \varepsilon_g(k)u_c(k) \quad (3.183)$$

Equation 3.182 can be rewritten

$$r(k+1) = k_v r(k) + \bar{e}_f^T(k) + \bar{e}_g^T(k) + \varepsilon(k) + d(k) + g(x(k))u_d(k) \quad (3.184)$$

where

$$\bar{e}_f(k) = \tilde{W}_f^T(k)\varphi_f(k) \quad (3.185)$$

$$\bar{e}_g(k) = \tilde{W}_g^T(k)\varphi_g(k)u_c(k) \quad (3.186)$$

Note that for the system defined in (3.184), the input $u_c(k) \in R^{m \times 1}$, and $\varphi_f(k) \in R^{mn \times 1}$ where $\varphi_i(k) \in R^{n \times 1}; i = 1, \dots, m$, and $\varphi_g(k) \in R^{mn \times n}$, in which each $\varphi_i(k) \in R^{n \times 1}; i = 1, \dots, m$. The error in dynamics for the weight update laws are given for this region as

$$\begin{aligned} \tilde{W}_f(k+1) = & (I - \alpha\varphi_f(k)\varphi_f^T(k))\tilde{W}_f(k) - \alpha\varphi_f(k)(k_v r(k) + \bar{e}_g(k) \\ & + g(x(k))u_d(k) + \varepsilon(k) + d(k))^T \end{aligned} \quad (3.187)$$

and

$$\begin{aligned} \tilde{W}_g(k+1) = & (I - \beta\varphi_g(k)\varphi_g^T(k))\tilde{W}_g(k) - \beta\varphi_g(k)(k_v r(k) + \bar{e}_f(k) \\ & + g(x(k))u_d(k) + \varepsilon(k) + d(k))^T \end{aligned} \quad (3.188)$$

Substituting (3.184), (3.185), and (3.188) in (3.180) and simplifying one obtains

$$\begin{aligned}\Delta J = & -r^T(k)[I - k_v^T k_v]r(k) + 2\eta(k_v r(k))^T(g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & + (1 + \eta)(g(x)u_d(k) + \varepsilon(k) + d(k))^T(g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & - (1 - \eta) \left\| (\bar{e}_f(k) + \bar{e}_g(k)) - \frac{\eta}{1 - \eta}(k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ & + \frac{\eta}{1 - \eta}(k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k))^T(k_v r(k) + g(x)u_d(k) \\ & + \varepsilon(k) + d(k))\end{aligned}\quad (3.189)$$

where

$$\eta = \alpha \|\varphi_f(k)\|^2 + \beta \|\varphi_g(k)\|^2 \quad (3.190)$$

Equation 3.189 can be rewritten as

$$\begin{aligned}\Delta J = & -(1 - a_1 k_{v \max}^2) \|r(k)\|^2 + 2a_2 k_{v \max} \|r(k)\| (g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & + a_3 (g(x)u_d(k) + \varepsilon(k) + d(k))^T (g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & - (1 - \eta) \left\| (\bar{e}_f(k) + \bar{e}_g(k)) - \frac{\eta}{1 - \eta}(k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right\|^2\end{aligned}\quad (3.191)$$

where

$$a_1 = 1 + \eta + \frac{\eta}{(1 - \eta)} \quad (3.192)$$

$$a_2 = \eta + \frac{\eta}{(1 - \eta)} \quad (3.193)$$

$$a_3 = 1 + \eta + \frac{\eta}{(1 - \eta)} \quad (3.194)$$

Now applying the condition on the function $g(x)$ on a compact set, one can conclude that

$$\|g(x)\| \leq C_{01} + C_{12} \|r(k)\| \quad (3.195)$$

with C_{01}, C_{12} being computable constants. Now, in this region, the bound for $u_d(k)$ (3.191) can be obtained as a constant since all the terms on the right side are bounded and this bound is denoted by

$$\|u_d(k)\| \leq C_2 \quad (3.196)$$

Now the bound for $g(x)u_d(k)$ is obtained as

$$\begin{aligned} \|g(x)u_d(k)\| &\leq C_2(C_{01} + C_{12}\|r(k)\|) \\ &\leq C_0 + C_1\|r(k)\| \end{aligned} \quad (3.197)$$

Using the bound presented in (3.197) for $g(x)u_d(k)$, the first difference of the Lyapunov function (3.191) is rewritten as

$$\begin{aligned} \Delta J = & -(1 - a_1 k_{v \max}^2) \|r(k)\|^2 + 2a_2 k_{v \max} \|r(k)\| (C_0 + C_1\|r(k)\| + \varepsilon_N + d_M) \\ & + a_3 (C_0 + C_1\|r(k)\| + \varepsilon_N + d_M)^T (C_0 + C_1\|r(k)\| + \varepsilon_N + d_M) \\ & - (1 - \eta) \left\| (\bar{e}_f(k) + \bar{e}_g(k)) - \frac{\eta}{1 - \eta} (k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \end{aligned} \quad (3.198)$$

with the bound for $\varepsilon(k)$ obtained as

$$\begin{aligned} \|\varepsilon(k)\| &\leq \|\varepsilon_f\| + \|\varepsilon_g u_c(k)\| \\ &\leq (\varepsilon_{Nf} + s\varepsilon_{Ng}) \\ &\leq \varepsilon_N \end{aligned} \quad (3.199)$$

Simplifying (3.198), one obtains

$$\begin{aligned} \Delta J = & -(1 - a_4) \|r(k)\|^2 + 2a_5 \|r(k)\| + a_6 - (1 - \eta) \left\| (\bar{e}_f(k) + \bar{e}_g(k)) \right. \\ & \left. - \frac{\eta}{1 - \eta} (k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \end{aligned} \quad (3.200)$$

where

$$a_4 = a_1 k_{v \max}^2 + 2a_2 C_1 k_{v \max} + a_3 C_1 \quad (3.201)$$

$$a_5 = a_2 k_{v \max} (\varepsilon_N + d_M + C_0) + a_3 C_1 (\varepsilon_N + d_M) + a_3 C_0 C_1 \quad (3.202)$$

and

$$a_6 = a_3 C_0^2 + 2a_3 C_0 (\varepsilon_N + d_M) + (\varepsilon_N + d_M)^2 \quad (3.203)$$

The second term in (3.200) is always negative as long as the condition (3.176) holds. Since a_4, a_5 , and a_6 are positive constants, $\Delta J \leq 0$ as long as (3.176) holds and

$$\|r(k)\| > \delta_{r1} \quad (3.204)$$

where

$$\delta_{r1} > \frac{1}{(1-a_4)} \left[a_3 + \sqrt{a_5^2 + a_6(1-a_4)} \right] \quad (3.205)$$

$|\sum_{k=k_0}^{\infty} \Delta J(k)| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (3.176) holds. The definitions of J and inequality (3.204) imply that every initial condition in the set χ will evolve entirely within χ . In other words, whenever the tracking error $\|r(k)\|$ is outside the region defined by (3.205), $J(r(k), \tilde{W}_f(k), \tilde{W}_g(k))$ will decrease. This further implies that the tracking error $r(k)$ is UUB for all $k \geq 0$ and it remains to show that the weight estimation errors $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$ or equivalently $\hat{W}_f(k)$ and $\hat{W}_g(k)$ are bounded.

Generally, in order to show the boundedness of the weight estimation errors, one uses the error in weight updates (3.187) and (3.188), the tracking error bound (3.204), the PE condition from (3.21), and Lemma 3.1.2. Using (3.187) and (3.188) it can be realized that the output of each NN is driving the other. Therefore, the boundedness of the tracking error, the PE condition and the Lemma are necessary but not sufficient. If the initial weight estimation errors for both NN are considered to be bounded, then applying the bound for the tracking error (3.204), the PE condition (3.21), and Lemma 3.1.2, we can show that the weight estimation errors $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$ or equivalently $\hat{W}_f(k)$ and $\hat{W}_g(k)$ are bounded. This concludes the boundedness of both, the tracking error and the weight estimates for both NN in this region. On the other hand, a similar and elegant way to show the boundedness of tracking error and weight estimates is to apply passivity theory. The proof using passivity theory is shown in Section 3.4.

Region II: $\|\hat{g}(x)\| \leq \underline{g}$ and $\|u_c(k)\| > s$.

Since the input $u_c(k)$ may not be defined in this region because of the notational simplicity, we will use it in the form of either $\hat{g}(x(k))u_c(k)$ or $u_c(k)e^{-\gamma(\|u_c(k)\|-s)}$. Therefore, in this region, the tracking error system given

in (3.168) is rewritten as

$$r(k+1) = k_v r(k) + \bar{e}_f^T(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k) \quad (3.206)$$

where

$$\overline{g(x)u_d(k)} = g(x)u(k) - \hat{g}(x(k))u_c(k) \quad (3.207)$$

Note that the extremum of the function $ye^{-\gamma y}$ for $\forall y > 0$ can be found as a solution to the following equation:

$$\frac{\partial(y^{-\gamma y})}{\partial y} = (1 - \gamma y)e^{-\gamma y} = 0 \quad (3.208)$$

which is $y = 1/\gamma$ and it is a maximum. Evaluating the function $u_c(k)e^{-\gamma u_c(k)}$ yields an upper bound for $u_c(k) = 1/\gamma e$ and this bound is used in the forthcoming set of equations.

Let us compute the bound for $g(x)u_c(k)$ and $\hat{g}(x)u_c(k)$. Consider the following cases in this region when $u_c(k) \leq s$ and $u_c(k) > s$. The bound on $u(k)$ from (3.170) can be written for this region as

$$\|u(k)\| \leq \frac{u_r(k) - u_c(k)}{2} e^{-\gamma(\|u_c(k)\|-s)} \quad (3.209)$$

Using (3.172) for $u_r(k)$, Equation 3.209 can be rewritten as

$$\|u_c(k)\| \leq \frac{1}{2} \left(\frac{\mu}{g} \|u_c(k)\| + \|u_c(k)\| \right) e^{-\gamma(|u_c(k)|-s)} \quad (3.210)$$

If $\|u_c(k)\| \leq s$ then $e^{\gamma s} \leq 2$, Equation 3.210 can be written as

$$|u(k)| \leq d_1 \quad (3.211)$$

where

$$d_1 = \left(\frac{\mu}{g} s + d_0 s \right) \quad (3.212)$$

bounded above by some positive constant. On the other hand, if $\|u_c(k)\| > s$, Equation 3.209 can be expressed as

$$\|u(k)\| \leq d_1 \quad (3.213)$$

where

$$d_1 = \frac{1}{2} \left(\frac{\mu}{\underline{g}} \frac{1}{\gamma e} + d_0 \frac{1}{\gamma e} \right) \quad (3.214)$$

Note here for simplicity the upper bound for $u(k)$ is denoted as d_1 in both cases. Now the bound for $g(x)u(k)$ can be obtained as

$$\|g(x)u(k)\| \leq C_0 + C_1 \|r(k)\| \quad (3.215)$$

where $C_0 = d_1 C_{01}$ and $C_1 = d_1 C_{12}$. Similarly the bound for $\hat{g}(x)u_c(k)$ can be deducted as

$$\begin{aligned} \|g(x)u(k)\| &\leq \underline{g}s && \text{if } \|u_c(k)\| \leq s \\ &\leq \frac{\underline{g}}{\gamma e} && \text{if } \|u_c(k)\| > s \end{aligned} \quad (3.216)$$

which is denoted as C_2 . Using the individual upper bounds of $g(x)u(k)$ and $\hat{g}(x)u_c(k)$, the upper bound for $|\overline{g(x)u_d(k)}|$ can be obtained as

$$\overline{|g(x)u_d(k)|} = |g(x)u(k) - \hat{g}u_c(k)| \leq C_3 + C_4 \|r(k)\| \quad (3.217)$$

where $C_3 = C_0 + C_2$ and $C_4 = C_1$. Now using the Lyapunov function (3.179), the first difference (3.180) after manipulation can be obtained for this region as

$$\begin{aligned} \Delta J &= -r(k)^T (I - k_v^T k_v) r(k) + 2(k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) \\ &\quad + d(k))^T (\overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k)) \\ &\quad + \frac{1}{(1 - \alpha \varphi_f^T(k) \varphi_f(k))} (k_v r(k) + \overline{g(x)u_d(k)} \\ &\quad + \varepsilon_f(k) + d(k))^T (k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k)) \\ &\quad - (1 - \eta) \left\| \bar{e}_f(k) - \frac{\eta}{(1 - \eta)} (k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k)) \right\|^2 \end{aligned} \quad (3.218)$$

where η is given in (3.177).

Substituting for $\overline{g(x)u_d(k)}$ from (3.217) in (3.218) and rearranging terms in (3.218) results in

$$\begin{aligned}\Delta J = & -(1 - b_0)\|r(k)\|^2 + 2b_1\|r(k)\| + b_2 \\ & - (1 - \eta) \left\| \bar{e}_f(k) - \frac{\eta}{1 - \eta} (k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k)) \right\|^2\end{aligned}\quad (3.219)$$

where

$$b_0 = k_{v \max}^2 + 2C_4(C_4 + k_{v \max}) + \frac{(C_4 + k_{v \max})^2}{1 - \alpha\|\varphi_f(k)\|^2} \quad (3.220)$$

$$\begin{aligned}b_1 = & C_3(C_4 + k_{v \max}) + C_3C_4(C_4 + k_{v \max})(\varepsilon_{Nf} + d_M) \\ & + \frac{C_3(C_4 + k_{v \max})}{1 - \alpha\|\varphi_f(k)\|^2} + \frac{(\varepsilon_{Nf} + d_M)(C_4 + k_{v \max})}{1 - \alpha\|\varphi_f(k)\|^2}\end{aligned}\quad (3.221)$$

$$\begin{aligned}b_2 = & 2C_3^2 + 2C_3(\varepsilon_{Nf} + d_M) + (\varepsilon_{Nf} + d_M)^2 \\ & + \frac{C_3^2 + 2C_3(\varepsilon_{Nf} + d_M) + (\varepsilon_{Nf} + d_M)^2}{(1 - \alpha\|\varphi_f(k)\|^2)}\end{aligned}\quad (3.222)$$

and

$$\|\varepsilon_f(k)\| \leq \varepsilon_{Nf} \quad (3.223)$$

The second term in (3.219) is always negative as long as the condition (3.176) holds. Since b_0, b_1 , and b_2 are positive constants, $\Delta J \leq 0$ as long as

$$\|r(k)\| > \delta_{r2} \quad (3.224)$$

with

$$\delta_{r2} = \frac{1}{(1 - b_0)} \left[b_1 + \sqrt{b_1^2 + b_2(1 - b_0)} \right] \quad (3.225)$$

One has $|\sum_{k=k_0}^{\infty} \Delta J(k)| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (3.176) holds. The definitions of J and inequality (3.224) imply that every initial condition in the set χ and will evolve entirely within χ . In other words, whenever the tracking error $\|r(k)\|$ is outside the region defined by (3.224),

$J(r(k), \tilde{W}_f(k), \tilde{W}_g(k))$ will decrease. This further implies that the tracking error $r(k)$ is UUB for all $k \geq 0$ and it remains to show that the weight estimation errors $\tilde{W}_f(k)$ or equivalently $\hat{W}_f(k)$ are bounded.

In order to show the boundedness of the weight estimation errors, one uses the error in weight updates (3.187) for $f(\cdot)$, the tracking error bound (3.224), the PE condition (3.21), and Lemma 3.1.2. Since the weight estimates for $\hat{g}(x)$ are not updated in this region, the boundedness of the weight estimates for $\hat{g}(x)$ need not be shown. However, to show the boundedness of the weight estimates for $\hat{f}(x)$, the dynamics relative to the error in weight estimates using (3.174) for this region are given by

$$\begin{aligned}\tilde{W}_f(k+1) = & (I - \alpha\varphi_f(k)\varphi_f^T(k))\tilde{W}_f(k) - \alpha\varphi_f(k)(k_v r(k) + C_3 + C_4\|r(k)\| \\ & + \varepsilon_f(k) + d(k))^T\end{aligned}\quad (3.226)$$

where the tracking error $r(k)$ is shown to be bounded. Applying the PE condition (3.21) and Lemma 3.1.2, the boundedness of weight estimation errors $\tilde{W}_f(k)$ or equivalently $\hat{W}_f(k)$ can be shown. Let us denote the bound by δ_{f2} . This concludes the boundedness of both the tracking error and the weight estimates for both NN in this region.

Reprise: Combining the results from Region I and II, one can readily set $\delta_r = \max\{\delta_{r1}, \delta_{r2}\}$, $\delta_f = \max\{\delta_{f1}, \delta_{f2}\}$, and δ_g . Thus for both regions, if $\|r(k)\| > \delta_{r1}$, then $\Delta J \leq 0$ and $u(k)$ is bounded. Let us denote $(\|r(k)\|, \|\tilde{W}_f\|, \|\tilde{W}_g\|)$ by new coordinate variables (ξ_1, ξ_2, ξ_3) . Define the region

$$\Xi : \xi | \xi_1 < \delta_r, \xi_2 < \delta_f, \xi_3 < \delta_g$$

then there exists an open set

$$\Omega : \xi | \xi_1 < \overline{\delta_r}, \xi_2 < \overline{\delta_f}, \xi_3 < \overline{\delta_g}$$

where $\overline{\delta_i} > \delta_i$ implies that $\Xi \subset \Omega$. In other words, we have proved that whenever $\xi_i > \delta_i$, then $J(\xi)$ will not increase and will remain in the region Ω which is an invariant set. Therefore all the signals in the closed-loop system remain bounded. This concludes the proof.

In applications, the right-hand sides of (3.204) or (3.224), (3.187) or (3.226) and (3.188) may be taken as practical bounds on the norms of the error $r(k)$ and the weight estimation errors $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$. Since the target weight values are bounded, it follows that the NN weights, $\hat{W}_f(k)$ and $\hat{W}_g(k)$, provided by the tuning algorithms are bounded; hence the control input is bounded.

Remarks: Note from (3.204) or (3.224) that the tracking error increases with the NN reconstruction error-bound ε_N and the disturbance-bound d_M , yet small tracking errors (but not arbitrarily small) may be achieved by selecting small gains k_v . In other words, placing the closed-loop poles closer to the origin inside the unit circle forces smaller tracking errors. Again as mentioned in Chapter 2, selecting $k_{v\max} = 0$ results in a deadbeat controller, but it should be avoided as it is not robust.

It is important to note that the problem of initializing the net weights (referred to as symmetric breaking) (Mpitos and Burton 1992) occurring in other techniques in the literature does not arise, since when $\hat{W}_f(0)$ and $\hat{W}_g(0)$ are taken as zero the PD term $k_v r(k)$ stabilizes the plant on an interim basis for a restricted class of nonlinear systems such as robotic systems. Thus, the NN controller requires no off-line learning phase.

3.3.3.2 Projection Algorithm

The NN adaptation gains, $\alpha > 0$ and $\beta > 0$, are constant parameters in the update laws presented in (3.174) and (3.175). These update laws correspond to the delta rule, also referred to as the Widrow–Hoff rule (Mpitsos and Burton 1992). This reveals that the update tuning mechanisms employing the delta rule have a major drawback. In fact, using (3.176), the upper bound on the adaptation gain for $g(x(k))$ can be obtained as

$$\beta < \frac{1}{\|\varphi(k)\|^2} \quad (3.227)$$

since $\varphi_g(k) \in \Re^{N_2}$, with N_2 the number of hidden-layer neurons. It is evident that the upper bound on the adaptation gain β depends upon the number of hidden-layer neurons. Specifically, if there are N_2 hidden-layer neurons and the maximum value of each hidden-node output is taken as unity (as for the sigmoid), then the bounds on the adaptation gain to assure stability of the closed-loop system are given by

$$0 < \beta < \frac{1}{N_2} \quad (3.228)$$

In other words, the upper bound on the adaptation gain for the case of delta rule decreases with an increase in the number of hidden-layer nodes, so that learning must slow down for guaranteed performance even with a feedback linearizing controller. The phenomenon of large NN requiring very slow learning rates has often been encountered in the practical NN literature

(Mpitsos and Burton 1992) but never explained adequately until the work of Jagannathan and Lewis (1996a).

This major drawback can be easily overcome by modifying the update rule at each layer to obtain a projection algorithm (Goodwin and Sin 1984). To wit, replace the constant adaptation gain at each layer by

$$\beta = \frac{\xi}{\zeta + \|\varphi_g(k)\|^2} \quad (3.229)$$

where

$$\zeta > 0 \quad (3.230)$$

and

$$0 < \xi < 1 \quad (3.231)$$

are constants. Note that ξ is now the new adaptation gain and it is always true that

$$\frac{\xi}{\zeta + \|\varphi_g(k)\|^2} \|\varphi_g(k)\|^2 < 1 \quad (3.232)$$

hence guaranteeing (3.176) for every N_2 . Similarly, using (3.176), it can be shown that the adaptation gain α should satisfy the constraint $\alpha < 1/\|\varphi_f(k)\|^2$.

Note that now for guaranteed closed-loop stability, it is necessary that the hidden-layer outputs $\varphi_f(k)$ and $\varphi_g(k)$ be PE. Equation 3.174 and Equation 3.175 are nothing but the delta-rule-based weight-tuning algorithms for the one-layer case. Then, the PE condition is required to guarantee boundedness of the weight estimates. However, it is very difficult to verify the PE of the hidden-layer output functions $\varphi_f(k)$ and $\varphi_g(k)$ and this problem is compounded due to the presence of hidden layers in the case of multilayered NN. In Section 3.3.3.3, improved weight-tuning paradigms are presented so that PE is not required.

3.3.3.3 Weight Updates Not Requiring PE

Approaches such as σ -modification (Polycarpou and Ioannou 1991) or ε -modification (Narendra and Annaswamy 1987) are available for the robust adaptive control of continuous systems where the PE condition is not needed. On the other hand, modification to the standard weight-tuning mechanisms in discrete-time to avoid the necessity of PE is also investigated in Jagannathan and Lewis (1996a, 1996b).

TABLE 3.5
Discrete-Time Controller Using One-Layer Neural Net: PE Not Required

The control input is

$$\begin{aligned} u(k) &= u_c(k) + \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 0 \\ &= u_r(k) - \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 1 \end{aligned}$$

where

$$u_c(k) = \hat{g}(x)^{-1}(-\hat{f}(x) + v(k))$$

and

$$u_r(k) = -\mu \frac{\|u_c(k)\|}{g} \operatorname{sgn}(r(k))$$

The indicator I is

$$I = \begin{cases} 1, & \text{if } \|\hat{g}(x)\| \geq g \text{ and } \|u_c(k)\| \leq s \\ 0, & \text{otherwise} \end{cases}$$

The NN weight tuning for $f(x(k))$ is given by

$$\hat{W}_f(k+1) = \hat{W}_f(k) + \alpha \varphi_f(k) r^T(k+1) - \delta \|I - \alpha \varphi_f(k) \varphi_f^T(k)\| \hat{W}_f(k)$$

and the NN weight tuning for $g(x(k))$ is given by

$$\begin{aligned} \hat{W}_g(k+1) &= \hat{W}_g(k) + \beta \varphi_g(k) r^T(k+1) - \rho \|I - \beta \varphi_g(k) \varphi_g^T(k)\| \quad I = 1 \\ &= \hat{W}_g(k) \quad I = 0 \end{aligned}$$

with $\alpha = \xi_f / (\xi_f + \|\varphi_f(k)\|^2)$ and $\beta = \xi_g / (\xi_g + \|\varphi_g(k)\|^2)$ where $\xi_f > 0$, $\xi_g > 0$, $0 < \xi_f < 1$, $0 < \xi_g < 1$ denoting learning rate parameters or adaptation gains.

In Jagannathan (1996b) an approach similar to ε -modification was derived for discrete-time NN for feedback linearization. The following theorem from that paper shows the tuning algorithms that do not require persistence of excitation. The controller derived therein is given in Table 3.5.

Theorem 3.3.2 (One-Layer Discrete-Time NN Feedback Linearization without PE): Assume the hypotheses presented in Theorem 3.3.1 and consider the modified weight-tuning algorithms provided for $f(x(k))$ by

$$\hat{W}_f(k+1) = \hat{W}_f(k) + \alpha \varphi_f(k) r^T(k+1) - \delta \|I - \alpha \varphi_f(k) \varphi_f^T(k)\| \hat{W}_f(k) \quad (3.233)$$

and the NN weight updates for $g(x(k))$ are provided by

$$\begin{aligned}\hat{W}_g(k+1) &= \hat{W}_g(k) + \beta\varphi_g(k)r^T(k+1) - \rho\|I - \beta\varphi_g(k)\varphi_g^T(k)\|\hat{W}_g(k) \quad I = 1 \\ &= \hat{W}_g(k) \quad I = 0\end{aligned}\quad (3.234)$$

with $\alpha > 0$, $\beta > 0$, $\delta > 0$, $\rho > 0$ design parameters. Then the filtered tracking error $r(k)$ and the NN weight estimates $\hat{W}_f(k)$ and $\hat{W}_g(k)$ are UUB, with the bounds specifically given by (3.257) or (3.278), and (3.261) or (3.282) and (3.265) provided the following conditions hold:

$$1. \quad \beta\|\varphi_g(k)u_c(k)\| = \beta\|\varphi_g(k)\|^2 < 1 \quad (3.235)$$

$$2. \quad \alpha\|\varphi_f(k)\|^2 < 1 \quad (3.236)$$

$$3. \quad \eta + \max(P_1, P_3, P_4) < 1 \quad (3.237)$$

$$4. \quad 0 < \delta < 1 \quad (3.238)$$

$$5. \quad 0 < \rho < 1 \quad (3.239)$$

$$6. \quad \max(a_2, b_0) < 1 \quad (3.240)$$

with P_1 , P_3 , and P_4 constants which depend upon η , δ , ρ where

$$\begin{aligned}\eta &= \alpha\|\varphi_f(k)\|^2 + \beta\|\varphi_g(k)u_c(k)\|^2 = \alpha\|\varphi_f(k)\|^2 + \beta\|\varphi_g(k)\|^2 \quad I = 1 \\ &= \alpha\|\varphi_f(k)\|^2 \quad I = 0\end{aligned}\quad (3.241)$$

and a_2 , b_0 design parameters chosen using the gain matrix k_v .

Note: The parameters α , β , and η depend upon the trajectory.

Proof: *Region I:* $\|\hat{g}(x)\| \geq g$ and $\|u_c(k)\| \leq s$.

Select the Lyapunov function candidate (3.179) whose first difference is given by (3.180). The error in dynamics for the weight-update laws are given for this region as

$$\begin{aligned}\tilde{W}_f(k+1) &= (I - \alpha\varphi_f(k)\varphi_f^T(k))\tilde{W}_f(k) - \alpha\varphi_f(k) \\ &\quad \times (k_{vr}(k) + \bar{e}_g(k) + g(x)u_d(k) + \varepsilon(k) + d(k))^T \\ &\quad + \delta\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|\hat{W}_f(k)\end{aligned}\quad (3.242)$$

and

$$\begin{aligned}\tilde{W}_g(k+1) = & (I - \beta\varphi_g(k)\varphi_g^T(k))\tilde{W}_g(k) - \beta\varphi_g(k) \\ & \times (k_v r(k) + \bar{e}_f(k) + g(x)u_d(k) + \varepsilon(k) + d(k))^T \\ & + \rho \|I - \beta\varphi_g(k)\varphi_g^T(k)\| \hat{W}_g(k)\end{aligned}\quad (3.243)$$

Substituting (3.242) and (3.243) in (3.180) and combining, rewriting, and completing the squares and simplifying we get

$$\begin{aligned}\Delta J = & -r(k)^T [I - (2 + \eta)k_v^T k_v] r(k) + 2(2 + \eta)(k_v r(k))^T (g(x)u_d(k) + \varepsilon(k) \\ & + d(k)) + (2 + \eta)(g(x)u_d(k) + \varepsilon(k) + d(k))^T (g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & + 2P_2 \|k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)\|^2 + 2\eta(g(x)u_d(k))^T (\varepsilon_k + d(k)) \\ & + 2\eta\varepsilon(k)d(k) - (1 - \eta - P_1) \|\bar{e}_f(k)\| \|\bar{e}_g(k)\| - \frac{1}{\eta} \|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2 \\ & \times [\delta(2 - \delta) \|\tilde{W}_f(k)\|^2 - 2\delta(1 - \delta) \|\tilde{W}_f(k)\| W_{f \max} - \delta^2 W_{f \max}^2] \\ & - \frac{1}{\beta} \|I - \beta\varphi_g(k)\varphi_g^T(k)\|^2 [\rho(2 - \rho) \|\tilde{W}_g(k)\|^2 - 2\rho(1 - \rho) \|\tilde{W}_g(k)\| \\ & \times W_{g \max} - \rho^2 W_{g \max}^2]\end{aligned}\quad (3.244)$$

where η is given in (3.241) and

$$P_1 = 2(\delta \|I - \alpha\varphi_f(k)\varphi_f^T(k)\| + \rho \|I - \beta\varphi_g(k)\varphi_g^T(k)\|) \quad (3.245)$$

$$P_2 = 2(\delta \|I - \alpha\varphi_f(k)\varphi_f^T(k)\| + \rho \|I - \beta\varphi_g(k)\varphi_g^T(k)\| W_{g \max} \varphi_{g \max}) \quad (3.246)$$

$$P_3 = (\eta + \delta \|I - \alpha\varphi_f(k)\varphi_f^T(k)\|)^2 \quad (3.247)$$

and

$$P_4 = (\eta + \rho \|I - \beta\varphi_g(k)\varphi_g^T(k)\|)^2 \quad (3.248)$$

Now, in this region, the bound on $u_d(k)$ can be obtained as

$$\begin{aligned}\|u_d(k)\| & \leq \|u(k) - u_c(k)\| \\ & \leq \left\| \frac{u_r(k) - u_c(k)}{2} e^{\gamma(|u_c(k)| - s)} \right\|\end{aligned}\quad (3.249)$$

In this region, since $\|u_c(k)\| \leq s$, and the auxiliary input $u_r(k)$ is given by (3.172), the bound in (3.249) can be taken as a constant since all terms on the right side are bounded and this bound is denoted as

$$\|u_d(k)\| \leq C_2 \quad (3.250)$$

Then the bound for $g(x)u_d(k)$ can be obtained as in (3.217). Using the bound for $g(x)u_d(k)$ and substituting in (3.244) and completing the squares we obtain

$$\begin{aligned} \Delta J &\leq -(1 - a_2)\|r(k)\|^2 + 2a_3\|r(k)\| + a_4 \\ &\quad - (1 - \eta - P_3)\|\bar{e}_f(k)\|^2 - (1 - \eta - P_4)\|\bar{e}_g(k)\|^2 \\ &\quad - 2(1 - \eta - P_1)\|\bar{e}_f(k)\|\|\bar{e}_g(k)\| \\ &\quad - \left\| \left(\sqrt{P_3}\bar{e}_f(k) + \sqrt{P_4}\bar{e}_g(k) \right) - (k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ &\quad - \frac{1}{\alpha} \|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2 \delta (2 - \delta) \left[\|\tilde{W}_f(k)\| - \frac{(1 - \delta)}{(2 - \delta)} W_{f \max}^2 \right]^2 \\ &\quad - \frac{1}{\beta} \|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2 \rho (2 - \rho) \left[\|\tilde{W}_g(k)\| - \frac{(1 - \rho)}{(2 - \rho)} W_{g \max}^2 \right]^2 \end{aligned} \quad (3.251)$$

where

$$a_2 = (2 + \eta)k_{v \max}^2 + 2(1 + \eta)C_1 k_{v \max} + (2 + \eta)C_1^2 + 2k_{v \max} C_1 \quad (3.252)$$

$$a_{31} = (1 + \eta)k_{v \max}(\varepsilon_N + d_M + C_0) + P_2 k_{v \max} + P_2 C_1 + \eta C_1(\varepsilon_N + d_M) \quad (3.253)$$

$$a_3 = a_{31} + \frac{1}{2}(2 + \eta)C_1(\varepsilon_N + d_M + C_0) + 2k_{v \max}(\varepsilon_N + d_M + C_0) \quad (3.254)$$

$$\begin{aligned} a_{44} &= 2P_2(\varepsilon_N + d_M + C_0) + 2\eta C_0(\varepsilon_N + d_M) + (2 + \eta)(\varepsilon_N + d_M + C_0)^2 \\ &\quad + 2\eta \varepsilon_N d_M \end{aligned} \quad (3.255)$$

and

$$\begin{aligned} a_4 &= a_{44} + \frac{1}{\alpha} \|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2 \frac{\delta^2}{(2 - \delta)} W_{f \ max}^2 \\ &\quad + \frac{1}{\beta} \|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2 \frac{\rho^2}{(2 - \rho)} W_{g \ max}^2 \end{aligned} \quad (3.256)$$

All the terms in (3.251) are always negative except the first term as long as the conditions (3.235) through (3.240) hold. Since a_2 , a_3 , and a_4 are positive

constants, $\Delta J \leq 0$ as long as (3.235) through (3.240) hold with

$$\|r(k)\| > \delta_{r1} \quad (3.257)$$

where

$$\delta_{r1} = \frac{1}{(1-a_2)} \left[a_3 + \sqrt{a_3^2 + a_4(1-a_2)} \right] \quad (3.258)$$

Similarly, completing the squares for $\|r(k)\|$, $\|\tilde{W}_g(k)\|$ using (3.251) yields

$$\begin{aligned} \Delta J = & -(1-a_2) \left[\|r(k)\| - \frac{a_3}{(1-a_2)} \right]^2 - (1-\eta-P_3)\|\bar{e}_f(k)\|^2 \\ & - (1-\eta-P_4)\|\bar{e}_g(k)\|^2 - 2(1-\eta-P_1)\|\bar{e}_f(k)\|\|\bar{e}_g(k)\| \\ & - \left\| \left(\sqrt{(P_3)}\bar{e}_f(k) + \sqrt{(P_4)}\bar{e}_g(k) - (k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \right) \right\|^2 \\ & - \frac{1}{\alpha} \|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2 \delta (2-\delta) \left[\|\tilde{W}_f(k)\| - \frac{(1-\delta)}{(2-\delta)} W_{f \max} - a_4 \right]^2 \\ & - \frac{1}{\beta} \|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2 \rho (2-\rho) \left[\|\tilde{W}_g(k)\| - \frac{(1-\rho)}{(2-\rho)} W_{g \max} \right]^2 \end{aligned} \quad (3.259)$$

where

$$\begin{aligned} a_4 = & \delta^2 W_{f \max}^2 + \frac{\alpha}{\|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2} \\ & \times \left[a_{44} + \frac{a_3^2}{(1-a_2)} + \frac{1}{\beta} \|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2 \frac{\rho^2}{2-\rho} W_{g \max}^2 \right] \end{aligned} \quad (3.260)$$

Then $\Delta J \leq 0$ as long as (3.235) through (3.240) hold and the quadratic term for $\tilde{W}_f(k)$ in (3.259) is positive, which is guaranteed when

$$\|\tilde{W}_f(k)\| > \delta_{f1} \quad (3.261)$$

where

$$\delta_{f1} = \frac{1}{(2-\delta)} \left[(1-\delta) + \sqrt{(1-\delta)^2 + a_4(2-\delta)} \right] \quad (3.262)$$

Similarly completing the squares for $\|r(k)\|, \|\tilde{W}_f(k)\|$ using (3.251) yields

$$\begin{aligned} \Delta J = & -(1 - a_2) \left[\|r(k)\| - \frac{a_3}{(1 - a_2)} \right]^2 \\ & - (1 - \eta - P_3) \|\bar{e}_f(k)\|^2 \\ & - (1 - \eta - P_4) \|\bar{e}_g(k)\|^2 - 2(1 - \eta - P_1) \|\bar{e}_f(k)\| \|\bar{e}_g(k)\| \\ & - \left\| \left(\sqrt{(P_3)} \bar{e}_f(k) + \sqrt{(P_4)} \bar{e}_g(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ & - \frac{1}{\alpha} \|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2 \delta (2 - \delta) \left[\|\tilde{W}_f(k)\| - \frac{(1 - \delta)}{(2 - \delta)} W_{f \max} \right]^2 \\ & - \frac{1}{\beta} \|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2 \rho (2 - \rho) \left[\|\tilde{W}_g(k)\| - \frac{(1 - \rho)}{(2 - \rho)} W_{g \max} - a_4 \right]^2 \end{aligned} \quad (3.263)$$

where

$$\begin{aligned} a_4 = & \rho^2 W_{g \max}^2 + \frac{\beta}{\|I - \beta \varphi_g(k) \varphi_g^T(k)\|^2} \left[a_{44} + \frac{a_3^2}{(1 - a_2)} \right. \\ & \left. + \frac{1}{\alpha} \|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2 \frac{\delta^2}{(2 - \delta)} W_{f \max}^2 \right] \end{aligned} \quad (3.264)$$

Then $\Delta J \leq 0$ as long as (3.235) through (3.240) hold and the quadratic term for $\tilde{W}_g(k)$ in (3.263) is positive, which is guaranteed when

$$\|\tilde{W}_g(k)\| > \delta_g \quad (3.265)$$

where

$$\delta_g = \frac{1}{(2 - \rho)} \left[(1 - \rho) + \sqrt{(1 - \rho)^2 + a_4(2 - \rho)} \right] \quad (3.266)$$

We have shown upper bounds for the tracking error and the NN weight estimation errors for this region for all $\|u_c(k)\| \leq s$.

Region II: $\|\hat{g}(x(k))\| < \underline{g}$, $\|u_c(k)\| > s$

Select the Lyapunov function candidate (3.179) whose first difference is given by (3.181). The tracking error system in (3.184) can be rewritten as

$$r(k+1) = k_v r(k) + \bar{e}_f^T(k) + \bar{g}(x) u_d(k) + \varepsilon_f(k) + d(k) \quad (3.267)$$

where

$$\overline{g(x)u_d(k)} = g(x)u(k) - \hat{g}(x)u_c(k) \quad (3.268)$$

For the case of modified weight tuning (3.323) through (3.324) in this region, let us denote the bound given in (3.268) as d_1 . The bound for $\hat{g}u_c(k)$ can be obtained as

$$\begin{aligned} \|\hat{g}(x)u_c(k)\| &\leq \underline{gs} \quad \|u_c(k)\| \leq s \\ &\leq \frac{g}{\gamma e} \quad \|u_c(k)\| > s \end{aligned} \quad (3.269)$$

whose upper bound in either case is denoted by C_2 . Using the individual upper bounds, the upper bound for $\overline{g(x)u_d(k)}$ can be obtained as (3.217).

Consider the first difference of the Lyapunov function, substitute the bounds for $\overline{g(x)u_d(k)}$, complete the squares and rearrange terms to obtain

$$\begin{aligned} \Delta J &= -(1 - b_0)\|r(k)\|^2 + 2b_1\|r(k)\| + b_2 - (1 - \eta) \\ &\times \left\| \overline{e_f(k)} - \frac{(n + \delta\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|)}{(1 - \eta)}(k_v r(k) + \overline{g(x)u_d(k)}) \right. \\ &\quad \left. + \varepsilon_f(k) + d(k) \right\|^2 - \frac{1}{\alpha}\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2[\delta(2 - \delta)\|\tilde{W}_f(k)\|^2 \\ &\quad - 2\delta(1 - \delta)\|\tilde{W}_f(k)\|W_{f \max} - \delta^2 W_{f \max}^2] \end{aligned} \quad (3.270)$$

where

$$b_0 = a_0 k_{v \max}^2 + 2a_0 C_4 k_{v \max} + a_0 C_4^2 \quad (3.271)$$

$$b_1 = a_0 k_{v \max} (C_3 + \varepsilon_{Nf} + d_M) + a_0 (C_3 + \varepsilon_{Nf} + d_M) C_4 (C_3 + \varepsilon_{Nf} + d_M) \quad (3.272)$$

$$b_2 = a_0 (C_3 + \varepsilon_{Nf} + d_M)^2 + 2\delta \|I - \alpha\varphi_f(k)\varphi_f^T(k)\| (C_3 + \varepsilon_{Nf} + d_M) W_{f \ max} \varphi_{f \ max} \quad (3.273)$$

$$a_0 = 1 + \alpha\varphi_f(k)\varphi_f^T(k) + \frac{\alpha\varphi_f(k)\varphi_f^T(k) + 2\delta \|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2}{(1 - \alpha\varphi_f(k)\varphi_f^T(k))} \quad (3.274)$$

and

$$\|\varepsilon_f(k)\| \leq \varepsilon_{Nf} \quad (3.275)$$

The second term in (3.270) is always negative as long as the conditions (3.235) through (3.241) hold. Completing the squares for $\|\tilde{W}_f(k)\|$ results in

$$\begin{aligned} \Delta J = & -(1 - b_0)\|r(k)\|^2 + 2b_1\|r(k)\| + b_3 - (1 - \eta) \\ & \times \left\| \overline{e_f}(k) - \frac{(n + \delta\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|)}{(1 - \eta)}(k_v r(k) + \overline{g(x)u_d(k)} \right. \\ & \left. + \varepsilon_f(k) + d(k)) \right\|^2 - \frac{1}{\alpha}\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2\delta(2 - \delta) \\ & \times \left[\|\tilde{W}_f(k)\| - \frac{(1 - \delta)}{(2 - \delta)}W_f^{\max} \right]^2 \end{aligned} \quad (3.276)$$

where

$$b_3 = b_2 + \frac{1}{\alpha}\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2 \frac{\delta}{(2 - \delta)}W_f^{\max} \quad (3.277)$$

Since b_0 , b_1 , and b_2 are positive constants in (3.276) and the second and third terms are always negative $\Delta J \leq 0$ as long as

$$\|r(k)\| > \delta_{r2} \quad (3.278)$$

where

$$\delta_{r2} = \frac{1}{(1 - b_0)} \left[b_1 + \sqrt{b_1^2 + b_3(1 - b_0)} \right]$$

Similarly completing the squares for $\|r(k)\|$ using (3.276) yields

$$\begin{aligned} \Delta J = & -(1 - b_0) \left[\|r(k)\| - \frac{b_1}{(1 - b_0)} \right]^2 - (1 - \eta) \left\| \overline{e_f}(k) \right. \\ & \left. - \frac{(n + \delta\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|)}{(1 - \eta)}(k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon_f(k) + d(k)) \right\|^2 \\ & - \frac{1}{\alpha}\|I - \alpha\varphi_f(k)\varphi_f^T(k)\|^2\delta(2 - \delta) \left[\|\tilde{W}_f(k)\| - 2\frac{(1 - \delta)}{(2 - \delta)}\|\tilde{W}_f(k)\|W_f^{\max} - b_4 \right] \end{aligned} \quad (3.279)$$

with

$$b_4 = \frac{1}{2-\delta} \left[\frac{\alpha}{\|I - \alpha \varphi_f(k) \varphi_f^T(k)\|^2} \frac{b_1^2}{(1-b_0)} + \delta^2 W_f \max \right] \quad (3.280)$$

Since b_0 , b_1 , and b_4 are positive constants in (3.279) and the second and third terms are always negative $\Delta J \leq 0$ as long as

$$\|\tilde{W}_f(k)\| > \delta_{f2} \quad (3.281)$$

where

$$\delta_{f2} = \frac{1}{(2-\delta)} \left[(1-\delta) + \sqrt{(1-\delta)^2 + b_4(2-\delta)} \right] \quad (3.282)$$

$|\sum_{k=k_0}^{\infty} \Delta J(k)| = J(\infty) - J(0) < \infty$ since $\Delta J \leq 0$ as long as (3.235) through (3.241) hold. The definitions of J and inequalities (3.278) and (3.281) imply that every initial condition in the set χ will evolve entirely within χ . Thus according to the standard Lyapunov extension (Lewis et al. 1999), it can be concluded that the tracking error $r(k)$ and the error in weight updates are UUB.

Reprise: Combining the results from region I and II, one can readily set

$$\delta_r = \max(\delta_{r1}, \delta_{r2}), \quad \delta_f = \max(\delta_{f1}, \delta_{f2}), \quad \text{and} \quad \delta_g$$

Thus, for both regions, if $\|r(k)\| > \delta_r$, then $\Delta J \leq 0$ and $u(k)$ is bounded. Let us denote $(\|r(k)\|, \|\tilde{W}_f(k)\|, \|\tilde{W}_g(k)\|)$ by the new coordinate variables (ξ_1, ξ_2, ξ_3) . Define the region

$$\Xi : \xi | \xi_1 < \delta_r, \quad \xi_2 < \delta_f, \quad \xi_3 < \delta_g$$

Then there exists an open set

$$\Omega : \xi | \xi_1 < \overline{\delta_r}, \quad \xi_2 < \overline{\delta_f}, \quad \xi_3 < \overline{\delta_g}$$

where $\overline{\delta_i} > \delta_i$, implies that $\Xi \subset \Omega$. In other words, it was shown that whenever $\xi_i > \delta_i$ then $V(\xi)$ will not increase and will remain in the region Ω which is an invariant set. Therefore all the signals in the closed-loop system remain UUB. This concludes the proof.

Remarks:

1. For practical purposes (3.257) or (3.178), (3.261) or (3.281), and (3.265) can be considered as bounds for $r(k)$, $\tilde{W}_f(k)$, and $\tilde{W}_g(k)$ in both regions.
2. The NN reconstruction errors and the bounded disturbances are all embodied in the constants given by δ_r , δ_f , and δ_g . Note that the bound on the tracking error may be kept small if the closed-loop poles are placed closer to the origin.
3. If the switching parameter s is chosen small, it will limit the control input and result in a large tracking error which results in undesirable closed-loop performance. Large value of s results in the saturation of the control input $u(k)$.
4. Uniform ultimate boundedness of the closed-loop system is shown without making assumptions on the initial weights. PE condition on the input signals is not required and the CE principle is not used. The NN can be easily initialized as $\tilde{W}_f(k) = 0$ and $\tilde{W}_g(k) > g^{-1}(\underline{g})$. In addition, the NN presented here do not need an off-line learning phase. Assumptions such as the existence of an invariant set, region of attraction, or a feasible region are not needed.

Note that the NN reconstruction error bound ε_N and the bounded disturbances d_M increase the bounds on $\|r(k)\|$ and $\|\tilde{W}_f(k)\|$ and $\|\tilde{W}_g(k)\|$ in a very interesting way. Note that small tracking error bounds, but not arbitrarily small, may be achieved by placing the closed-loop poles inside the unit circle and near the origin through the selection of the largest eigenvalue $k_{v\max}$. On the other hand, the NN weight error estimates are fundamentally bounded by $W_{f\max}$ and $W_{g\max}$ which are the known bounds on ideal weights W_f and W_g . The parameters δ and ρ offer a design trade-off between the relative eventual magnitudes of $\|r(k)\|$ and $\|\tilde{W}_f(k)\|$ and $\|\tilde{W}_g(k)\|$; a smaller δ yields a smaller $\|r(k)\|$ and a larger $\|\tilde{W}_f(k)\|$ and vice versa. For the tuning weights $\|\tilde{W}_g(k)\|$ similar effects are observed.

The effect of adaptation gains α and β at each layer on the weight estimation errors, $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$, and tracking error $r(k)$ can easily be observed by using the bounds presented in (3.258) or (3.282). Large values of α and β force smaller tracking and larger weight estimation errors. In contrast, small values of α and β force larger tracking and smaller weight estimation errors.

3.4 MULTILAYER NN FOR FEEDBACK LINEARIZATION

A family of multilayer NN weight-tuning paradigms that guarantee the stability of the closed-loop system (3.169) is presented in this section. It is required to

demonstrate that the tracking error $r(k)$ is suitably small and that the NN weights $\hat{W}_{1f}(k)$, $\hat{W}_{2f}(k)$, $\hat{W}_{3f}(k)$, and $\hat{W}_{1g}(k)$, $\hat{W}_{2g}(k)$, $\hat{W}_{3g}(k)$, remain bounded. To proceed further, the machinery presented in Lemma 3.1.2 and definition of Chapter 3 is needed.

3.4.1 WEIGHT UPDATES REQUIRING PE

In the following theorem we present a discrete-time weight-tuning algorithm given in Table 3.6 based on the filtered tracking error. The algorithm guarantees that both the tracking error and the error in the weight estimates are bounded if the PE condition holds. (This PE requirement is relaxed in Theorem 3.4.2.)

TABLE 3.6
Discrete-Time Controller Using Multilayer Neural NET: PE Required

The control input is

$$\begin{aligned} u(k) &= u_c(k) + \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 0 \\ &= u_r(k) - \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 1 \end{aligned}$$

where

$$u_c(k) = \hat{g}(x)^{-1}(-\hat{f}(x) + v(k))$$

and

$$u_r(k) = -\mu \frac{\|u_c(k)\|}{g} \operatorname{sgn}(r(k))$$

The indicator I is

$$I = \begin{cases} 1, & \text{if } \|\hat{g}(x)\| \geq g \text{ and } \|u_c(k)\| \leq s \\ 0, & \text{otherwise} \end{cases}$$

The NN weight tuning for $f(x(k))$ is given by

$$\begin{aligned} \hat{W}_{if}(k+1) &= \hat{W}_{if}(k) + \alpha_{if} \hat{\varphi}_{if}(k) (\hat{y}_{if}(k) + B_{if} k_v r(k))^T \quad i = 1, 2 \\ \hat{W}_{3f}(k+1) &= \hat{W}_{3f}(k) + \alpha_{3f} \hat{\varphi}_{3f}(k) r^T(k+1) \end{aligned}$$

and the NN weight tuning for $g(x(k))$ is given by

$$\begin{aligned} \hat{W}_{ig}(k+1) &= \hat{W}_{ig}(k) + \beta_{ig} \hat{\varphi}_{ig}(k) (\hat{y}_{ig}(k) + B_{ig} k_v r(k))^T \quad i = 1, 2 \\ \hat{W}_{3g}(k+1) &= \hat{W}_{3g}(k) + \alpha_{3g} \hat{\varphi}_{3g}(k) r^T(k+1) \quad I = 1 \\ &= \hat{W}_{3g}(k) \quad I = 0 \end{aligned}$$

with $\alpha_{if} > 0$; $i = 1, 2, 3$; $\beta_{ig} > 0$; $i = 1, 2, 3$ denoting constant learning rate parameters or adaptation gains.

Theorem 3.4.1 (Multilayer Discrete-Time NN Controller Requiring PE): Let the desired trajectory $x_{nd}(k)$ be bounded and the NN functional reconstruction error bound ε_{Nf} and ε_{Ng} along with the disturbance-bound d_M be known constants. Take the control input for (3.117) as (3.170) with weight-tuning for $f(x(k))$ provided by

$$\hat{W}_{if}(k+1) = W_{if}(k) + \alpha_{if}\hat{\varphi}_{if}(k)(\hat{y}_{if}(k) + B_{if}k_v r(k))^T \quad (3.283)$$

$$\hat{W}_{3f}(k+1) = W_{3f}(k) + \alpha_{3f}\hat{\varphi}_{3f}(k)r^T(k+1) \quad (3.284)$$

and the weight tuning for $g(x(k))$ is expressed as

$$\hat{W}_{ig}(k+1) = W_{ig}(k) + \beta_{ig}\hat{\varphi}_{ig}(k)(y_{ig}(k) + B_{ig}k_v r(k))^T \quad (3.285)$$

$$\hat{W}_{3g}(k+1) = W_{3g}(k) + \beta_{3g}\hat{\varphi}_{3g}(k)r^T(k+1) \quad (3.286)$$

with $\alpha_{if} > 0$; $i = 1, 2, 3$; $\beta_{ig} > 0$; $i = 1, 2, 3$ denoting constant learning rate parameters or adaptation gains. Then the filtered tracking error $r(k)$ and the error in weight estimates are UUB.

Note: PE is required.

Proof: See Jagannathan (1996e).

Example 3.4.1 (NN Control of Continuous-Time Nonlinear System): To illustrate the performance of the NN Controller, a continuous-time nonlinear system is considered and the objective is to control this feedback linearizable MIMO system by using a three-layer NN controller. Note that it is extremely difficult to discretize a nonlinear system and therefore offer stability proofs. It is important to note that the NN controllers derived here require no a priori knowledge of the dynamics of the nonlinear systems unlike conventional adaptive control and no initial learning phase is needed.

Consider the nonlinear system described by

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= F(X_1, X_2) + G_1(X_1, X_2)U \end{aligned} \quad (3.287)$$

where $X_1 = [x_1, x_2]^T$, $X_2 = [x_3, x_4]^T$, and the input vector is given by $U = [u_1, u_2]^T$ and the nonlinear function in (3.287) is described

by $F(X_1, X_2) = [M(X_1)]^{-1}G(X_1, X_2)$, with

$$M(X_1)$$

$$= \begin{bmatrix} (b_1 + b_2)a_1^2 + b_2a_2^2 + 2b_2a_1a_2 \cos(x_2) & b_2a_2^2 + b_2a_1a_2 \cos(x_1 + x_2) \\ b_2a_2^2 + b_2a_1a_2 \cos(x_2) & b_2a_2^2 \end{bmatrix} \quad (3.288)$$

$$G(X_1, X_2)$$

$$= \begin{bmatrix} -b_2a_1a_2(2x_3x_4 + x_4^2) \sin(x_2) + 9.8(b_1 + b_2)a_1 \cos(x_1) + \\ 9.8b_2a_2 \cos(x_1 + x_2) \\ b_2a_1a_2x_1^2 \sin(x_2) + 9.8b_2a_2 \cos(x_1 + x_2) \end{bmatrix} \quad (3.289)$$

and

$$G_1(X_1, X_2) = M^{-1}(X_1, X_2) \quad (3.290)$$

The parameters for the nonlinear system were selected as $a_1 = a_2 = 1$, $b_1 = b_2 = 1$. Desired sinusoidal, $\sin(2\pi t/25)$ and cosine inputs, $\cos(2\pi t/25)$ were preselected for the axes 1 and 2, respectively. The continuous-time gains of the PD controller were chosen as $k_v = \text{diag}(20, 20)$ with $\Lambda = \text{diag}(5, 5)$ and a sampling interval of 10 msec was considered. Three-layer NN were selected with ten hidden-layer nodes. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for X_1 were chosen to be $[0.5, 0.1]^T$, and the weights for $F(\cdot)$ were initialized to zero whereas the weights of $G(\cdot)$ were initialized to identity matrix. No off-line learning is performed initially to train the networks. Figure 3.21 presents the tracking response of the NN controller with delta-rule weight tuning (3.283) through (3.286) with $\alpha_3 = 0.1$, $\alpha_i = 1.0$; $\forall i = 1, 2$ and $\beta_3 = 0.1$, $\beta_i = 1.0$; $\forall i = 1, 2$. From the figure, it can be seen that the delta-rule-based weight tuning performs impressively.

3.4.2 WEIGHT UPDATES NOT REQUIRING PERSISTENCE OF EXCITATION

Approaches such as σ -modification (Polycarpou and Ioannou 1991) or ε -modification (Narendra and Annaswamy 1987) are available for the robust adaptive control of continuous systems where the PE condition is not needed. On the other hand, modification to the standard weight-tuning mechanisms in discrete-time to avoid the necessity of PE is also investigated in Jagannathan and Lewis (1996b) using multilayered NN for a specific class of nonlinear systems.

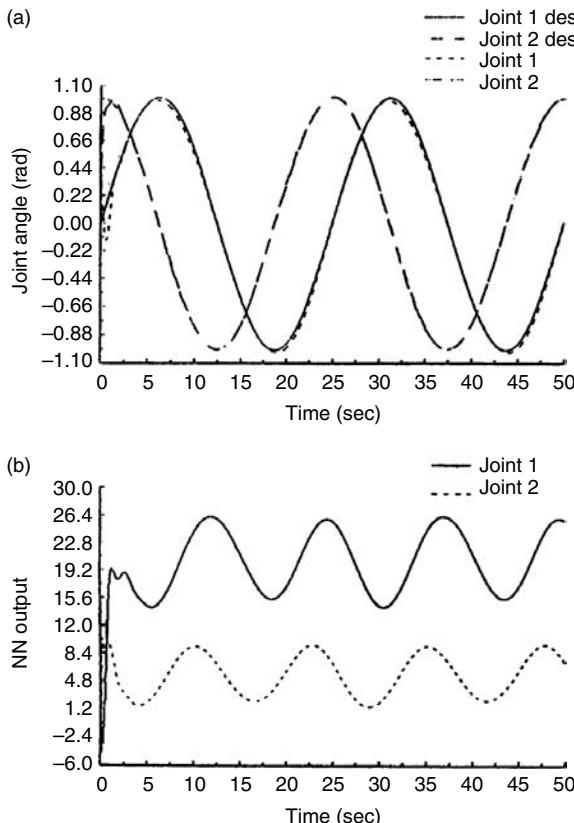


FIGURE 3.21 Response of the NN controller with delta-rule weight-tuning. (a) Actual and desired joint angles. (b) NN outputs.

In Jagannathan (1996e) an approach similar to ε -modification was derived for discrete-time NN for feedback linearization. The following theorem from that paper shows the tuning algorithms that do not require persistence of excitation. The controller derived therein is given in Table 3.7.

Theorem 3.4.2 (Multilayer NN Feedback Linearization without PE): Assume the hypotheses presented in Theorem 3.4.1 and consider the modified weight-tuning algorithms provided for $f(x(k))$ by

$$\begin{aligned} \hat{W}_{if}(k+1) &= \hat{W}_{if}(k) + \alpha_{if}\hat{\varphi}_{if}(k)(\hat{y}_{if}(k) + B_{if}k_v r(k))^T \\ &\quad - \delta_{if}\|I - \alpha_{if}\hat{\varphi}_{if}(k)\hat{\varphi}_{if}^T(k)\| \quad i = 1, 2 \end{aligned} \quad (3.291)$$

TABLE 3.7**Discrete-Time Controller Using Multilayer NN: PE Not Required**

The control input is

$$\begin{aligned} u(k) &= u_c(k) + \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 0 \\ &= u_r(k) - \frac{u_r(k) - u_c(k)}{2} e^{\gamma(\|u_c(k)\| - s)} \quad I = 1 \end{aligned}$$

where

$$u_c(k) = \hat{g}(x)^{-1}(-\hat{f}(x) + v(k))$$

and

$$u_r(k) = -\mu \frac{\|u_c(k)\|}{g} \operatorname{sgn}(r(k))$$

The indicator I is

$$I = \begin{cases} 1, & \text{if } \|\hat{g}(x)\| \geq g \text{ and } \|u_c(k)\| \leq s \\ 0, & \text{otherwise} \end{cases}$$

The NN weight tuning for $f(x(k))$ is given by

$$\begin{aligned} \hat{W}_{if}(k+1) &= \hat{W}_{if}(k) + \alpha_{if} \hat{\varphi}_{if}(k) (\hat{y}_{if}(k) + B_{if} k_v r(k))^T \\ &\quad - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\| \quad i = 1, 2 \\ \hat{W}_{3f}(k+1) &= \hat{W}_{3f}(k) + \alpha_{3f} \hat{\varphi}_{3f}(k) r^T(k+1) - \delta_{3f} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\| \end{aligned}$$

and the NN weight tuning for $g(x(k))$ is given by

$$\begin{aligned} \hat{W}_{ig}(k+1) &= \hat{W}_{ig}(k) + \beta_{ig} \hat{\varphi}_{ig}(k) (\hat{y}_{ig}(k) + B_{ig} k_v r(k))^T \\ &\quad - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\| \quad i = 1, 2 \\ \hat{W}_{3g}(k+1) &= \hat{W}_{3g}(k) + \beta_{3g} \hat{\varphi}_{3g}(k) r^T(k+1) - \rho_{3g} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\| \quad I = 1 \\ \hat{W}_{3g}(k+1) &= \hat{W}_{3g}(k) \quad I = 0 \end{aligned}$$

with $\alpha_{if} > 0; i = 1, 2, 3; \beta_{ig} > 0; i = 1, 2, 3, \delta_{if} > 0; i = 1, 2, 3, \rho_{ig} > 0; i = 1, 2, 3$

denoting constant learning rate parameters or adaptation gains.

$$\hat{W}_{3f}(k+1) = \hat{W}_{3f}(k) + \alpha_{3f} \hat{\varphi}_{3f}(k) r^T(k+1) - \delta_{3f} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\| \quad (3.292)$$

and the weight tuning for $g(x(k))$ is expressed as

$$\begin{aligned} \hat{W}_{ig}(k+1) &= \hat{W}_{ig}(k) + \beta_{ig} \hat{\varphi}_{ig}(k) (\hat{y}_{ig}(k) + B_{ig} k_v r(k))^T \\ &\quad - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\| \quad i = 1, 2 \end{aligned} \quad (3.293)$$

$$\begin{aligned}\hat{W}_{3g}(k+1) &= \hat{W}_{3g}(k) + \beta_{3g}\hat{\varphi}_{3g}(k)r^T(k+1) \\ &\quad - \rho_{3g}\|I - \beta_{3g}\hat{\varphi}_{3g}(k)\hat{\varphi}_{3g}^T(k)\| \quad I = 1 \\ \hat{W}_{3g}(k+1) &= \hat{W}_{3g}(k) \quad I = 0\end{aligned}\quad (3.294)$$

with $\alpha_{ig} > 0$; $i = 1, 2, 3$, $\beta_{ig} > 0$; $i = 1, 2, 3$, $\delta_{ig} > 0$; $i = 1, 2, 3$, $\rho_{ig} > 0$; $i = 1, 2, 3$ design parameters. Then the filtered tracking error $r(k)$ and the NN weight estimates $\hat{W}_{if}(k)$; $i = 1, 2, 3$ and $\hat{W}_{ig}(k)$; $i = 1, 2, 3$ are UUB, with the bounds specifically given by (3.326) or (3.342), (3.331) or (3.346) and (3.336) provided the following conditions hold:

$$1. \quad \beta_{3g}\|\overline{\varphi_{3g}}(k)u_c(k)\| = \beta_{3g}\|\hat{\varphi}_{3g}(k)\|^2 < 1 \quad (3.295)$$

$$2. \quad \alpha_{if}\|\hat{\varphi}_{if}(k)\|^2 < 2 \quad i = 1, 2 \quad (3.296)$$

$$3. \quad \beta_{if}\|\hat{\varphi}_{if}(k)\|^2 < 2 \quad i = 1, 2 \quad (3.297)$$

$$4. \quad \eta + \max(P_1, P_3, P_4) < 1 \quad (3.298)$$

$$5. \quad 0 < \delta_{ig} < 1 \quad \forall i = 1, 2, 3 \quad (3.299)$$

$$6. \quad 0 < \rho_{if} < 1 \quad \forall i = 1, 2, 3 \quad (3.300)$$

$$7. \quad \max(a_5, b_6) < 1 \quad (3.301)$$

with P_1 , P_3 , and P_4 constants which depend upon η , δ , ρ where

$$\eta = \alpha_{3f}\|\hat{\varphi}_{3f}(k)\|^2 + \beta_{3g}\|\overline{\hat{\varphi}_{3g}}(k)u_c(k)\|^2 \quad (3.302)$$

$$\eta = \alpha_{3f}\|\hat{\varphi}_{3f}(k)\|^2 + \beta_{3g}\|\hat{\varphi}_{3g}(k)\|^2 \quad I = 1 \quad (3.303)$$

$$\eta = \alpha_{3f}\|\hat{\varphi}_{3f}(k)\|^2 \quad I = 0$$

and a_5 and b_6 are design parameters chosen using the gain matrix k_v .

Proof: The proof is done by dividing the state space into two different regions.

Region I: $\|\hat{g}(x(k))\| \geq g$, and $\|u_c(k)\| \leq s$

Define the Lyapunov function candidate

$$J = r^T(k)r(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_i} \text{tr}(\tilde{W}_{if}^T(k)\tilde{W}_{if}(k)) + \frac{1}{\beta_i} \text{tr}(\tilde{W}_{ig}^T(k)\tilde{W}_{ig}(k)) \right] \quad (3.304a)$$

whose first difference is given by

$$\begin{aligned} \Delta J = & r^T(k+1)r(k+1) - r^T(k)r(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_i} \text{tr}(\tilde{W}_{if}^T(k+1)\tilde{W}_{if}(k+1) \right. \\ & \left. - \tilde{W}_{if}^T(k)\tilde{W}_{if}(k)) + \frac{1}{\beta_i} \text{tr}(\tilde{W}_{ig}^T(k+1)\tilde{W}_{ig}(k+1) - \tilde{W}_{ig}^T(k)\tilde{W}_{ig}(k)) \right] \end{aligned} \quad (3.304b)$$

Select the Lyapunov function candidate (3.304a) whose first difference is given by (3.304b). The error in dynamics for the weight update laws are given for this region as

$$\begin{aligned} \tilde{W}_{if}(k+1) = & (I - \alpha_{if}\hat{\varphi}_{if}(k)\hat{\varphi}_{if}^T(k))\tilde{W}_{if}(k) - \alpha_{if}\hat{\varphi}_{if}(k)(y_{if} + B_{if}k_v r(k))^T \\ & - \delta_{if}\|I - \alpha_{if}\hat{\varphi}_{if}(k)\hat{\varphi}_{if}^T(k)\|W_{if}(k) \quad i = 1, 2 \end{aligned} \quad (3.305)$$

$$\begin{aligned} \tilde{W}_{ig}(k+1) = & (I - \beta_{ig}\hat{\varphi}_{ig}(k)\hat{\varphi}_{ig}^T(k))\tilde{W}_{ig}(k) - \beta_{ig}\hat{\varphi}_{ig}(k)(y_{ig} + B_{ig}k_v r(k))^T \\ & - \rho_{ig}\|I - \beta_{ig}\hat{\varphi}_{ig}(k)\hat{\varphi}_{ig}^T(k)\|W_{ig}(k) \quad i = 1, 2 \end{aligned} \quad (3.306)$$

$$\begin{aligned} \tilde{W}_{3f}(k+1) = & (I - \alpha_{3f}\hat{\varphi}_{3f}(k)\hat{\varphi}_{3f}^T(k))\tilde{W}_{3f}(k) - \alpha_{3f}\hat{\varphi}_{3f}(k)(k_v r(k) + \bar{e}_g(k) \\ & + g(x(k))u_d(k) + \varepsilon(k) + d(k))^T - \delta_{3f}\|I - \alpha_{3f}\hat{\varphi}_{3f}(k)\hat{\varphi}_{3f}^T(k)\| \end{aligned} \quad (3.307)$$

$$\begin{aligned} \tilde{W}_{3g}(k+1) = & (I - \beta_{3g}\hat{\varphi}_{3g}(k)\hat{\varphi}_{3g}^T(k))\tilde{W}_{3g}(k) - \beta_{3g}\hat{\varphi}_{3g}(k) \\ & \times (k_v r(k) + \bar{e}_f(k) + g(x(k))u_d(k) + \varepsilon(k) + d(k))^T \\ & - \rho_{3g}\|I - \beta_{3g}\hat{\varphi}_{3g}(k)\hat{\varphi}_{3g}^T(k)\| \end{aligned} \quad (3.308)$$

Substituting (3.305) through (3.308) in (3.304b), combining, substituting for $g(x)u_d(k)$ from (3.197), rewriting and completing the squares for $\|\tilde{W}_{if}(k)\|$; $i = 1, 2, 3$ and $\|\tilde{W}_{ig}(k)\|$; $i = 1, 2, 3$, one obtains

$$\begin{aligned} \Delta J \leq & -(1 - a_2)\|r(k)\|^2 + 2a_3\|r(k)\| + a_4 - (1 - \eta - P_3)\|\bar{e}_f(k)\|^2 \\ & - (1 - \eta - P_4)\|\bar{e}_g(k)\|^2 - 2(1 - \eta - P_1)\|\bar{e}_f(k)\|\|\bar{e}_g(k)\| \\ & - \|(\sqrt{P_3}\bar{e}_f(k) + \sqrt{P_4}\bar{e}_g(k)) - (k_v r(k) + g(x)u_d(k) + \varepsilon(k) + d(k))\|^2 \end{aligned}$$

$$\begin{aligned}
& + \sum_{i=1}^3 \frac{1}{\alpha_{if}} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|^2 \text{tr}[\delta_{if}^2 \hat{W}_{if}^T(k) \hat{W}_{if}(k)] \\
& + \sum_{i=1}^3 \frac{1}{\beta_{ig}} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|^2 \text{tr}[\rho_{ig}^2 \hat{W}_{ig}^T(k) \hat{W}_{ig}(k)] \\
& - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\
& \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \left. \right\|^2 \\
& + P_5 \|r(k)\|^2 + 2P_6 \|r(k)\| + P_7
\end{aligned} \tag{3.309}$$

where

$$a_2 = (2 + \eta) k_{v \max}^2 + 2(1 + \eta) C_1 k_{v \max} + (2 + \eta) C_1^2 + 2k_{v \max} C_1 \tag{3.310}$$

$$a_3 = (1 + \eta) k_{v \max} (\varepsilon_N + d_M + C_0) + P_2 k_{v \max} + P_2 C_1 + \eta C_1 (\varepsilon_N + d_M)$$

$$+ \frac{1}{2} (2 + \eta) C_1 (\varepsilon_N + d_M + C_0)^2 + 2k_{v \max} (\varepsilon_N + d_M + C_0) \tag{3.311}$$

$$\begin{aligned}
a_{44} &= 2P_2 (\varepsilon_N + d_M + C_0) + 2\eta C_0 (\varepsilon_N + d_M) \\
&+ (2 + \eta) C_1 (\varepsilon_N + d_M + C_0)^2 + 2\eta \varepsilon_N d_M
\end{aligned} \tag{3.312}$$

and

$$a_{41} = a_{44} + \frac{1}{\alpha_{3f}} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\|^2 \frac{\delta_{3f}^2}{(2 - \delta_{3f})} W_{3f \max}^2 \tag{3.313}$$

$$a_4 = a_{41} + \frac{1}{\beta_{3g}} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\|^2 \frac{\rho_{3g}^2}{(2 - \rho_{3g})} W_{3g \max}^2 \tag{3.314}$$

$$P_1 = 2(\delta_{3f} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\| + \rho_{3g} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\|) \tag{3.315}$$

$$\begin{aligned}
P_2 &= 2(\delta_{3f} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\| W_{3f \max} \tilde{\varphi}_{3f \max} \\
&+ \rho_{3g} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\| W_{3g \max} \tilde{\varphi}_{3g \max})
\end{aligned} \tag{3.316}$$

$$P_3 = (\eta + \delta_{3f} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\|)^2 \tag{3.317}$$

$$P_4 = (\eta + \rho_{3g} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\|)^2 \quad (3.318)$$

$$\begin{aligned} P_5 = & \left(2\delta_{if} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\| + \left(\alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k) \right. \right. \\ & + \frac{((1 - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)) - \delta_{if} \|I - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k)\|)^2}{(2 - \alpha_{3f} \hat{\varphi}_{3f}(k) \hat{\varphi}_{3f}^T(k))} \Big) \\ & \times \|\hat{\varphi}_{if}(k)\|^2 \|W_{if}\|^2 \Big) + \left(2\rho_{ig} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\| + \left(\beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k) \right. \right. \\ & + \frac{((1 - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)) - \rho_{ig} \|I - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k)\|)^2}{(2 - \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k))} \Big) \\ & \times \|\hat{\varphi}_{ig}(k)\|^2 \|W_{ig}\|^2 \Big) \end{aligned} \quad (3.319)$$

$$\begin{aligned} P_6 = & \left(\alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k) + \frac{((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|)^2}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\ & + \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\| \Big) \kappa_{if} k_{v \max} + \left(\beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k) \right. \\ & + \frac{((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|)^2}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \\ & \left. \left. + \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\| \right) \kappa_{ig} k_{v \max} \right) \end{aligned} \quad (3.320)$$

$$\begin{aligned} P_7 = & \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k) + \frac{((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|)^2}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \\ & \times \kappa_{if}^2 k_{v \max}^2 + \left(\beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k) \right. \\ & + \frac{((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|)^2}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \Big) \kappa_{ig}^2 k_{v \max}^2 \end{aligned} \quad (3.321)$$

with η as given in (3.302).

Equation 3.309 can be rewritten as

$$\begin{aligned}
\Delta J \leq & -(1 - a_5) \|r(k)\|^2 + 2a_6 \|r(k)\| + a_7 - (1 - \eta - P_3) \|\bar{e}_f(k)\|^2 \\
& - (1 - \eta - P_4) \|\bar{e}_g(k)\|^2 - 2(1 - \eta - P_1) \|\bar{e}_f(k)\| \|\bar{e}_g(k)\| \\
& - \left\| \left(\sqrt{P_3} \bar{e}_f(k) + \sqrt{P_4} \bar{e}_g(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\
& - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\
& \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \Big\|^2 \\
& - \sum_{i=1}^2 (2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) \left\| \tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(k) - \frac{1}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \right. \\
& \times ((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|) (y_{ig} + B_{ig} k_v r(k)) \Big\|^2 \\
& - \text{tr}(\hat{Z}_f^T(k) C_{1f} \hat{Z}_f(k) - 2 \hat{Z}_f(k) C_{2f} \tilde{Z}_f) - \text{tr}(\hat{Z}_g^T(k) C_{1g} \hat{Z}_g(k) - 2 \hat{Z}_g(k) C_{2g} \tilde{Z}_g)
\end{aligned} \tag{3.322}$$

where $a_5 = a_2 + P_5$, $a_6 = a_3 + P_6$, $a_7 = P_7 + a_4$.

Rewriting Equation 3.322 one obtains

$$\begin{aligned}
\Delta J \leq & -(1 - a_5) \|r(k)\|^2 + 2a_6 \|r(k)\| + a_8 - (1 - \eta - P_3) \|\bar{e}_f(k)\|^2 \\
& - (1 - \eta - P_4) \|\bar{e}_g(k)\|^2 - 2(1 - \eta - P_1) \|\bar{e}_f(k)\| \|\bar{e}_g(k)\| \\
& - \left\| \left(\sqrt{P_3} \bar{e}_f(k) + \sqrt{P_4} \bar{e}_g(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\
& - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\
& \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \Big\|^2 \\
& - \sum_{i=1}^2 (2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) \left\| \tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(k) - \frac{1}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \right. \\
& \times ((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|) (y_{ig} + B_{ig} k_v r(k)) \Big\|^2 \\
& - (2C_{2f \min} - C_{1f \max}) \left[\|\tilde{Z}_f(k)\|^2 - 2 \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} \|\tilde{Z}_f(k)\| Z_{Mf} \right. \\
& \left. - 2 \frac{C_{1f \max}}{(2C_{2f \min} - C_{1f \max})} \|\tilde{Z}_g(k)\| Z_{Mg} - \frac{C_{1g \max}}{(2C_{2g \min} - C_{1g \max})} Z_{Mg}^2 \right]
\end{aligned} \tag{3.323}$$

where

$$C_{1f} = \text{diag}\left(\delta_{if}^2 \frac{1}{\alpha_{if}} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|^2\right)$$

and

$$C_{2f} = \text{diag}\left(\delta_{if} \frac{1}{\alpha_{if}} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|^2\right)$$

Similarly

$$C_{1g} = \text{diag}\left(\delta_{ig}^2 \frac{1}{\alpha_{ig}} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|^2\right)$$

and

$$C_{2g} = \text{diag}\left(\delta_{ig} \frac{1}{\alpha_{ig}} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|^2\right)$$

Completing the squares for $\tilde{Z}_f(k)$ and $\tilde{Z}_g(k)$ in (3.323) one obtains

$$\begin{aligned} \Delta J \leq & -(1 - a_5) \|r(k)\|^2 + 2a_6 \|r(k)\| + a_8 - (1 - \eta - P_3) \|\overline{e_f}(k)\|^2 \\ & - (1 - \eta - P_4) \|\overline{e_g}(k)\|^2 - 2(1 - \eta - P_1) \|\overline{e_f}(k)\| \|\overline{e_g}(k)\| \\ & - \left\| \left(\sqrt{P_3} \overline{e_f}(k) + \sqrt{P_4} \overline{e_g}(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ & - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\ & \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \Big\|^2 \\ & - \sum_{i=1}^2 (2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) \left\| \tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(k) - \frac{1}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \right. \\ & \times ((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|) (y_{ig} + B_{ig} k_v r(k)) \Big\|^2 \\ & - (2C_{2f \min} - C_{1f \max}) \left[\|\tilde{Z}_f(k)\| - \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} Z_{Mf} \right]^2 \\ & - (2C_{2g \min} - C_{1g \max}) \left[\|\tilde{Z}_f(k)\| - \frac{(C_{1g \min} + C_{2g \min})}{(2C_{2g \min} - C_{1g \max})} Z_{Mg} \right]^2 \end{aligned} \quad (3.324)$$

where

$$\begin{aligned} a_8 = a_7 + \frac{C_{1f \max}}{(2C_{2f \min} - C_{1f \max})} Z_{Mf}^2 + \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} Z_{Mf}^2 \\ + \frac{C_{1g \max}}{(2C_{2g \min} - C_{1g \max})} Z_{Mg}^2 + \frac{(C_{1g \min} + C_{2g \min})}{(2C_{2g \min} - C_{1g \max})} Z_{Mg}^2 \end{aligned} \quad (3.325)$$

All the terms in (3.324) are always negative except the first term as long as the conditions (3.295) through (3.301) hold. Since a_5 , a_6 , and a_8 are positive constants, $\Delta J \leq 0$ as long as (3.295) through (3.301) hold with

$$\|r(k)\| > \delta_{r1} \quad (3.326)$$

where

$$\delta_{r1} = \frac{1}{(1 - a_5)} \left[a_6 + \sqrt{a_6^2 + a_8(1 - a_5)} \right] \quad (3.327)$$

Similarly, completing squares for $\|r(k)\|$, $\|\tilde{Z}_g(k)\|$ using (3.323) yields

$$\begin{aligned} \Delta J \leq & -(1 - a_5) \left[\|r(k)\| - \frac{a_6}{(1 - a_5)} \right]^2 - (1 - \eta - P_3) \|\bar{e}_f(k)\|^2 \\ & - (1 - \eta - P_4) \|\bar{e}_g(k)\|^2 - 2(1 - \eta - P_1) \|\bar{e}_f(k)\| \|\bar{e}_g(k)\| \\ & - \left\| \left(\sqrt{P_3} \bar{e}_f(k) + \sqrt{P_4} \bar{e}_g(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ & - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\ & \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \Big\|^2 \\ & - \sum_{i=1}^2 (2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) \left\| \tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(k) - \frac{1}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \right. \\ & \times ((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|) (y_{ig} + B_{ig} k_v r(k)) \Big\|^2 \\ & - (2C_{2f \min} - C_{1f \max}) \left[\|\tilde{Z}_f(k)\| - \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} \|\tilde{Z}_f(k)\| Z_{Mf} - a_{10} \right]^2 \\ & - (2C_{2g \min} - C_{1g \max}) \left[\|\tilde{Z}_g(k)\| - \frac{(C_{1g \min} + C_{2g \min})}{(2C_{2g \min} - C_{1g \max})} Z_{Mg} \right]^2 \end{aligned} \quad (3.328)$$

where

$$a_{10} = \frac{(C_{1f\ min} + C_{2f\ min})^2}{(2C_{2f\ min} - C_{1f\ max})} Z_{Mf} \quad (3.329)$$

$$a_{11} = \frac{C_{1f\ max}}{(2C_{2f\ min} - C_{1f\ max})} Z_{Mf}^2 + \frac{(C_{1g\ min} + C_{2g\ min})^2}{(2C_{2g\ min} - C_{1g\ max})} Z_{Mg}^2 \quad (3.330)$$

Then $\Delta J \leq 0$ as long as (3.295) through (3.301) hold and the quadratic term for $\tilde{Z}_f(k)$ in (3.327) is positive, which is guaranteed when

$$\|\tilde{Z}_f(k)\| > \delta_f \quad (3.331)$$

where

$$\delta_f = a_{10} + \sqrt{a_{10}^2 + a_{11}} \quad (3.332)$$

Similarly completing the squares for $\|r(k)\|$, $\|\tilde{Z}_f(k)\|$ using (3.323) yields

$$\begin{aligned} \Delta J \leq & -(1 - a_5) \left[\|r(k)\| - \frac{a_6}{(1 - a_5)} \right]^2 - (1 - \eta - P_3) \|\overline{e_f}(k)\|^2 \\ & - (1 - \eta - P_4) \|\overline{e_g}(k)\|^2 - 2(1 - \eta - P_1) \|\overline{e_f}(k)\| \|\overline{e_g}(k)\| \\ & - \left\| \left(\sqrt{P_3} \overline{e_f}(k) + \sqrt{P_4} \overline{e_g}(k) \right) - (k_v r(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) \right\|^2 \\ & - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \tilde{W}_{if}^T(k) \hat{\varphi}_{if}(k) - \frac{1}{(2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\ & \times ((1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) - \delta_{if} \|I - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)\|) (y_{if} + B_{if} k_v r(k)) \left. \right\|^2 \\ & - \sum_{i=1}^2 (2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) \left\| \tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(k) - \frac{1}{(2 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k))} \right. \\ & \times ((1 - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)) - \rho_{ig} \|I - \beta_{ig} \hat{\varphi}_{ig}(k) \hat{\varphi}_{ig}^T(k)\|) (y_{ig} + B_{ig} k_v r(k)) \left. \right\|^2 \\ & - (2C_{2g\ min} - C_{1g\ max}) \left[\|\tilde{Z}_g(k)\| - \frac{(C_{1g\ min} + C_{2g\ min})}{(2C_{2g\ min} - C_{1g\ max})} \|\tilde{Z}_g(k)\| Z_{Mg} - a_{10} \right]^2 \\ & - (2C_{2f\ min} - C_{1f\ max}) \left[\|\tilde{Z}_f(k)\| - \frac{(C_{1f\ min} + C_{2f\ min})}{(2C_{2f\ min} - C_{1f\ max})} Z_{Mf} \right]^2 \end{aligned} \quad (3.333)$$

where

$$a_{10} = \frac{(C_{1g\min} + C_{2g\min})^2}{(2C_{2g\min} - C_{1g\max})} Z_{Mg} \quad (3.334)$$

$$a_{11} = \frac{C_{1g\max}}{(2C_{2g\min} - C_{1g\max})} Z_{Mg}^2 + \frac{(C_{1f\min} + C_{2f\min})^2}{(2C_{2f\min} - C_{1f\max})} Z_{Mf}^2 \quad (3.335)$$

Then $\Delta J \leq 0$ as long as (3.295) through (3.301) hold and the quadratic term for $\tilde{Z}_g(k)$ in (3.333) is positive, which is guaranteed when

$$\|\tilde{Z}_g(k)\| > \delta_{g1} \quad (3.336)$$

where

$$\delta_{g1} = a_{10} + \sqrt{a_{10}^2 + a_{11}} \quad (3.337)$$

We have shown upper bounds for the tracking error and the NN weight estimation errors for this region for all $\|u_c(k)\| \leq s$.

Region II: $\|\hat{g}(x)\| \leq g$ and $\|u_c(k)\| > s$

The filtered tracking error dynamics can be written for this region as (3.267). Now using the Lyapunov function (3.304a) and the first difference (3.304b) after substituting for $g(x)u_d(k)$ from (3.197) in (3.304b) and manipulating accordingly, one can obtain

$$\begin{aligned} \Delta J = & -(1 - b_0)\|r(k)\|^2 + 2b_1\|r(k)\| + b_2 - (1 - \eta) \\ & \times \left\| \overline{e_f}(k) - \frac{\eta}{1 - \eta}(k_v r(k) + \overline{g(x)u_d(k)} + \varepsilon(k) + d(k)) \right\|^2 \\ & - \sum_{i=1}^2 (2 - \alpha_{if}\hat{\varphi}_{if}(k)\hat{\varphi}_{if}^T(k)) \left\| \hat{y}_{if}(k) - \frac{(1 - \alpha_{if}\hat{\varphi}_{if}(k)\varphi_{if}^T(k))}{(1 - \alpha_{if}\hat{\varphi}_{if}(k)\hat{\varphi}_{if}^T(k))} \right. \\ & \times (y_{if} + B_{if}k_v r(k)) \left. \right\|^2 + b_3\|r(k)\|^2 + b_4\|r(k)\| + b_5 \\ & - \sum_{i=1}^2 \frac{1}{\alpha_{if}} \|I - \alpha_{if}\varphi_{if}(k)\varphi_{if}^T(k)\|^2 [\delta_{if}^2 \hat{W}_{if}^T(k) \hat{W}_{if}(k) + 2\delta_{if} \tilde{W}_{if} \hat{W}_{if}(k)] \end{aligned} \quad (3.338)$$

where b_0, b_1, b_2, b_3, b_4 , and b_5 are computable constants (Jagannathan 1996e), with η given by (3.303).

$$\begin{aligned} \Delta J = & -(1 - b_6) \|r(k)\|^2 + 2b_7 \|r(k)\| + b_8 - (1 - \eta) \left\| \overline{e_f}(k) - \frac{\eta}{1 - \eta} (k_v r(k) \right. \\ & \left. + \overline{g(x) u_d(k)} + \varepsilon(k) + d(k)) \right\|^2 - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \\ & \times \left\| \hat{y}_{if}(k) - \frac{(1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))}{(1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} (y_{if} + B_{if} k_v r(k)) \right\|^2 \\ & - (2C_{2f \min} - C_{1f \max}) \left[\|\tilde{Z}_f(k)\|^2 - 2 \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} \|\tilde{Z}_f(k)\| Z_{Mf} \right. \\ & \left. - \frac{C_{1f \max}}{(2C_{2f \min} - C_{1f \max})} Z_{Mf}^2 \right] \end{aligned} \quad (3.339)$$

with

$$b_6 = b_0 + b_3, \quad b_7 = b_2 + b_4, \quad b_8 = b_2 + b_5 \quad (3.340)$$

Completing the squares for $\|Z_f(k)\|$ using (3.339) to obtain

$$\begin{aligned} \Delta J = & -(1 - b_6) \|r(k)\|^2 + 2b_7 \|r(k)\| + b_9 - (1 - \eta) \\ & \times \left\| \overline{e_f}(k) - \frac{\eta}{1 - \eta} (k_v r(k) + \overline{g(x) u_d(k)} + \varepsilon(k) + d(k)) \right\|^2 \\ & - \sum_{i=1}^2 (2 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k)) \left\| \hat{y}_{if}(k) - \frac{(1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))}{(1 - \alpha_{if} \hat{\varphi}_{if}(k) \hat{\varphi}_{if}^T(k))} \right. \\ & \times (y_{if} + B_{if} k_v r(k)) \left. \right\|^2 - (2C_{2f \min} - C_{1f \max}) \left[\|\tilde{Z}_f(k)\|^2 \right. \\ & \left. - 2 \frac{(C_{1f \min} + C_{2f \min})}{(2C_{2f \min} - C_{1f \max})} \|\tilde{Z}_f(k)\| Z_{Mf} - \frac{C_{1f \max}}{(2C_{2f \min} - C_{1f \max})} Z_{Mf}^2 \right] \end{aligned} \quad (3.341)$$

The terms in (3.341) are always negative as long as the conditions (3.295) through (3.301) hold. Since b_6, b_7, b_9 are positive constants, $\Delta J \leq 0$ as long

as

$$\|r(k)\| > \delta_{r2} \quad (3.342)$$

where

$$\delta_{r2} = \frac{1}{(1 - b_6)} \left[b_7 + \sqrt{b_7^2 + b_9(1 - b_6)} \right] \quad (3.343)$$

with

$$b_9 = b_8 + \frac{(2C_{1f\max} + C_{2f\max})}{(2C_{2f\min} - C_{1f\max})} Z_{Mf}^2 \quad (3.344)$$

Similarly, completing the squares for $\|r(k)\|$ in (3.339) one obtains

$$\begin{aligned} \Delta J = & -(1 - b_6) \left[\left\| r(k) - \frac{b_7}{(1 - b_6)} \right\|^2 - (1 - \eta) \left\| \bar{e}_f(k) - \frac{\eta}{1 - \eta} (k_v r(k) \right. \right. \\ & \left. \left. + \overline{g(x)u_d(k)} + \varepsilon(k) + d(k)) \right\|^2 - \sum_{i=1}^2 (2 - \alpha_{if} \varphi_{if}(k) \varphi_{if}^T(k)) \right\| \hat{y}_{if}(k) \\ & - \frac{(1 - \alpha_{if} \varphi_{if}(k) \varphi_{if}^T(k))}{(1 - \alpha_{if} \varphi_{if}(k) \varphi_{if}^T(k))} (y_{if} + B_{if} k_v r(k)) \right\|^2 - (2C_{2f\min} - C_{1f\max}) \left[\|\tilde{Z}_f(k)\|^2 \right. \\ & \left. - 2 \frac{(C_{1f\min} + C_{2f\min})}{(2C_{2f\min} - C_{1f\max})} \|\tilde{Z}_f(k)\| Z_{Mf} - \frac{C_{1f\max}}{(2C_{2f\min} - C_{1f\max})} Z_{Mf}^2 \right. \\ & \left. - \frac{1}{(2C_{2f\min} - C_{1f\max})} \left(Z_M^2 + b_8 + \frac{b_7^2}{1 - b_6} \right) \right] \end{aligned} \quad (3.345)$$

The terms in (3.348) are always negative as long as the conditions (3.295) through (3.301) hold. Since b_6 , b_7 , b_9 are positive constants, $\Delta J \leq 0$ as long as

$$\|\tilde{Z}_f\| > \delta_{f2} \quad (3.346)$$

with

$$\delta_{f2} = b_9 + \sqrt{b_9^2 + b_{10}} \quad (3.347)$$

with

$$b_9 = \frac{(2C_{1f\ min} + C_{2f\ min})}{(2C_{2f\ min} - C_{1f\ max})} Z_{Mf}^2 \quad (3.348)$$

and

$$b_{10} = \frac{C_{1f\ max}}{(2C_{2f\ min} - C_{1f\ max})} Z_{Mf}^2 - \frac{1}{(2C_{2f\ min} - C_{1f\ max})} \left(Z_M^2 + b_8 + \frac{b_7^2}{(1-b_6)} \right) \quad (3.349)$$

One has $|\sum_{k=k_0}^{\infty} \Delta J(k)| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (3.295) through (3.301) hold. The definitions of J and inequalities (3.342) and (3.346) imply that every initial condition in the set X will evolve entirely within X . Thus according to the standard Lyapunov extension, it can be concluded that the tracking error $r(k)$ and the error in weight updates are UUB.

Reprise: Combining the results from Region I and II, one can readily set

$$\delta_r = \max(\delta_{r1}, \delta_{r2}), \quad \delta_f = \max(\delta_{f1}, \delta_{f2}), \quad \delta_g$$

Thus, for both regions it follows that if $\|r(k)\| > \delta_r$, then $\Delta J \leq 0$ and $u(k)$ is bounded. Let us denote $(\|r(k)\|, \|\tilde{W}_f(k)\|, \|\tilde{W}_g(k)\|)$ by the new coordinate variables (ξ_1, ξ_2, ξ_3) . Define the region

$$\Xi : \xi \mid \xi_1 < \delta_r, \quad \xi_2 < \delta_f, \quad \xi_3 < \delta_g$$

Then there exists an open set

$$\Omega : \xi \mid \xi_1 < \overline{\delta_r}, \quad \xi_2 < \overline{\delta_f}, \quad \xi_3 < \overline{\delta_g}$$

where $\overline{\delta_i} > \delta_i$ implying that $\Xi \subset \Omega$. In other words, it was shown that whenever $\xi_i > \delta_i$ then $J(\xi)$ will not increase and will remain in the region Ω which is an invariant set. Therefore all the signals in the closed-loop system remain UUB. This concludes the proof.

Example 3.4.2 (Control Using NN Tuning Not Requiring PE): For Example 3.4.1 the response of the NN controller with the improved weight tuning (3.291) through (3.294) and projection algorithm is presented in Figure 3.22. The design parameters Γ_i ; $i = 1, 2, 3$ and ρ_i ; $i = 1, 2, 3$ are selected as 0.01. Note that with the improved weight tuning, the output of the NN remains bounded because the weights are guaranteed to remain bounded without the

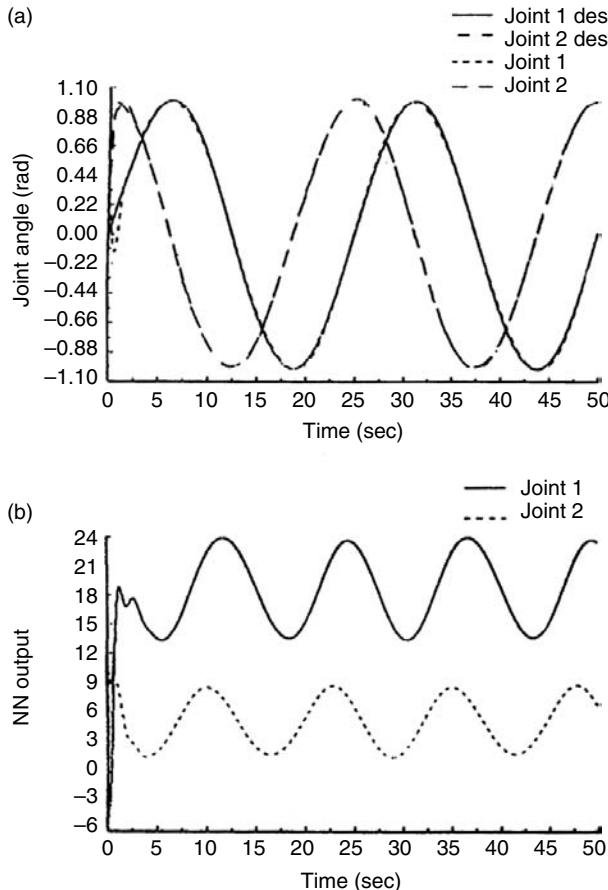


FIGURE 3.22 Response of the NN controller with improved weight-tuning and projection algorithm. (a) Actual and desired joint angles. (b) NN outputs.

necessity of PE. Figure 3.23 can be used to study the contribution of the neural network as it shows the response of the PD controller with no NN. It is clear that the addition of the NN makes a significant improvement in the tracking performance.

Example 3.4.3 (NN Control of Discrete-Time Nonlinear System): Consider the first order MIMO discrete-time nonlinear system described by

$$X(k+1) = F(X) + G(X)U(k) \quad (3.350)$$

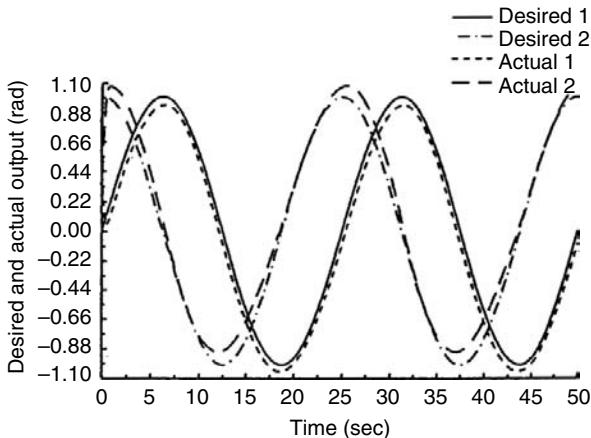


FIGURE 3.23 Response of the PD controller.

where

$$X(k) = [x_1(k), x_2(k)]^T$$

$$F(X) = \begin{bmatrix} \frac{x_2(k)}{1 + x_1^2(k)} \\ \frac{x_2(k)}{1 + x_1^2(k)} \end{bmatrix}$$

$$G(X) = \begin{bmatrix} \frac{1}{1 + x_1^2(k)} & 0 \\ 0 & \frac{1}{1 + x_1^2(k)} \end{bmatrix}$$

and the input is given by $U(k) = [u_1(k), u_2(k)]^T$. The objective is to track a periodic step input of magnitude two units with a period of 30 sec. The elements of the diagonal matrix were chosen as $k_v = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ and a sampling interval of 10 msec was considered. Multilayer NN were selected with 12 hidden-layer nodes. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for the plant were chosen as $[1, -1]^T$. The weights were initialized to zero for $F(\cdot)$ and to an identity matrix for $G(\cdot)$ with an initial threshold value of 3.0. The design parameters $\Gamma_i; i = 1, 2, 3$ and $\rho_i; i = 1, 2, 3$ were selected to be 0.01. No learning is performed initially to train the networks. The design parameters for

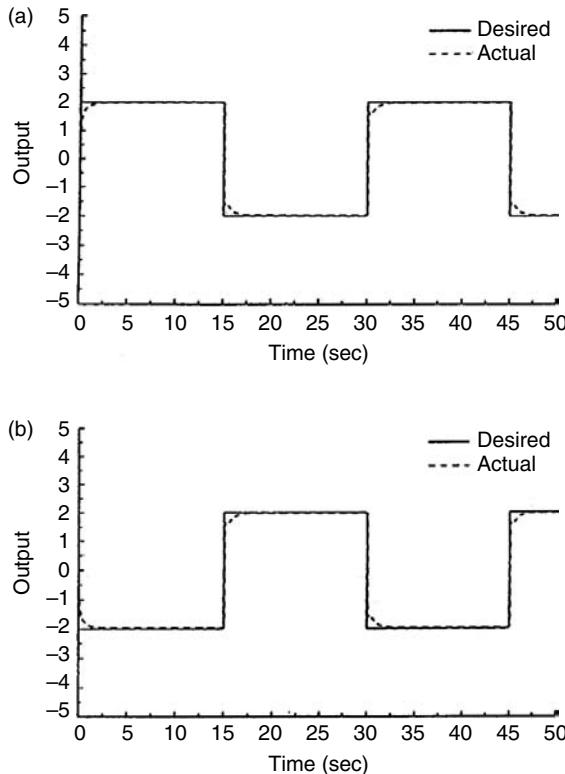


FIGURE 3.24 Response of the NN controller with improved weight-tuning and projection algorithm. (a) Desired and actual state 1. (b) Desired and actual state 2.

the projection algorithm were selected to be $\xi_3 = 0.5$, $\xi_i = 1.0$; $i = 1, 2$ with $\zeta_i = 0.001$; $\forall i = 1, 2, 3$ for both NN.

In this example, only results using the improved weight tuning are presented. The response of the controller with the improved weight tuning (3.291) through (3.294) is shown in Figure 3.24. Note from Figure 3.24 as expected, the performance of the controller is extremely impressive.

Let us consider the case when a bounded disturbance given by

$$w(k) = \begin{cases} 0.0, & 0 \leq kT_m < 12 \\ 0.1, & kT_m \geq 12 \end{cases} \quad (3.351)$$

is acting on the plant at time instant k . Figure 3.25 presents the tracking response of NN controllers with the improved weight tuning and the projection algorithm.

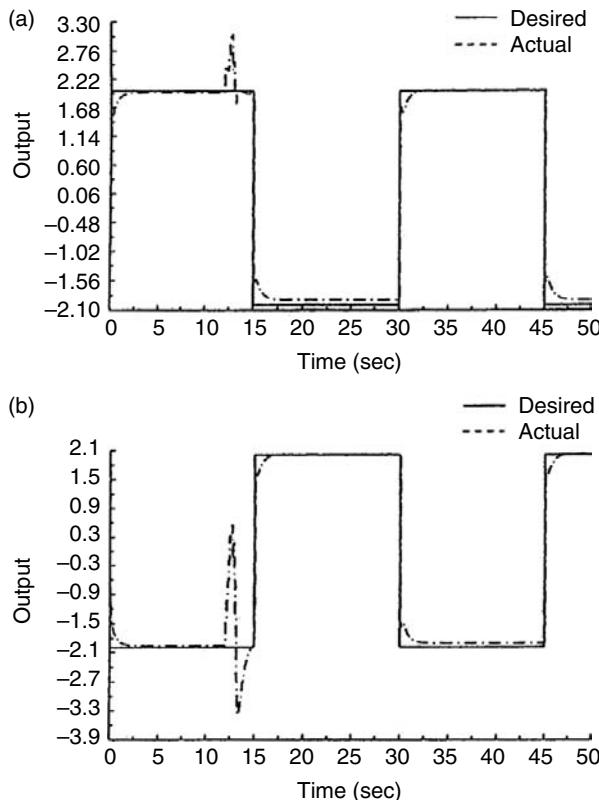


FIGURE 3.25 Response of the NN controller with improved weight-tuning in the presence of bounded disturbances. (a) Desired and actual state 1. (b) Desired and actual state 2.

The magnitude of the disturbance can be increased but the value should be bounded. It can be seen from the figure that the bounded disturbance induces bounded tracking errors at the output of the plant. From the results, it can be inferred that the bounds presented and the theoretical claims were justified through simulation studies both in continuous- and discrete-time.

3.5 PASSIVITY PROPERTIES OF THE NN

In this section, an interesting property of the NN controller is shown. Namely, the NN controller makes the closed-loop system passive. The practical importance of this is that additional unknown bounded disturbances do not destroy the stability and tracking of the system. Passivity was discussed in Chapter 2.

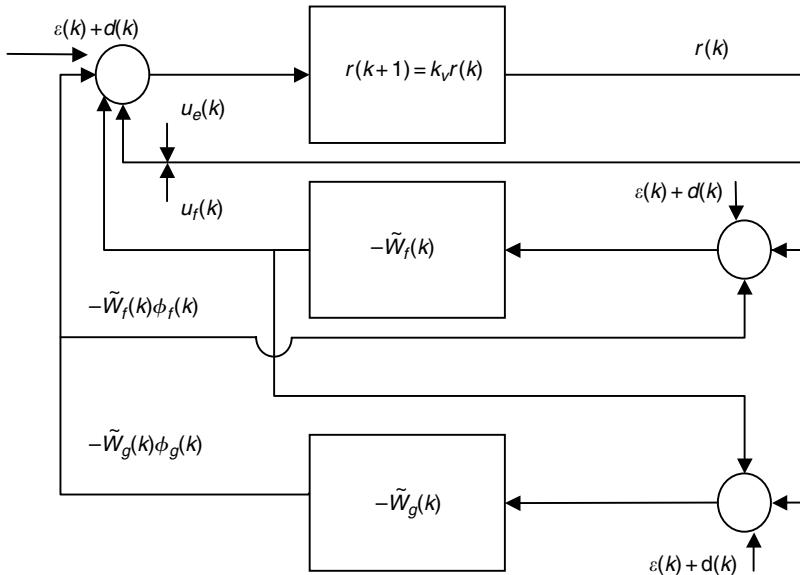


FIGURE 3.26 The NN closed-loop system using a one-layer NN.

Note that the NN used in the controllers in this chapter are feedforward NN with no dynamics. However, tuning them online turns them into dynamic systems, so that passivity properties can be defined.

The closed-loop error system (3.168) shown in Figure 3.26 uses a one-layer NN; note that the NN is now in the standard feedback configuration as opposed to the NN controller in Figure 3.19, which has both feedback and feedforward connections. Passivity is essential in a closed-loop system as it guarantees the boundedness of the signals and consequently suitable performance even in the presence of unforeseen bounded disturbances. This equates to robustness of the closed-loop system. Therefore, in this section the passivity properties of the NN and the closed-loop system are explored for various weight-tuning algorithms.

3.5.1 PASSIVITY PROPERTIES OF THE TRACKING ERROR SYSTEM

Even though the closed-loop error system (3.168) is SSP, the closed-loop system is not passive unless the weight-update laws guarantee the passivity of the lower block in Figure 3.26. It is usually difficult to demonstrate that the error in weight updates is passive. However, in the next section it is shown that the delta-rule-based weight-tuning algorithm (3.174) and (3.175) for a one-layer NN yields a passive net.

3.5.2 PASSIVITY PROPERTIES OF ONE-LAYER NN CONTROLLERS

It is shown in Jagannathan (1996d) that the one-layer NN tuning algorithms in Theorem 3.3.1, where PE is required make the NN passive, but the tuning algorithms in Theorem 3.3.2, where the PE is not required make the NN SSP. The implications for the closed-loop passivity using the NN controller in Table 3.4 and Table 3.5 are then discussed.

The next result details the modified tuning algorithms in Table 3.5 that yield a stronger passivity property for the NN.

Theorem 3.5.1 (*One-Layer NN Passivity for Tuning Algorithms, No PE*): The modified weight tuning algorithms (3.233) and (3.234) make the map from $(k_v r(k) + \bar{e}_g(k) + g(x)u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.233) and $(k_v r(k) + \bar{e}_f(k) + g(x)u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.234), to $-\tilde{W}_f^T(k)\varphi_f(k)$ and $-\tilde{W}_g^T(k)\varphi_g(k)$ SSP maps.

Proof: See Jagannathan and Lewis (1996).

It has been shown that the filtered tracking error system (3.168) in Figure 3.26 is state strict passive, while the NN weight error block is passive using the tuning rules in Table 3.4. Thus, using standard results (Landau 1979) it can be concluded that the closed-loop system is passive. Therefore according to the passivity theorem one can conclude that the input/output signals of each block are bounded as long as the disturbances are bounded. Though passive, the closed-loop system is not SSP so this does not yield boundedness of the internal states of the lower blocks (e.g., $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$) unless PE holds for the case of one-layer NN.

On the other hand, the enhanced tuning rules of Table 3.5 yield an SSP weight-tuning block in the figure so that the closed-loop system is SSP. Thus, the internal states of the lower blocks (e.g., $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$) are bounded even if PE does not hold. Thus, the modified tuning algorithms guarantee SSP of the weight tuning blocks, so the closed-loop system is SSP. Therefore inherent stability can be guaranteed even in the absence of PE.

3.5.3 PASSIVITY PROPERTIES OF MULTILAYER NN CONTROLLERS

It is shown here that the multilayer NN tuning algorithms in Theorem 3.4.1, where PE is required make the NN passive, but the tuning algorithms in Theorem 3.4.2, where PE is not required to make the NN to be SSP. The implications for the closed-loop passivity using the NN controller in Table 3.6 and Table 3.7 are then discussed.

The next result details the passivity properties engendered by the tuning rules in Table 3.6.

Theorem 3.5.2 (*Multilayer NN Passivity, Tuning Algorithms with PE*):

The weight-tuning algorithms (3.283) and (3.284) make the maps from $(k_v r(k) + \bar{e}_g(k) + g(x)u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.283) and $(k_v r(k) + \bar{e}_f(k) + g(x)u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.284) to $-\tilde{W}_{3f}^T(k)\hat{\varphi}_{3f}(k)$ and $-\tilde{W}_{3g}^T(k)\hat{\varphi}_{3g}(x(k))$ passive maps.

The weight-tuning algorithms for the hidden layers (3.283) and (3.285) make the maps from $y_{if}(k) + B_{if}k_v r(k)$ for the case of (3.283) and $y_{ig}(k) + B_{ig}k_v r(k)$ for the case of (3.285) to $\tilde{W}_{if}^T(k)\hat{\varphi}_{if}(x(k))$ and $\tilde{W}_{ig}^T(k)\hat{\varphi}_{ig}(x(k))$ passive maps.

Proof: Define the Lyapunov function candidate

$$J = \frac{1}{\alpha_{3f}} \text{tr}[\tilde{W}_{3f}(k)\tilde{W}_{3f}^T(k)] \quad (3.352)$$

whose first difference is given by

$$J = \frac{1}{\alpha_{3f}} \text{tr}[\tilde{W}_{3f}(k+1)\tilde{W}_{3f}^T(k+1) - \tilde{W}_{3f}(k)\tilde{W}_{3f}^T(k)] \quad (3.353)$$

Substituting the weight-update law (3.284) in (3.353) yields

$$\begin{aligned} \Delta J = & (2 - \alpha_{3f}\hat{\varphi}_{3f}^T(k)\varphi_{3f}(k))(-\tilde{W}_{3f}(k)\hat{\varphi}_{3f}(k))^T(-\tilde{W}_{3f}^T(k)\hat{\varphi}_{3f}(k)) \\ & + \alpha_{3f}\hat{\varphi}_{3f}(k)\hat{\varphi}_{3f}^T(k)(k_v r(k) + \bar{e}_g(k) + g(x)u_d(k) + \varepsilon(k) \\ & + d(k))^T(k_v r(k) + \bar{e}_g(k) + g(x)u_d(k) + \varepsilon(k) + d(k)) \\ & + 2(1 - \alpha_{3f}\hat{\varphi}_{3f}^T(k)\varphi_{3f}(k))(-\tilde{W}_{3f}(k)\hat{\varphi}_{3f}(k))(k_v r(k) + \bar{e}_g(k) + g(x)u_d(k) \\ & + \varepsilon(k) + d(k)) \end{aligned} \quad (3.354)$$

Note (3.353) is in the power form (2.33) defined in Chapter 2 as long as conditions (3.290) through (3.295) hold. This in turn guarantees the passivity of the weight-tuning mechanism (3.284).

Similarly one can also prove that the error in weight updates presented in (3.286) is passive as long as the PE condition is satisfied. In fact, if one chooses the first difference as

$$J = \frac{1}{\beta_{3g}} \text{tr}[\tilde{W}_{3g}(k+1)\tilde{W}_{3g}^T(k+1) - \tilde{W}_{3g}(k)\tilde{W}_{3g}^T(k)] \quad (3.355)$$

Using the error in update law (3.286) and simplifying one obtains

$$\begin{aligned}\Delta J = & (2 - \beta_{3g} \hat{\varphi}_{3g}^T(k) \varphi_{3g}(k))(-\tilde{W}_{3g}(k) \hat{\varphi}_{3g}(k))^T (-\tilde{W}_{3g}^T(k) \hat{\varphi}_{3g}(k)) \\ & + \beta_{3g} \hat{\varphi}_{3g}(k) \hat{\varphi}_{3g}^T(k) (k_v r(k) + \bar{e}_f(k) + g(x) u_d(k) + \varepsilon(k) + d(k))^T (k_v r(k) \\ & + \bar{e}_f(k) + g(x) u_d(k) + \varepsilon(k) + d(k)) + 2(1 - \beta_{3g} \hat{\varphi}_{3g}^T(k) \varphi_{3g}(k)) \\ & \times (-\tilde{W}_{3g}(k) \hat{\varphi}_{3g}(k)) (k_v r(k) + \bar{e}_f(k) + g(x) u_d(k) + \varepsilon(k) + d(k))\end{aligned}\quad (3.356)$$

Similarly it can be shown that the hidden layer updates yield passive NN.

The next result shows that the modified tuning algorithms in Table 3.7 yield a stronger passivity property for the NN. The proof is an extension of the previous one.

Theorem 3.5.3 (Multilayer NN Passivity for Algorithms without PE): The weight tuning algorithms (3.292) and (3.294) make the maps from $(k_v r(k) + \bar{e}_g(k) + g(x) u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.292) and $(k_v r(k) + \bar{e}_f(k) + g(x) u_d(k) + \varepsilon(k) + d(k))$ for the case of (3.294) to $-\tilde{W}_{3f}^T(k) \hat{\varphi}_{3g}(k)$ and $-\tilde{W}_{3g}^T(k) \hat{\varphi}_{3g}(k)$ passive maps.

The weight tuning algorithms for the hidden layers (3.291) and (3.293) make the maps from $y_{if}(k) + B_{if} k_v r(k)$ for the case of (3.291) and $y_{ig}(k) + B_{ig} k_v r(k)$ for the case of (3.293), to $\tilde{W}_{if}^T(k) \hat{\varphi}_{if}(x(k))$ and $\tilde{W}_{ig}^T(k) \hat{\varphi}_{ig}(x(k))$ passive maps.

It has been shown that the filtered tracking error system (3.168) in Figure 3.26 and (3.169) is SSP, while the NN weight error block is passive using the tuning rules in Table 3.6. Thus, it can be concluded that the closed-loop system is passive. Therefore, according to the passivity theorem, one can conclude that the inputs/output signals of each block are bounded as long as the disturbances are bounded. Though passive, the closed-loop system is not SSP so this does not yield boundedness of the internal states of the lower blocks (e.g., $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$) unless PE holds.

On the other hand, the enhanced tuning rules of Table 3.7 yield an SSP weight-tuning block in the figure so that the closed-loop system is SSP. Thus, the internal states of the lower blocks (e.g., $\tilde{W}_f(k)$ and $\tilde{W}_g(k)$) are bounded even if PE does not hold. Thus, the modified tuning algorithms guarantee SSP of the weight-tuning blocks, so that the closed-loop system is SSP. Therefore, internal stability can be guaranteed even in the absence of PE. Similar analysis can be extended to multilayer case as well.

3.6 CONCLUSIONS

A family of one-layer and multilayer NN controllers has been developed for the control of a class of nonlinear dynamical systems. The NN has a structure derived from passivity/tracking error notions, and offers guaranteed performance. Delta-rule-based tuning has been shown to yield a passive NN, so that it performs well under ideal conditions of no parameter or functional reconstruction errors, unmodeled dynamics, bounded disturbances, and no uncertainties. In addition, it has been found that the adaptation gain in the standard delta-rule-based parameter update algorithms must decrease with increasing number of hidden-layer neurons, so that adaptation slows down.

In order to overcome the above deficiencies, a family of improved weight-tuning algorithms has been derived. The improved weight-tuning paradigms consist of a delta-rule-based update term plus a correction term similar to the ε -modification approach in the case of continuous-time adaptive control. The improved weight-tuning algorithms make the NN to be SSP so that bounded weight estimates are guaranteed in practical nonideal situations. Furthermore, the adaptation gain is modified to obtain a projection algorithm so that the adaptation rate is independent of the number of hidden-layer neurons. Simulation results in discrete-time have been presented in order to illustrate the performance of the controller even in the presence of bounded disturbances. Finally, this section has introduced a comprehensive theory in the development of adaptive NN control schemes for discrete-time systems based on Lyapunov analysis.

In the first few sections of this chapter, we showed how to design NN controllers that use discrete-time updates for a class of nonlinear systems and for Brunovsky form systems having known control influence coefficient. If one samples a continuous-time system, the discrete-time system is of the form $x(k+1) = f(x(k)) + g(x(k))u(k)$, with $f(x(k)), g(x(k))$ both unknown. In the later sections of this chapter, we demonstrated how to use two NN to estimate both $f(x(k)), g(x(k))$. This causes great problems, since to keep the control signals bounded we have to guarantee that the NN estimate for $g(x(k))$ never goes to zero. This was accomplished using a switching sort of tuning law for the NN that estimates $g(x(k))$. Two families of controllers were derived — one using linear-in-the-parameter NN and another using multilayer NN. Passivity properties of the NN controllers are covered.

REFERENCES

- Åström, K.J. and Wittenmark, B., *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.
Chen, F.-C. and Khalil, H.K., Adaptive control of nonlinear systems using neural networks, *Int. J. Contr.*, 55, 1299–1317, 1992.

- Chen, F.-C., and Khalil, H.K., Adaptive control of nonlinear discrete-time systems using neural networks, *IEEE Trans. Autom. Contr.*, 40, 791–801, 1995.
- Commuri, S. and Lewis, F.L., CMAC neural networks for control of nonlinear dynamical systems: structure, stability and passivity, *Proceedings of IEEE International Symposium on Intelligent Control*, pp. 123–129, Monterey, 1995.
- Cybenko, G., Approximations by superpositions of sigmoidal activation function, *Math. Contr. Signals, Syst.*, 2, 303–314, 1989.
- Goodwin, G.C. and Sin, K.S., *Adaptive Filtering, Prediction and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- He, P. and Jagannathan, S., Discrete-time neural network output feedback control of strict feedback systems, *Proceedings of the American Controls Conference*, Boston, MA, pp. 2439–2444, 2004.
- Ioannou, P. and Kokotovic, P., *Adaptive Systems with Reduced Models*. Springer-Verlag, New York, 1983.
- Jagannathan, S., *Intelligent Control of Nonlinear Dynamical Systems Using Neural Networks*, Ph.D. Dissertation, University of Texas at Arlington, Arlington, TX, 1994.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Automat. Control*, 41, 1693–1699, 1996a.
- Jagannathan, S. and Lewis, F.L., Multilayer discrete-time neural net controller with guaranteed performance, *IEEE Trans. Neural Netw.*, 7, 107–130, 1996b.
- Jagannathan, S. and Lewis, F.L., Robust implicit self-tuning regulator: convergence and stability, *Automatica*, 32, 1629–1644, 1996c.
- Jagannathan, S., Discrete-time adaptive control of feedback linearizable nonlinear systems, *Proceedings of the IEEE Conference on Decision and Control*, pp. 4747–4751, Kobe, Japan, 1996d.
- Jagannathan, S., Adaptive control of unknown feedback linearizable systems in discrete-time using neural networks, *Proceedings of the IEEE Conference on Robotics and Automation*, Minneapolis, Minnesota, vol. 1, pp. 258–263, 1996e.
- Kanellakopoulos, I., A discrete-time adaptive nonlinear system, *IEEE Trans. Autom. Contr.*, AC-39, 2362–2364, 1994.
- Kanellakopoulos, I., Kokotovic, P.V., and Morse, A.S., Systematic design of adaptive controllers for feedback linearizable systems, *IEEE Trans. Autom. Contr.*, 36, 1241–1253, 1991.
- Landau, I.D., *Adaptive Control: The Model Reference Approach*, Marcel Dekker, New York, 1979.
- Lewis, F.L., Jagannathan, S., and Yesiderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, London, 1999.
- Lewis, F.L., Liu, K., and Yesilderek, A., Multilayer neural robot controller with guaranteed performance, *IEEE Trans. Neural Netw.*, 6, 703–715, 1995.
- Lin, Y. and Narendra, K.S., A new error model for adaptive systems, *IEEE Trans. Autom. Contr.*, AC-25, 1980.
- Liu, C.C. and Chen, F., Adaptive control of nonlinear continuous systems using neural networks — general degree and relative degree and MIMO cases, *Int. J. Contr.*, 58, 317–335, 1991.

- Miller III, W.T., Sutton, R.S., and Werbos, P.J., *Neural Networks for Control*, MIT Press, Cambridge, 1991.
- Mpitlos, G.J. and Burton, Jr., R.M., Convergence and divergence in neural networks: processing of chaos and biological analogy, *Neural Netw.*, 5, 605–625, 1992.
- Narendra, K.S. and Annaswamy, A.M., A new adaptive law for robust adaptation without persistent excitation, *IEEE Trans. Autom. Contr.*, 32, 134–145, 1987.
- Narendra, K.S. and Annaswamy, A.M., *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Narendra, K.S. and Parthasarathy, K.S., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, 4–27, 1990.
- Polycarpou, M.M. and Ioannou, P.A., Identification and control using neural network models: design and stability analysis, *Department of Electrical Engineering*, Tech Report. 91-09-01, September 1991.
- Rovithakis, G.A. and Christodoulou, M.C., Adaptive control of unknown plants using dynamical neural networks, *IEEE Trans. Neural Netw.*, 24, 400–411, 1994.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation, in *Readings in Machine Learning*, J.W. Shavlik, Ed., Morgan Kaufmann, San Mateo, pp. 115–137, 1990.
- Sadegh, N., A perceptron network for functional identification and control of nonlinear systems, *IEEE Trans. Neural Netw.*, 4, 982–988, 1993.
- Sanner, R.M. and Slotine, J.-E., Gaussian networks for direct adaptive control, *IEEE Trans. Neural Netw.*, 3, 837–863, 1992.
- Sira-Ramirez, H.J. and Zak, S.H., The adaptation of perceptrons with applications to inverse dynamics identification of unknown dynamic systems, *IEEE Trans. Syst., Man, Cybern.*, 21, 534–543, 1991.
- Slotine, J.J. and Li, W., *Applied Nonlinear Control*, Prentice Hall Inc., Englewood Cliffs, NJ, 1991.
- Sontag, E., Feedback stabilization using two-hidden-layer-nets, *IEEE Trans. Neural Netw.*, 3, 981–990, 1992.
- Sussman, H.J., Uniqueness of the weights for minimal feedforward nets with given input–output map, *Neural Netw.*, 5, 589–593, 1992.
- Werbos, P.J., *Beyond Regression: New Tools for Prediction and Analysis in the Behavior Sciences*, Ph.D. Thesis, Committee on Applied Mathematics. Harvard University, 1974.
- Werbos, P.J., Back propagation: past and future, *Proceedings of 1988 International Conference on Neural Nets*, Washington, DC, vol. 1, pp. 1343–1353, 1989.
- White, D.A. and Sofage, D.A., Eds., *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, 1992.
- Yesildirek, A. and Lewis, F.L., Feedback linearization using neural networks, *Automatica*, 31, 1659–1664, 1995.
- Zhang, T., Hang, C.C., and Ge, S.S., Robust adaptive control for general nonlinear systems using multilayer neural networks, *Preprint*, 1998.

PROBLEMS

SECTION 3.1

3.1-1: *One-layer NN.* Consider the system described by

$$x(k+1) = f(x(k), x(k-1)) + 10u(k)$$

where $f(x(k), x(k-1)) = (x(k)x(k-1)[x(k) + 3.0])/(1 + x^2(k) + x(k-1))$. Design a one-layer NN controller with or without learning phase and by using the developed delta-rule-based weight-tuning algorithm and appropriately choosing the adaptation gains. Repeat the problem by using the modified weight-update weight-tuning methods.

3.1-2: *One-layer NN.* For the system described by

$$x(k+1) = f(x(k), x(k-1)) + 2u(k)$$

where $f(x(k), x(k-1)) = (x(k))/(1 + x(k)) + u^3(k)$. Design a one-layer NN controller with or without learning phase and by using the developed delta-rule-based weight-tuning algorithm and appropriately choosing the adaptation gains. Repeat the problem by using the modified weight-update weight-tuning methods.

3.1-3: *Stability and convergence for an n-layer NN using Algorithm (a).* Assume the hypotheses presented for three-layer NN and use the weight updates presented in (3.62) to (3.66) and show the stability and boundedness of tracking error and error in weight updates.

3.1-4: *Stability and convergence for an n-layer NN using Algorithm (b).* Assume the hypotheses presented for the three-layer NN and use the weight updates presented in (3.62) to (3.1.65) with projection algorithm and show the stability and boundedness of tracking error and error in weight updates.

3.1-5: *Three-layer NN continuous-time simulation example using Algorithm (a).* Perform a MATLAB® simulation for Example 3.1.1 using a multilayer NN with delta-rule-based weight tuning.

3.1-6: *Three-layer NN using Algorithm (b).* Perform a MATLAB simulation for systems described in (3.60) and (3.61) using a multilayer NN with delta-rule-based weight tuning.

3.1-7: *Three-layer NN discrete-time simulation example using Algorithm (a).* Perform a MATLAB simulation for Example 3.1.4 using a multilayer NN with delta-rule-based weight tuning.

3.1-8: *Delta rule slows down using Algorithm (a).* Perform a MATLAB simulation using a large value of adaptation gains for Example 3.1.1.

3.1-9: *n-layer NN for control.* Perform a MATLAB simulation using a n -layer NN and with Algorithms (a) and (b) for the Example 3.1.1. Show the advantage of adding more layers by picking less number of hidden-layer neurons and more than three layers. Use both delta-rule and projection algorithm.

3.1-10: *Stability and convergence of an n-layer NN with modified weight tuning.* Show for a n -layer NN and use the modified weight tuning (use both Algorithm[a] and [b]) to show the boundedness of the tracking error and the weight estimates.

3.1-11: *Example (3.1.1) using modified weight tuning.* Perform a MATLAB simulation for the Example 3.1.1 using a three-layer NN and with Algorithm (a).

3.1-12: *Discrete-time simulation example using modified weight tuning.* Perform a MATLAB for the Example 3.1.4 using a three-layer NN and with Algorithm (a).

3.1-13: *Three-layer NN using Algorithm (b).* Perform a MATLAB simulation for systems described in (3.60) and (3.61) using a multilayer NN with improved weight tuning.

3.1-14: *n-layer NN and modified tuning methods.* Repeat Example 3.1.1 and Example 3.1.4 using an n -layer network (choose more than three layers) with fewer number of hidden-layer.

3.1-15: *Passivity properties for an n-layer NN.* Show the passivity properties of the input and hidden layers for an n -layer NN using delta-rule-based weight tuning and with Algorithms (a) and (b).

3.1-16: *Passivity properties for an n-layer NN using modified weight tuning.* Show the passivity properties of the input and hidden layers for an n -layer NN using improved weight tuning and with Algorithms (a) and (b).

SECTION 3.3

3.3-1: *One-layer NN.* Consider the system described by

$$x(k+1) = f(x(k), x(k-1)) + g(x(k), x(k-1))u(k)$$

where $f(x(k), x(k-1)) = (x(k)x(k-1)[x(k) + 3.0])/(1 + x^2(k) + x(k-1))$ and $g(x(k), x(k-1)) = (x(k)^2)/(1 + x^2(k) + x(k-1))$. Design a one-layer NN controller with or without learning phase and by using the developed delta-rule-based weight-tuning algorithm and appropriately choosing the adaptation

gains. Repeat the problem by using the modified weight-update weight-tuning methods.

3.3-2: *One-layer NN.* For the system described by

$$x(k+1) = f(x(k), x(k-1)) + (1 + x^2(k))u(k)$$

where $f(x(k), x(k-1)) = (x(k)x(k-1))/(1+x(k))$. Design a one-layer NN controller with or without learning phase and by using the developed delta-rule-based weight-tuning algorithm and appropriately choosing the adaptation gains. Repeat the problem by using the modified weight-update weight-tuning methods.

SECTION 3.4

3.4-1: *Stability and convergence for an n-layer NN.* Assume the hypotheses presented for three-layer NN and use the weight updates presented in (3.283) to (3.286) and extend the stability and boundedness of tracking error and error in weight updates for n -layer NN.

3.4-2: *Stability and convergence for an n-layer NN.* Assume the hypotheses presented for three-layer NN and use the weight updates presented in (3.291) to (3.294) with projection algorithm and show the stability and boundedness of tracking error and error in weight updates for n -layer NN.

3.4-3: *n-layer NN for control.* Perform a MATLAB simulation using an n -NN for the Example 3.4.1. Show the advantage of adding more layers by picking less number of hidden-layer neurons and more than three layers. Use both delta-rule and projection algorithm.

4 Neural Network Control of Uncertain Nonlinear Discrete-Time Systems with Actuator Nonlinearities

Many systems in nature, including biological systems, are dynamical in the sense that they are acted upon by external inputs, have internal memory, and behave in certain ways that can be captured by the notion of the development of activities through time. The name system was formalized in the early 1900s by Whitehead (1953) and von Bertalanffy (1968). A system is viewed as an entity distinct from its environment, whose interactions with the environment can be characterized through input and output signals. An intuitive feel for dynamical systems is provided by Luenberger (1979), which has many excellent examples. The dynamics of a nonlinear system is expressed in state-space form as a nonlinear difference or differential equation (see Equation 2.1). This state equation can describe a variety of dynamical behaviors, including mechanical and electrical systems, earth atmosphere dynamics, planetary orbital dynamics, aircraft dynamics, population growth dynamics, and chaotic behavior.

Industrial systems are generally nonlinear and the dynamics are normally not known beforehand due to the presence of nonlinearities. If the input and output behavior is described by nonlinear difference or differential equations, then the dynamics are considered to have system nonlinearities. If the nonlinearities are known, then they can be cancelled by suitably designing controllers. On the other hand, if the system dynamics are unknown, then controller design is challenging and difficult as presented in this chapter.

4.1 BACKGROUND ON ACTUATOR NONLINEARITIES

Industrial processes such as CNC machines, robots, nano- and micro-manipulation systems, and so on are moved by actuators. An actuator is a device that provides the motive power to the process by mechanically driving it. Actuators are classified as process or control actuators. Joint motors in robotic arms are process actuators whereas actuators used to operate controller components, such as servo valves, are referred to as control actuators. Since processes are modeled as continuous-time systems, most actuators used in control applications are continuous-drive devices. Examples are direct current (DC) motors, induction motors, hydraulic and pneumatic motors, and piston-cylinder drives. There are incremental-drive actuators like stepper motors; these actuators can be treated as digital actuators.

Mechanical parts and elements are unavoidable in all actuator devices. Inaccuracies of mechanical components and the nature of physical laws mean that all these actuator devices are nonlinear. If the input–output relations of the device are nonlinear algebraic equations, this represents a static nonlinearity. On the other hand, if the input–output relations are nonlinear differential or difference equations, it represents a dynamic nonlinearity. Examples of actuator nonlinearities include friction, deadzone, saturation (all static), and backlash and hysteresis (both dynamic).

A general class of industrial processes has the structure of a dynamical system preceded by the nonlinearities of the actuator. Problems in controlling these processes are particularly exacerbated when a high positioning accuracy is required, as in micropositioning or nanopositioning devices. Due to the nonanalytic nature of the actuator nonlinearities and the fact that their exact nonlinear functions are unknown, such processes present a challenge for the control design engineer. Moreover, if the system dynamics are nonlinear and unknown, then designing a controller becomes even more difficult. It is quite common in the real world to observe unknown nonlinear systems with actuator nonlinearities. We refer to these systems as uncertain nonlinear systems with unknown actuator nonlinearities. Next, we will discuss the actuator nonlinearities.

4.1.1 FRICTION

Friction is a natural resistance to relative motion between two contacting bodies and is essential for the operation of common mechanical systems (e.g., wheels, clutches, etc.) But in most industrial processes it also represents a problem, since it is difficult to model or deal with in the controls design. Manufacturers of components for prediction control systems take efforts to minimize friction, and this represents a significant increase in costs. However, notwithstanding efforts at reducing friction, its problems remain and it is necessary to contend

with them in precise control systems. The possible undesirable effects of friction include hangoff and limit cycling. Hangoff prevents the steady-state error from becoming zero with a step command input (this can be interpreted as a DC limit cycle). A limit cycle is the behavior in which the steady-state error oscillates about zero.

Friction is a complicated nonlinear phenomenon in which a force is produced that tends to oppose the motion in a mechanical system. Motion between two contacting bodies causes the dissipation of energy in the system. The physical mechanism for the dissipation depends on the materials of the rubbing surfaces, their finish, the lubrication applied, and other factors, many of which are not yet fully understood. The concern for a controls engineer is not reducing friction but dealing with friction that cannot be reduced. To compensate for friction, it is necessary to understand and have a model of the friction process. Many researchers have studied friction modeling. Extensive work can be found in Armstrong-Helouvry et al. (1994).

4.1.1.1 Static Friction Models

The classic models of frictional force that is proportional to load, opposes the motion, and is independent of contact area was known to Leonardo da Vinci and the model was rediscovered by Coulomb, which is widely used today as the simplest friction model, described by

$$F(v) = a \operatorname{sgn}(v) \quad (4.1)$$

where v is the relative velocity and $F(v)$ is the corresponding force or torque. The parameter a is generally taken as a constant for simplicity. Coulomb friction is shown in Figure 4.1a. A more detailed friction model is shown in Figure 4.1c, which includes viscous friction, a term proportional to the velocity.

Physical experiments have shown that in many cases the force required to initiate relative motion is larger than the force that opposes the motion once it

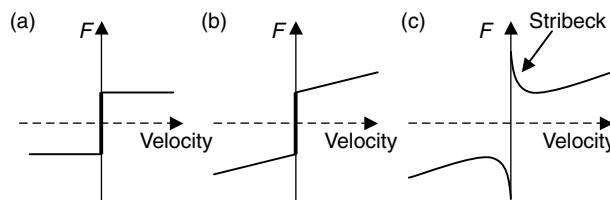


FIGURE 4.1 Friction models. (a) Coulomb friction. (b) Coulomb and viscous friction. (c) Complete friction model.

starts. This effect is known as static friction or stiction. Modeling stiction effects is accomplished by use of a nonlinearity of the form shown in Figure 4.1.

An empirical formula sometimes used for expressing the dependence of the friction force upon velocity is

$$F(v) = (a - b e^{c|v|} + d|v|)\operatorname{sgn}(v) \quad (4.2)$$

in which the parameters a, b, c , and d are chosen to impart the desired shape to the friction function. A complete model for friction suitable for industrial controller design is given in Canudas de Wit et al. (1995) as

$$F(v) = [\alpha_0 + \alpha_1 e^{-\beta_1|v|} + \alpha_2(1 - e^{-\beta_2|v|})]\operatorname{sgn}(v) \quad (4.3)$$

where Coulomb friction is given by α_0 (Nm), static friction is $(\alpha_0 + \alpha_1)$ (Nm), and α_2 (Nm sec/rad) represents the viscous friction model. The effect whereby for small v the frictional force is decreasing with velocity is called negative viscous friction or the Stribeck effect. The Stribeck effect is modeled with an exponential second term in the model (4.3). This friction model captures all the effects shown in Figure 4.1c.

4.1.1.2 Dynamic Friction Models

Though friction is usually modeled as a static discontinuous map between velocity and friction torque, which depends on the velocity's sign, there are several interesting properties observed in systems with friction that cannot be explained only by static models. This is basically due to the fact that the friction does not have an instantaneous response to a change in velocity (i.e., it has internal dynamics). Examples of these dynamic properties are:

- Stick-slip motion, which consists of limit cycle oscillation at low velocities, caused by the fact that friction is larger at rest than during motion.
- Presliding displacement, which means that friction behaves like a spring when the applied force is less than the static friction break-away force.
- Frictional lag, which means that there is a hysteresis in the relationship between friction and velocity.

All these static and dynamic characteristics of friction were captured by the dynamical model proposed in Canudas de Wit et al. (1995). This model is referred to as Lugre (the Lund–Grenoble) model. The Lugre model in

continuous-time is given by

$$\begin{aligned}\frac{dz}{dt} &= \dot{q} - \frac{\sigma_0}{g(\dot{q})} z |\dot{q}| \\ g(\dot{q}) &= \alpha_0 + \alpha_1 e^{-(\dot{q}/v_0)^2} \\ F &= \sigma_0 z + \sigma_1 \frac{dz}{dt} + \alpha_2 \dot{q}\end{aligned}\quad (4.4)$$

where \dot{q} is the angular velocity and F is the frictional force. The first equation represents the dynamics of the friction internal state z , which describes the average relative deflection of the contact surfaces during the stiction phases. This state is not measurable. The function $g(\dot{q})$ describes the steady-state part of the model or constant velocity motions: v_0 is the Stribeck velocity, $(\alpha_0 + \alpha_1)$ is the static friction, and α_0 is the Coulomb friction. Thus the complete friction model is characterized by four static parameters, $\alpha_0, \alpha_1, \alpha_2$, and v_0 and two dynamic parameters, σ_0 and σ_1 . The parameter σ_0 can be understood as a stiffness coefficient of the microscopic deformations of z during the presliding displacement, while σ_1 is a damping coefficient associated with dz/dt . The overall friction model is highly nonlinear and it requires an advanced compensator. Therefore, a neural network (NN) controller to compensate the friction during object grasping using the above model is given in Jagannathan and Galan (2004). For friction model in discrete-time, one has to convert the Lugre model from continuous-time into discrete-time and subsequently a suitable controller has to be designed to compensate the friction in discrete-time.

4.1.2 DEADZONE

Deadzone (Tao and Kokotovic 1996) is a static nonlinearity that describes the insensitivity of the system to small signals. Although there are some open-loop applications where the deadzone characteristics are highly desirable, in most closed-loop applications, deadzone has undesirable effects on the feedback loop dynamics and control system performance. The signal is considered lost if it falls within the deadband and causes limit cycles, tracking errors, and so forth.

Deadzone has a static input–output relationship shown in Figure 4.2. A mathematical model is given by

$$\tau(k) = D(u(k)) = \begin{cases} m_-(u(k) + d(k)_-), & u(k) < -d(k)_- \\ 0, & -d(k)_- \leq u(k) < d_+(k) \\ m_+(u(k) - d_+(k)), & u(k) \geq d_+(k) \end{cases} \quad (4.5)$$

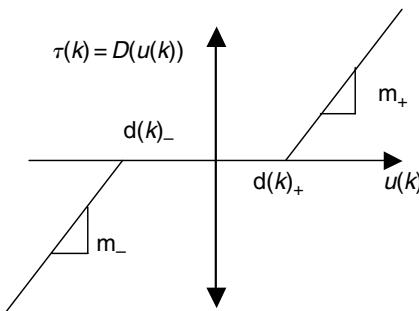


FIGURE 4.2 Deadzone nonlinearity.

One can see that there is no output as long as the input signal is in the deadband, defined by $-d(k)_- < u(k) < d(k)_+$. When the signal falls into this deadband, the output signal is zero, and one loses information about the input signal. Once the output appears, the slope between input and output stays constant. Note that (4.5) represents a nonsymmetric deadzone model since the slope on the left and right sides of the deadzone are not the same.

Deadzone characteristics (4.5) can be parameterized by the four constants $d(k)_-, m_-, d(k)_+,$ and m_+ . In practical motion control systems, these parameters are unknown and compensation of such nonlinearities is difficult. Deadzones usually appear at the input of the actuator systems, as in the case of a DC motor, but there are also deadzones at the output, where the nonlinearities appear at the output of the system.

A deadzone is usually caused by friction, which can vary with temperature and wear. Also, these nonlinearities may appear in mass produced components, such as valves and gears of a hydraulic or pneumatic system, which can vary from one component to another.

An example of deadzone caused by friction given in Tao and Kokotovic (1996) is shown in Figure 4.3. The input to the motor is motor torque T_m ; the transfer function in the forward path is a first-order system with time constant τ . There is a Coulomb friction in the feedback path. If the time constant τ is negligible, the low-frequency approximation of the feedback loop is given by the deadzone characteristic shown in Figure 4.3. Note that the friction torque characteristic is responsible for the break points d_+ and d_- , while the feedforward gain m determines the slope of the deadzone function.

The deadzone may be written as

$$\tau(k) = D(u(k)) = u(k) - \text{sat}_d(u(k)) \quad (4.6)$$

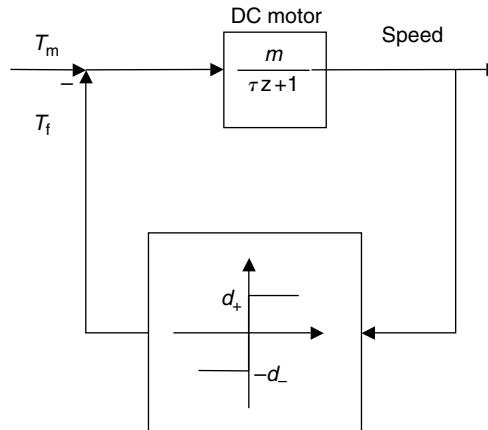


FIGURE 4.3 Deadzone caused by friction in a DC motor.

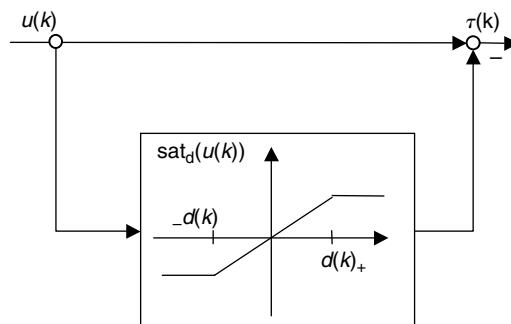


FIGURE 4.4 Decomposition of deadzone into feedforward plus unknown path.

where the nonsymmetric saturation function is defined as

$$\text{sat}_d(u(k)) = \begin{cases} -d(k)_-, & u(k) < -d(k)_- \\ u(k), & -d(k)_- \leq u(k) < d(k)_+ \\ d(k)_+, & d(k)_+ \leq u(k) \end{cases} \quad (4.7)$$

This decomposition, shown in Figure 4.4, represents a feedforward path plus an unknown parallel path and is extremely useful in the controls design.

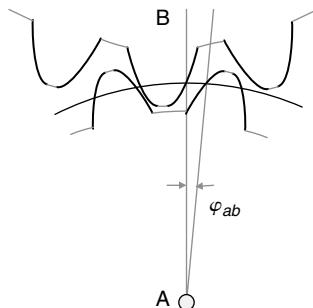


FIGURE 4.5 Backlash in a gear system.

4.1.3 BACKLASH

The space between the teeth on a mechanical gearing system must be made larger than the gear teeth width as measured on the pitch circle. If this were not the case, the gears will not mesh without jamming. The difference between tooth space and tooth width is known as backlash. Figure 4.5 shows the backlash present between two meshing spur gears. Any amount of backlash greater than the minimum amount necessary to ensure satisfactory meshing of gears can result in instability in dynamic situations as well as in position errors in gear trains. In fact, there are many applications such as instrument differential gear trains and servo mechanisms that require the complete elimination of backlash in order to function properly.

Backlash results in a delay in the system motion. One can see that when the driving gear changes its direction, the driven gear follows only after some delay. A model of the backlash in mechanical systems is shown in Figure 4.5.

A standard mathematical model is given by

$$\begin{aligned} \tau(k+1) &= B(\tau(k), u(k), u(k+1)) \\ &= \begin{cases} mu(k) - md_+, & \text{if } u(k+1) - u(k) > 0 \text{ and } \tau(k) \leq mu(k) - md_+ \\ mu(k) - md_-, & \text{if } u(k+1) - u(k) < 0 \text{ and } \tau(k) \geq mu(k) - md_- \\ \tau(k), & \text{otherwise} \end{cases} \quad (4.8) \end{aligned}$$

One can see that backlash is a first-order velocity-driven dynamical system with inputs $u(k)$ and $u(k+1)$ and state $\tau(k)$. It contains its own dynamics; therefore, its compensation requires the use of a dynamic compensator. Whenever, the driving motion $u(k)$ changes its direction, the resultant motion $\tau(k)$ is delayed from the motion of $u(k)$. The objective of a backlash compensator is to make this delay as small as possible (i.e., to make the throughput from $u(k)$ to $\tau(k)$ to

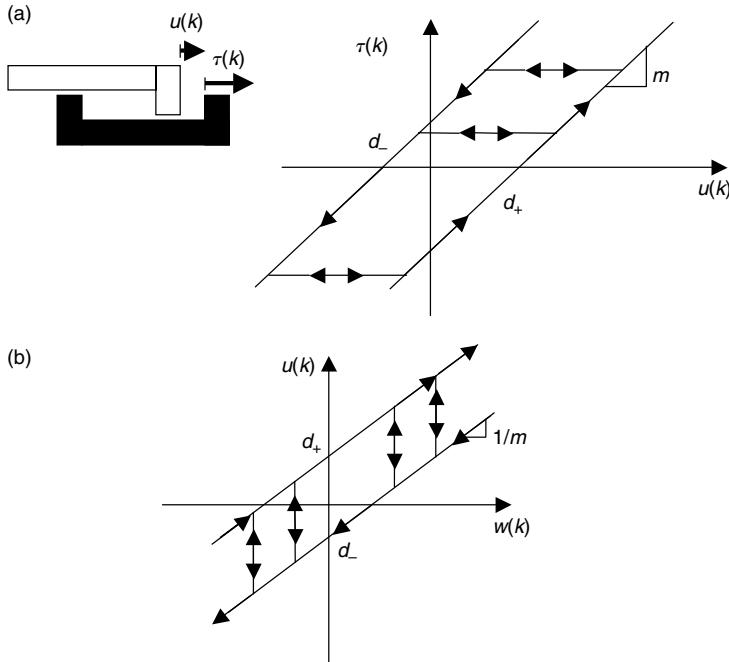


FIGURE 4.6 (a) Backlash nonlinearity and (b) its inverse.

be unity). The backlash precompensator must be a dynamic compensator which needs to generate the inverse of the backlash nonlinearity. The backlash inverse function is shown in Figure 4.6b. The dynamics of the backlash compensator may be written in the form

$$u(k+1) = B_{\text{inv}}(u(k), w(k), w(k+1)) \quad (4.9)$$

The backlash inverse characteristic shown in Figure 4.6b can be decomposed into two parts: a known direct feedforward term plus an additional parallel path containing a modified backlash inverse term shown in Figure 4.7. This decomposition allows the design of a compensator that has a better structure (Lewis et al. 2002) than when a backlash compensator is used directly in the feedforward path.

4.1.4 SATURATION

One of the major problems that arise while controlling dynamic systems is the magnitude of the control input. Physical limitations dictate that hard limits be

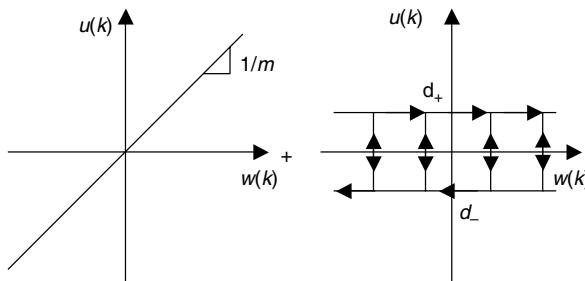


FIGURE 4.7 Backlash inverse decomposition with unity feedforward path.

imposed on the magnitude to avoid damage to or deterioration of the process. Hence, input that is determined online should meet the desired control objectives while remaining within certain limits. Hence, ensuring that the control input does not exceed certain limits while simultaneously realizing the performance objectives is a very important issue. There may be other instances where input saturation may even be desired from an optimality point of view.

The magnitude constraint on the control input is typically modeled as a saturation nonlinearity. Then one has to consider this nonlinearity in the controller design. This makes the closed-loop analysis difficult when the system is uncertain and nonlinear. The adaptive controller design for a linear time-invariant plant in continuous-time with input constraints was discussed in Karason and Annaswamy (1994). There is no work done in the area of discrete-time control when actuator limits are applied. In this chapter, the adaptive neural network (NN) controller design with saturation nonlinearity is given.

4.2 REINFORCEMENT NN LEARNING CONTROL WITH SATURATION

Nonlinear systems, for instance robot manipulators and high-power machinery, often have actuator nonlinearities such as deadzones and magnitude constraints typically modeled using the saturation function. Compensation of actuator nonlinearities in continuous-time using adaptive control techniques is discussed in Tao and Kokotovic (1995). As mentioned earlier, the adaptive control schemes require that the nonlinear systems under consideration satisfy the linear in the unknown parameters assumption.

On the other hand, learning-based control methodology using NN is an alternative to adaptive control since these NN can be considered as general tools for modeling nonlinear systems. Work on adaptive NN control using the universal NN approximation property is now being pursued by several groups of researchers (Calise 1996; Lewis et al. 1999, 2002; Jagannathan 2001). However, as shown in Chapter 3, significant work on NN control is accomplished

either using supervised NN training (Narendra and Parthasarathy 1990) where the user specifies a desired output for the mapping, or by using classical adaptive control-based methods (Calise 1996; Lewis et al. 1999) where a short-term system performance measure is normally defined by using the tracking error.

In recent years, adaptive critic NN approach using reinforcement learning has emerged as a promising approach to optimal control by using NN due to its potential to find approximate solutions to dynamic programming, where a long-term system performance measure can be optimized in contrast to the short-term performance measure used in the classical adaptive control. There are many variants of adaptive critic NN architecture (Werbos 1991, 1992; Barto 1992; Prokhorov and Wunsch 1997; Murray et al. 2002; Shervais et al. 2003) and some are (1) heuristic dynamic programming (HDP), (2) dual heuristic dynamic programming (DHP), and (3) globalized dual heuristic dynamic programming (GDHP). However, until now, there have been few papers in the literature (Bertsekas and Tsitsiklis 1996; Lin and Balakrishnan 2000; Murray et al. 2002) that present the convergence of the adaptive critic designs and the stability of the overall system. Moreover, an off-line training scheme is usually employed (Prokhorov and Wunsch 1997). Both the techniques in Murray et al. (2002) and Si and Wang (2001) study the convergence issue based on the recursive stochastic algorithms, where the convergence with probability one is achieved. In Lin and Balakrishnan (2000), the critic is used to approximate the Hamilton–Jacobi–Bellman (HJB) equation and error convergence for a linear time-invariant discrete system is addressed. In Shervais et al. (2003), hard computing techniques were utilized to verify the stability for nonlinear systems in continuous-time. In Prokhorov and Feldkamp (1998), an algorithm is presented to use the adaptive critic NN to approximate the Lyapunov function.

By contrast, in this chapter (He and Jagannathan 2003), the stability of the adaptive critic NN design for a class of discrete-time nonlinear systems is assured by demonstrating the existence of a Lyapunov function for the overall system. The selection of such a function is the critical part of the stability proof, and it is rarely straightforward. Moreover, the Lyapunov stability of the closed-loop system in the presence of NN approximation errors and unknown but bounded disturbances is presented unlike in the above existing works where the convergence is given in ideal circumstances. Moreover, the actuator constraints are not asserted in both tracking error and adaptive critic NN control techniques.

First, a conventional adaptive tracking error-based NN controller with input constraints is designed using Lyapunov stability analysis to control an uncertain nonlinear discrete-time system. Subsequently, a novel adaptive critic NN-based controller, which includes an action plus a critic NN, is introduced in this chapter (He and Jagannathan 2003) to control a class of nonlinear discrete-time systems. The critic NN approximates a certain strategic utility function, which is taken as the long-term performance measure of the system. The action NN weights

are tuned by both the critic NN signal and the filtered tracking error to minimize the strategic utility function and uncertain system dynamic estimation error so that the action NN can generate an optimal control signal. This optimal action NN control signal combined with an additional outer-loop conventional control signal is applied as the overall control input to the nonlinear discrete-time system. The outer-loop conventional signal allows the action and critic NN learn online while making the system stable. By selecting the appropriate objective functions for both critic and action NNs, the closed-loop stability is inferred. The proposed critic NN architecture overcomes the limitation of using the tracking error at one step ahead by minimizing a certain long-term performance measure. In fact, the proposed architecture can be viewed as the supervised actor-critic reinforcement learning architecture (Rosenstein and Barto 2004). Here the outer loop conventional signal can be treated as the additional feedback signal from the supervisor.

The available adaptive critic NN controllers (Si 2002) employ the gradient-descent backpropagation NN learning scheme to tune the weights of the action-generating and critic NNs so that an explicit off-line training phase is needed whereas with the proposed scheme, the initial weights are selected at zero or random and they are tuned online. Moreover, the actuator constraints are considered in the adaptive critic design in contrast with the previous research works, where no explicit magnitude constraint is treated. In fact, the proposed adaptive critic NN control design (He and Jagannathan 2003) addresses the input saturation for unknown nonlinear systems by introducing an auxiliary linear system similar to the one derived in Karason and Annaswamy (1994) where magnitude constraints are considered for a linear system. The online tuning, Lyapunov stability in the presence of NN approximation errors and bounded disturbances, and the consideration of the actuator constraints pave the way for practical use of the adaptive critic design. Moreover, by appropriately selecting the NN weight updates based on a quadratic performance index, an optimal/suboptimal control sequence can be generated in contrast with standard NN works.

4.2.1 NONLINEAR SYSTEM DESCRIPTION

Consider the following nonlinear system, to be controlled, given in the following form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \tag{4.10}$$

where $x(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T \in \Re^{nm}$ with each $x_i(k) \in \Re^m$; $i = 1, \dots, n$ is the state at time instant k , $f(x(k)) \in \Re^m$ is the unknown nonlinear dynamics of the system, $u(k) \in \Re^m$ is the input, and $d(k) \in \Re^m$ is the unknown but bounded disturbance, whose bound is assumed to be a known constant, $\|d(k)\| \leq d_m$. Several NN learning schemes have been proposed recently in the literature to control the class of nonlinear systems described in Chapter 3, but the main contribution of the section is the adaptive critic NN-based controller in the presence of magnitude constraints and the associated stability analysis. This section describes the results from the paper by He and Jagannathan (2003).

Given a trajectory, $x_{nd}(k) \in \Re^m$, and its past values, define the tracking error

$$e_i(k) = x_i(k) - x_{nd}(k + i - n) \quad (4.11)$$

and the filtered tracking error, $r(k) \in \Re^m$, as

$$r(k) = [\lambda \ I]e(k) \quad (4.12)$$

with $e(k) = [e_1(k), e_2(k), \dots, e_n(k)]^T$, $e_1(k+1) = e_2(k)$, where $e_1(k+1)$ is the next future value for the error $e_1(k)$, $e_{n-1}(k), \dots, e_1(k)$ are past values of the error $e_n(k)$, $I \in \Re^{m \times m}$ is an identity matrix, and $\lambda = [\lambda_{n-1}, \lambda_{n-2}, \dots, \lambda_1] \in \Re^{m \times (n-1)m}$ is a constant diagonal positive definite matrix selected such that the eigenvalues are within a unit disc. Consequently, if the filtered tracking error $r(k)$ tends to zero, then all the tracking errors go to zero. Equation 4.12 can be expressed as

$$r(k+1) = f(x(k)) - x_{nd}(k+1) + \lambda_1 e_n(k) + \dots + \lambda_{n-1} e_2(k) + u(k) + d(k) \quad (4.13)$$

The control objective is to make all the tracking errors bounded close to zero while ensuring that all the internal signals are uniformly ultimately bounded (UUB).

4.2.2 CONTROLLER DESIGN BASED ON THE FILTERED TRACKING ERROR

Define the control input $u(k) \in \Re^m$ as

$$u(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) \quad (4.14)$$

where $\hat{f}(x(k)) \in R^m$ is an estimate of the unknown function $f(x(k))$, and $l_v \in R^{m \times m}$ is a diagonal gain matrix. Then, the closed-loop system becomes

$$r(k+1) = l_v r(k) - \tilde{f}(x(k)) + d(k) \quad (4.15)$$

where the functional estimation error is given by $\tilde{f}(x(k)) = \hat{f}(x(k)) - f(x(k))$.

Equation 4.15 relates the filtered tracking error with the functional estimation error and the filtered tracking error system (4.15) can also be expressed as

$$r(k+1) = l_v r(k) + \delta_0(k) \quad (4.16)$$

where $\delta_0(k) = -\tilde{f}(x(k)) + d(k)$. If the functional estimation error $\tilde{f}(x(k))$ is bounded such that $\|\tilde{f}(x(k))\| \leq f_M$, for some known bounding function $f_M \in \Re$ then the next stability results hold.

Theorem 4.2.1: Consider the system given by (4.10). Let the control action be provided by (4.14). Assume the functional estimation error and the unknown disturbance to be bounded. The filtered tracking error system (4.16) is stable, provided

$$l_{v \max} < 1 \quad (4.17)$$

where $l_{v \max} \in \Re$ is the maximum singular value of the matrix l_v .

Proof: Let us consider the following Lyapunov function candidate

$$J(k) = r^T(k)r(k) \quad (4.18)$$

The first difference is

$$\Delta J(k) = r^T(k+1)r(k+1) - r^T(k)r(k) \quad (4.19)$$

Substituting the filtered tracking error dynamics (4.16) in (4.19) results in

$$\Delta J(k) = (l_v r(k) - \tilde{f}(x(k)) + d(k))^T (l_v r(k) - \tilde{f}(x(k)) + d(k)) - r^T(k)r(k) \quad (4.20)$$

This implies that $\Delta J(k) \leq 0$ provided $\|(l_v r(k) - \tilde{f}(x(k)) + d(k))\| \leq l_{v \max} \|r(k)\| + f_M + d_M < \|r(k)\|\). This further implies that$

$$\|r(k)\| < \frac{f_M + d_M}{(1 - l_{v \max})} \quad (4.21)$$

The closed-loop system is UUB.

4.2.3 ONE-LAYER NN CONTROLLER DESIGN

In the previous chapter, adaptive NN controller was designed using the tracking error methodology. Tracking error-based NN controllers use a short-term performance index. In this section, reinforcement learning-based adaptive NN controller is discussed using a long-term performance measure. This approach is analogous to other reinforcement learning-based methods but it is not the same. The primary motive is to use some sort of reinforcement learning in the controller as well as to show the closed-loop stability analysis unlike in the prior work where either stability analysis is ignored (Prokhorov and Wunsch 1997) or presented in an average sense (Si and Wang 2001). This approach can be viewed as the supervised actor-critic reinforcement learning scheme (Rosenstein and Barto 2004).

To minimize the computational overhead, a single-layer NN is considered first for both critic and action NNs and the NN controller is designed without taking into account the input (or actuator) magnitude constraints. The strategic utility function is defined first with the critic signal serving as an approximator for this strategic utility function and the action NN is constructed to minimize this strategic utility function and the unknown function estimation errors. Stability analysis using a Lyapunov direct method is carried out for the closed-loop system (4.15) using novel weight-tuning updates. In the next section, the controller design with input constraints is discussed.

4.2.3.1 The Strategic Utility Function

The utility function $p(k) = [p_i(k)]_{i=1}^m \in \Re^m$ is defined based on the current filtered tracking error $r(k)$ and it is given by

$$p_i(k) = \begin{cases} 0, & \text{if } r_i^2(k) \leq c \\ 1, & \text{if } r_i^2(k) > c \end{cases} \quad i = 1, 2, \dots, m \quad (4.22)$$

where $c \in \Re$ is a predefined threshold. The utility function $p(k)$ is viewed as the current system performance index: $p_i(k) = 0$ stands for the good tracking performance and $p_i(k) = 1$ stands for the bad tracking performance. The long-term

system performance measure given in terms of the strategic utility function $Q^T(k) \in \Re^m$, is defined as

$$Q^T(k) = \alpha^N p(k+1) + \alpha^{N-1} p(k+2) + \cdots + \alpha^{k+1} p(N) + \cdots \quad (4.23)$$

where $\alpha \in \Re$ and $0 < \alpha < 1$, and N denotes the stage number, when the number of stages N is large or infinite, this problem may be defined over a rolling horizon with a fixed number of stages. Equation 4.23 can also be expressed as $Q(k) = \min_{u(k)} \{\alpha Q(k-1) - \alpha^{N+1} p(k)\}$. This utility function is quite similar to the Bellman equation.

4.2.3.2 Critic NN

The critic NN is used to approximate the *strategic* utility function $Q^T(k)$. We define the prediction error (Si and Wang 2001) as

$$e_c(k) = \hat{Q}(k) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)), \quad (4.24)$$

where

$$\hat{Q}(k) = \hat{w}_1^T(k) \phi_1(v_1^T x(k)) = \hat{w}_1^T(k) \phi_1(k) \quad (4.25)$$

and $\hat{Q}(k) \in \Re^m$ is the critic signal, $\hat{w}_1(k) \in \Re^{n_1 \times m}$ and $v_1 \in \Re^{nm \times n_1}$ represent the matrix of weight estimates, $\phi_1(k) \in \Re^{n_1}$ is the activation function vector in the hidden layer, n_1 is the number of the nodes in the hidden layer, and the critic NN input is given by $x(k) \in \Re^{nm}$. The objective function to be minimized by the critic NN is defined as

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k) \quad (4.26)$$

The weight-update rule for the critic NN is a gradient-based adaptation, which is given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) + \Delta \hat{w}_1(k) \quad (4.27)$$

where

$$\Delta \hat{w}_1(k) = \alpha_1 \left[-\frac{\partial E_c(k)}{\partial \hat{w}_1(k)} \right] \quad (4.28)$$

or

$$\begin{aligned}\hat{w}_1(k+1) = & \hat{w}_1(k) - \alpha_1 \phi_1(k)(\hat{w}_1^T(k)\phi_1(k) \\ & + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1)\phi_1(k-1))^T\end{aligned}\quad (4.29)$$

where $\alpha_1 \in \Re$ is the NN adaptation gain.

4.2.3.3 Action NN

The output of the action NN signal will be used to approximate the unknown nonlinear function $f(x(k))$ and to provide an optimal control signal to be a part of overall input $u(k)$ as

$$\hat{f}(k) = \hat{w}_2^T(k)\phi_2(v_2^T x(k)) = \hat{w}_2^T(k)\phi_2(k) \quad (4.30)$$

where $\hat{w}_2(k) \in \Re^{n_2 \times m}$ and $v_2 \in \Re^{nm \times n_2}$ represent the matrix of weight estimates, $\phi_2(k) \in \Re^{n_2}$ is the activation function vector in the hidden layer, n_2 is the number of the nodes in the hidden layer, and the input to the critic NN is $x(k) \in \Re^{nm}$.

Suppose the unknown target output layer weights for the action NN is w_2 , then we have

$$f(k) = w_2^T \phi_2(v_2^T x(k)) + \varepsilon_2(x(k)) = w_2^T(k) \phi_2(k) + \varepsilon_2(x(k)) \quad (4.31)$$

where the $\varepsilon_2(x(k)) \in \Re^m$ is the NN reconstruction error. Combining (3.125) and (3.126) to get

$$\tilde{f}(k) = \hat{f}(k) - f(k) = (\hat{w}_2(k) - w_2)^T \phi_2(k) - \varepsilon_2(x(k)) \quad (4.32)$$

where $\tilde{f}(k) \in \Re^m$ is the functional estimation error. The action NN weights are tuned by using the functional estimation error, $\tilde{f}(k)$ and the error between the desired *strategic* utility function $Q_d(k) \in R^m$ and the critic signal $\hat{Q}(k)$. Define

$$e_a(k) = \tilde{f}(k) + (\hat{Q}(k) - Q_d(k)), \quad (4.33)$$

where $e_a(k) \in \Re^m$. The additional functional estimation error signal can be viewed as the supervisor's additional evaluation signal to the actor. As the actor gains proficiency, this signal will become close to zero which can be viewed as gradual withdrawal of the additional feedback to shape the learned policy toward optimality.

Our desired value for the utility function $Q_d(k)$ is “0” (Si and Wang 2001) at every step, then the nonlinear system can track the reference signal well.

Thus, (4.31) becomes

$$e_a(k) = \tilde{f}(k) + \hat{Q}(k) \quad (4.34)$$

The objective function to be minimized by the action NN is given by

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \quad (4.35)$$

The weight-update rule for the action NN is also a gradient-based adaptation, which is defined as

$$\hat{w}_2(k+1) = \hat{w}_2(k) + \Delta \hat{w}_2(k) \quad (4.36)$$

where

$$\Delta \hat{w}_2(k) = \alpha_2 \left[-\frac{\partial E_a(k)}{\partial \hat{w}_a(k)} \right] \quad (4.37)$$

or

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(k) (\hat{w}_2^T(k) \phi_2(k) + \tilde{f}(k))^T \quad (4.38)$$

where $\alpha_2 \in \Re$ is the NN adaptation gain.

The NN weight-update rule in (4.38) cannot be implemented in practice since the nonlinear function $f(x(k))$ is unknown. However, using (4.15), the functional estimation error is given by

$$\tilde{f}(x(k)) = l_v r(k) - r(k+1) + d(k) \quad (4.39)$$

Substituting (4.39) into (4.38) to get

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(k) (\hat{w}_2^T(k) \phi_2(k) + l_v r(k) - r(k+1) + d(k))^T \quad (4.40)$$

To implement the weight-update rule, the unknown but bounded disturbance $d(k)$ is taken to be zero. Then (4.40) is rewritten as

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(k) (\hat{w}_2^T(k) \phi_2(k) + l_v r(k) - r(k+1))^T \quad (4.41)$$

First the design of the NN controller without saturation nonlinearity is given and then the saturation nonlinearity is introduced for accommodating the actuator constraints.

4.2.4 NN CONTROLLER WITHOUT SATURATION NONLINEARITY

Assumption 4.2.1 (Bounded Ideal Weights): Let w_1 and w_2 be the unknown output-layer target NN weights for the critic and action-generating NNs and assume that they are bounded above so that

$$\|w_1\| \leq w_{1m} \quad \|w_2\| \leq w_{2m} \quad (4.42)$$

Here $w_{1m} \in \Re$ and $w_{2m} \in \Re$ represent the bounds on the unknown weights where the Frobenius norm (Lewis et al. 1999) is used throughout the discussion. The error in weights during estimation is given by

$$\tilde{w}_i(k) = \hat{w}_i(k) - w_i \quad i = 1, 2 \quad (4.43)$$

Fact 4.2.1: The activation functions are bounded by known positive values so that

$$\|\phi_i(k)\| \leq \phi_{im} \quad i = 1, 2 \quad (4.44)$$

where $\phi_{im} \in \Re$, $i = 1, 2$ is the upper bound for $\phi_i(k)$, $i = 1, 2$.

Let the control input, $u(k)$, be selected by (4.14) along with the unknown function estimation (4.12), then the filtered tracking error dynamics (4.15) become

$$r(k+1) = l_v r(k) - \zeta_2(k) + \varepsilon_2(x(k)) + d(k) \quad (4.45)$$

where $\zeta_2(k) = \tilde{w}_2^T(k)\phi_2(k)$ and $\varepsilon_2(x(k)) \in \Re^m$ is the NN reconstruction error vector.

Assumption 4.2.2 (Bounded NN Reconstruction Error): The NN reconstruction error $\varepsilon_2(x(k))$ is bounded over the compact set S by ε_{2m} .

Remark: It is shown in Igelnik and Pao (1995) that, if the number of hidden-layer nodes is sufficiently large, the reconstruction error can be made arbitrarily small on the compact set. Moreover, the Assumptions 4.2.1 and 4.2.2 do not guarantee that the functional estimation error $f(x(k))$ is bounded unless the weight estimation $\hat{w}_2(k)$ is bounded. Boundedness of the weight estimation error is demonstrated using the Lyapunov analysis.

The structure of the proposed adaptive critic NN controller is depicted in Figure 4.8 and the details are given in Table 4.1. In the NN controller structure, an inner action-generating NN loop compensates the nonlinear dynamics of the

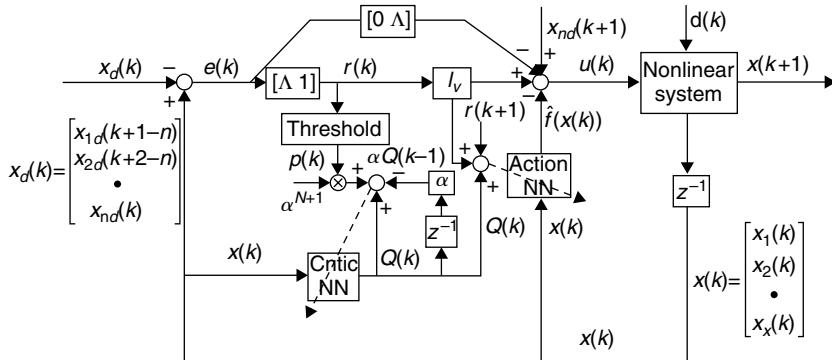


FIGURE 4.8 Adaptive critic NN-based controller structure. Solid lines denote signal flow, while the dashed lines represent weights tuning.

system. The outer loop designed via Lyapunov analysis guarantees the stability and accuracy in following the desired trajectory.

It is required to demonstrate that the filtered tracking error, $r(k)$ is suitably small and that the NN weights, $\hat{w}_1(k), \hat{w}_2(k)$ remain bounded. This can be achieved by suitably choosing the control parameter and the adaptation parameters. The selection of them is given by the direct Lyapunov method.

Theorem 4.2.2 (NN Controller without Saturation Nonlinearity): Let the desired trajectory, $x_{nd}(k)$ and its past values be bounded. Also, let the Assumptions 4.2.1 and 4.2.2 hold and the disturbance bound d_m be a known constant. Let the critic NN weight tuning be given by (4.29) and the action NN weight tuning provided by (4.32). Then the filtered tracking error, $r(k)$ and the NN weight estimates, $\hat{w}_1(k), \hat{w}_2(k)$ are UUB, with the bounds specifically given by (4.A.15) through (4.A.17), provided the controller design parameters are selected as:

$$\text{a. } \alpha_1 \|\phi_1(k)\|^2 < 1 \quad (4.46)$$

$$\text{b. } \alpha_2 \|\phi_2(k)\|^2 < 1 \quad (4.47)$$

$$\text{c. } 0 < \alpha < \frac{\sqrt{2}}{2} \quad (4.48)$$

$$\text{d. } l_v \max < \frac{\sqrt{3}}{3} \quad (4.49)$$

Proof: See Appendix 4.A.

TABLE 4.1**Reinforcement Learning NN Control without Magnitude Constraints**

Define the control input $u(k) \in \mathfrak{N}^m$ as

$$u(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \cdots - \lambda_{n-1} e_2(k)$$

The utility function $p(k) = [p_i(k)]_{i=1}^m \in \mathfrak{N}^m$ is defined as

$$p_i(k) = \begin{cases} 0, & \text{if } r_i^2(k) \leq c \\ 1, & \text{if } r_i^2(k) > c \end{cases} \quad i = 1, 2, \dots, m$$

where $c \in \mathfrak{N}$ is a predefined threshold. The long-term system performance measure given in terms of the strategic utility function $Q^T(k) \in \mathfrak{N}^m$, is given by

$$Q^T(k) = \alpha^N p(k+1) + \alpha^{N-1} p(k+2) + \cdots + \alpha^{k+1} p(N)$$

where $\alpha \in \mathfrak{N}$ and $0 < \alpha < 1$, and N is the final time instant.

Critic NN output and weight tuning:

$$\hat{Q}(k) = \hat{w}_1^T(k)\phi_1(v_1^T x(k)) = \hat{w}_1^T(k)\phi_1(k)$$

where $\hat{Q}(k) \in \mathfrak{N}^m$ is the critic NN signal and the critic NN weights are tuned by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(k)(\hat{w}_1^T(k)\phi_1(k) + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1)\phi_1(k-1))^T$$

Action NN output and weight tuning:

The action NN output which is part of overall input $u(k)$ is given by

$$\hat{f}(k) = \hat{w}_2^T(k)\phi_2(v_2^T x(k)) = \hat{w}_2^T(k)\phi_2(k)$$

and the action NN weight tuning is provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(k)(\hat{w}_2^T(k)\phi_2(k) + l_v r(k) - r(k+1))^T$$

where $\alpha_1, \alpha_2 \in \mathfrak{N}$ represent learning rate parameters. Here $\alpha \in \mathfrak{N}$ is a design parameter, and $l_{v \max} \in \mathfrak{N}$ the maximum singular value of the gain matrix l_v .

Remarks:

1. It is important to note that in this theorem there is no certainty equivalence (CE) and linear in the unknown parameter (LIP) assumptions for the NN controller, in contrast to standard work in discrete-time adaptive control (Åström and Wittenmark 1989; Kanellakopolous 1994). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption and another Lyapunov function is selected that weighs

only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in the appendix is of the form (4.A.1), which weights the filtered tracking errors, the NN estimation errors for the controller, $\tilde{w}_1(k)$ and $\tilde{w}_2(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof and not have to be selected a priori in an ad hoc manner. Here the weight-tuning schemes derived from minimizing certain quadratic objective functions are used in the Lyapunov proof.

2. The NN weight-updating rules (4.29) and (4.41) are much simpler than in Jagannathan and Lewis (1996) since they do not include an extra term, referred to as discrete-time ε -modification (Jagannathan and Lewis 1996; Jagannathan 2001), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required. As a result, the complexity of the proof as well as the computational overhead is reduced significantly without the persistence of excitation (PE) condition.
3. Both NN weight-tuning rules (4.29) and (4.41) are updated online in contrast to the off-line training in the previous works.
4. Condition (4.46) can be verified easily. For instance, the hidden layer of the critic NN consists of n_1 nodes with the hyperbolic tangent sigmoid function as its activation function, then $\|\phi_1(\cdot)\|^2 \leq n_1$. The NN learning rate α_1 can be selected as $0 < \alpha_1 < 1/n_1$ to satisfy (4.46). Similar analysis can be performed to obtain the NN learning rate α_2 .
5. Controller parameter $l_{v\max}$ and parameter α have to be selected using (4.49) and (4.48) in order for the closed-loop system to be stable.
6. The weights of the action-generating and critic NNs can be initialized at zero and stability will be maintained by the outer-loop conventional controller until the NNs learn. This means that there is no explicit off-line learning phase needed.
7. There is no information available yet to decide the number of hidden-layer neurons for a multilayer NN. However, the number of hidden-layer neurons required for suitable approximation can be addressed by using the stability of the closed-loop system and the error bounds of the NNs. From (4.46), (4.47) and Remark 4, to make the closed-loop system stable, the numbers of the hidden-layer nodes have to satisfy $n_1 > (1/\alpha_1)$ and $n_2 > (1/\alpha_2)$ once the NN learning

rate parameters α_1 and α_2 are selected. With regards to the error bounds of NN, in our paper, a single-layer NN is used to approximate the continuous functions on a compact set, S . According to Igelnik and Pao (1995), if the hidden layer nodes are large enough, the reconstruction error, $\varepsilon(k)$, approaches zero. If the continuous function is restricted to satisfy the Lipschitz condition, then the reconstruction error of order $O(C/\sqrt{n})$ is achieved, where n is the number of hidden layer nodes, and C is independent of n .

The adaptive critic NN controller presented above does not include the magnitude constraints on the control input. To embed the input constraints as a saturation nonlinearity in the controller structure, an auxiliary system (Karason and Annaswamy 1994) is introduced and the stability of the closed system is demonstrated as given next.

4.2.5 ADAPTIVE NN CONTROLLER DESIGN WITH SATURATION NONLINEARITY

In this section, we will apply the magnitude constraints of the actuator and evaluate the performance of the controller. The stability analysis carried out in the previous section has to be redone to accommodate the magnitude constraints. In order to accommodate the actuator magnitude constraints, we introduce an auxiliary control input $v(k)$ as presented next.

4.2.5.1 Auxiliary System Design

Define the auxiliary control input $v(k)$ as

$$v(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) \quad (4.50)$$

where $\hat{f}(x(k))$ is an estimate of the unknown function $f(x(k))$, and $l_v \in \Re^{m \times m}$ is a diagonal gain matrix. The actual control input after the incorporation of saturation constraints is selected as

$$u(k) = \begin{cases} v(k), & \text{if } \|v(k)\| \leq u_{\max} \\ u_{\max} \operatorname{sgn}(v(k)), & \text{if } \|v(k)\| > u_{\max} \end{cases} \quad (4.51)$$

where $u_{\max} \in \Re$ is the upper bound for the control input $u(k)$. Then, the closed-loop system becomes

$$r(k+1) = l_v r(k) - \tilde{f}(x(k)) + d(k) + \Delta u(k) \quad (4.52)$$

where the functional estimation error is given by $\tilde{f}(x(k)) = \hat{f}(x(k)) - f(x(k))$ and $\Delta u(k) = u(k) - v(k)$. To remove the effect of $\Delta u(k) \in \Re^m$, which can be considered as a disturbance, we generate a signal $e_\Delta(k) \in \Re^m$ as the output of a difference equation

$$e_\Delta(k+1) = l_v e_\Delta(k) + \Delta u(k) \quad (4.53)$$

with $e_\Delta(k_0) = 0$, where k_0 is the starting time instant.

Define now

$$e_u(k) = r(k) - e_\Delta(k) \quad (4.54)$$

to get

$$e_u(k+1) = l_v e_u(k) - \tilde{f}(x(k)) + d(k) \quad (4.55)$$

The auxiliary linear error system given by (4.55) is aimed at proving the stability of the filtered tracking error $r(k)$ and to take care of the effect of Δu . In the remainder of this section, (4.55) is used to focus on selecting NN-tuning algorithms that guarantee the stability of the auxiliary error $e_u(k)$. Once $e_u(k)$ is proven stable, it is required to show that the filtered tracking error system $r(k)$ is stable.

4.2.5.2 Adaptive NN Controller Structure with Saturation

The critic NN $\hat{w}_1^T(k)\phi_1(k)$ to be selected will be same as the one presented in Section 4.2.3.2. The action NN, $\hat{w}_3^T(k)\phi_3(k)$, is also similar to that in Section 4.2.3.3 without saturation except an auxiliary error signal $e_u(k)$ is now used instead of the filtered tracking error $r(k)$ to accommodate the input constraints. The procedure of obtaining the action NN weight update is very similar to that in Section 4.2.3.3 without saturation and it is given by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(k) (\hat{w}_1^T(k)\phi_1(k) + l_v e_u(k) - e_u(k+1))^T \quad (4.56)$$

where $\hat{w}_3(k) \in \Re^{n_3 \times m}$ represents the matrix of weight estimates, $\phi_3(k) \in \Re^{n_3}$ are the activation functions in the hidden layer, n_3 is the number of the nodes in the hidden layer, and the input to the action NN is given by $x(k) \in \Re^{nm}$.

4.2.5.3 Closed-Loop System Stability Analysis

Assumption 4.2.3 (Bounded Ideal Weights): Let w_3 be the unknown output-layer target NN weights for the action NN and assume that they are bounded

above so that

$$\|w_3\| \leq w_{3m} \quad (4.57)$$

where $w_{3m} \in \mathfrak{N}$ is the maximum bound on the unknown weights. Then the error in weights during estimation is given by

$$\tilde{w}_3(k) = \hat{w}_i(k) - w_3 \quad (4.58)$$

Fact 4.2.2: The activation functions are bounded by known positive values so that

$$\|\phi_3(k)\| \leq \phi_{3m} \quad (4.59)$$

where $\phi_{3m} \in \mathfrak{N}$ is the upper bound for $\phi_3(k)$.

Let the auxiliary control input, $v(k)$, be selected by (4.50) and actual control input chosen as (4.51), the auxiliary error system (4.55) is given as

$$e_u(k+1) = l_v e_u(k) - \xi_3(k) + \varepsilon_3(x(k)) + d(k) \quad (4.60)$$

where the NN estimation error is defined by

$$\xi_3(k) = \tilde{w}_3^T(k) \phi_3(k) \quad (4.61)$$

and $\varepsilon_3(x(k)) \in \mathfrak{N}^m$ is the NN reconstruction error.

Assumption 4.2.4 (Bounded NN Reconstruction Error): The NN reconstruction error $\varepsilon_3(x(k))$ is bounded over the compact set S by ε_{3m} .

The structure of the proposed adaptive critic NN controller with magnitude constraints is shown in Figure 4.9 in contrast to the one presented in Figure 4.8 where no constraints are utilized. The details of the NN controller are given in Table 4.2. The next theorem presents how to select the controller parameters and the adaptation gains so as to ensure the performance of the closed-loop tracking error dynamics is guaranteed and all the internal signals are UUB.

Theorem 4.2.3 (Adaptive NN Controller with Saturation): Consider the system given in (4.10) and control input given by (4.51). Consider the hypothesis presented in Theorem 4.2.2 along with Assumptions 4.2.3 and 4.2.4. Let the critic NN $\hat{w}_1^T(k) \phi_1(k)$ weight tuning be (3.124) and let the action generating NN $\hat{w}_3^T(k) \phi_3(k)$ weight tuning be provided by (3.152). Then the auxiliary error, $e_u(k)$ and the NN weight estimates, $\hat{w}_1(k)$, $\hat{w}_3(k)$ are UUB, with the bounds

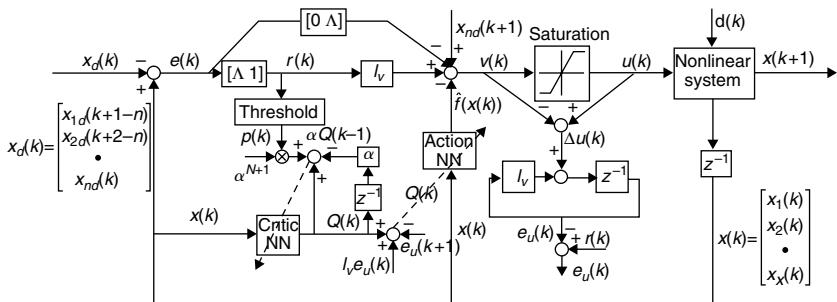


FIGURE 4.9 Adaptive critic NN controller structure with input constraints. Solid lines denote signal flow while the dashed lines represent weights tuning.

specifically given by (3.A.19) through (3.A.21) provided the design parameters are selected as (3.142), (3.144), (3.145) and

$$\alpha_3 \|\phi_3(k)\|^2 < 1 \quad (4.62)$$

Proof: See Appendix 4.B.

Remarks:

1. The critic NN weight tuning in this case is performed using the auxiliary error signal, which is obtained from the filtered tracking error and the output of the linear system driven by $\Delta u(k)$.
2. It is important to note that in this theorem CE assumption, PE condition and LIP assumptions are not used for the NN controller.

Simulation Example 4.2.1 (Adaptive NN Controller with Magnitude Constraints): The nonlinear system is described by

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \quad (4.63)$$

where $f(x(k)) = -(5/8)[x_1(k)/1 + x_2^2(k)] + 0.3x_2(k)$.

The objective is to make the state $x_2(k)$ track a reference signal using the proposed adaptive critic NN controller with input saturation. The reference

TABLE 4.2**Reinforcement Learning NN Control without Magnitude Constraints**

Define the auxiliary control input $v(k)$ as

$$v(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \cdots - \lambda_{n-1} e_2(k)$$

where $l_v \in \mathbb{R}^{m \times m}$ is a diagonal gain matrix. The actual control input is selected as

$$u(k) = \begin{cases} v(k), & \text{if } \|v(k)\| \leq u_{\max} \\ u_{\max} \operatorname{sgn}(v(k)), & \text{if } \|v(k)\| > u_{\max} \end{cases}$$

where $u_{\max} \in \mathbb{R}$ is the upper bound for the control input $u(k)$. The utility function $p(k) = [p_i(k)]_{i=1}^m \in \mathbb{R}^m$ is defined based on the current filtered tracking error $r(k)$ as

$$p_i(k) = \begin{cases} 0, & \text{if } r_i^2(k) \leq c \\ 1, & \text{if } r_i^2(k) > c \end{cases} \quad i = 1, 2, \dots, m$$

where $c \in \mathbb{R}$ is a predefined threshold. The long-term system performance measure given in terms of the strategic utility function $Q^T(k) \in \mathbb{R}^m$, is defined as

$$Q^T(k) = \alpha^N p(k+1) + \alpha^{N-1} p(k+2) + \cdots + \alpha^{k+1} p(N) + \cdots$$

where $\alpha \in \mathbb{R}$ and $0 < \alpha < 1$, and N denotes the number of stages.

Critic NN output and weight tuning:

$$\hat{Q}(k) = \hat{w}_1^T(k) \phi_1(v_1^T x(k)) = \hat{w}_1^T(k) \phi_1(k)$$

Tuning of the critic NN weights is accomplished by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(k) (\hat{w}_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1) \phi_1(k-1))^T$$

Action NN output and weight tuning:

The action NN output which is part of overall input $u(k)$ is given by

$$\hat{f}(k) = \hat{w}_3^T(k) \phi_3(v_3^T x(k)) = \hat{w}_3^T(k) \phi_3(k)$$

and the action NN weight tuning is provided by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(k) (\hat{Q}(k) + l_v e_u(k) - e_u(k+1))^T$$

where $x(k)$ is the action NN input, $\alpha_1, \alpha_2 \in \mathbb{R}$ represent learning rate parameters, $\alpha \in \mathbb{R}$ is a design parameter, and $l_v \max \in \mathbb{R}$ the maximum singular value of the gain matrix, l_v .

signal used was selected as

$$x_{2d} = \begin{cases} \sin(\omega k T + \xi), & \omega = 0.1, \quad \xi = \frac{\pi}{2}, \quad 0 \leq k \leq 3000 \\ -1, & 3000 < k \leq 4000 \text{ or } 5000 < k \leq 6000 \\ 1, & 4000 < k \leq 5000 \end{cases} \quad (4.64)$$

where the desired signal partly a sine wave and partly a unit step signal. These two different reference signals are used to evaluate the learning ability of the adaptive critic NN controller.

The sampling interval T is taken as 50 msec and a white Gaussian noise with the standard deviation of 0.005 is added to the system. The time duration is taken to be 300 sec. The unknown disturbance is taken as

$$d(k) = \begin{cases} 0, & k < 2000 \\ 1.5, & 6000 \geq k \geq 2000 \end{cases} \quad (4.65)$$

The gain of the proportional-plus derivatives (PD) controller is selected as $l_v = 0.1$ with $\lambda = 0.2$. The actuator limits for the control signal are set at 3.0 with $c = 0.0025$. Both critic $\hat{w}_1^T \phi_1(k)$ and action NNs $\hat{w}_3^T \phi_3(k)$ are selected to have ten nodes in the hidden layer. For weight updating, the learning rate is selected as $\alpha_1 = \alpha_3 = 0.1$, with parameter $\alpha = 0.5$. The initial weights are selected at random from interval $[0,1]$ and hyperbolic tangent sigmoid functions are employed. The states are initialized at zero.

Figure 4.10 illustrates the good tracking performance of the proposed adaptive critic NN controller with input saturation. The transients observed during the initial phase of the simulation are the result of online learning of NN weights where the NN is trying to learn the unknown dynamics online. In other words, the NN are not trained off-line. However, the NN learn within a short time as demonstrated in the simulation. When we introduce the unknown but bounded disturbance in the system, a large spike is observed that is indicative

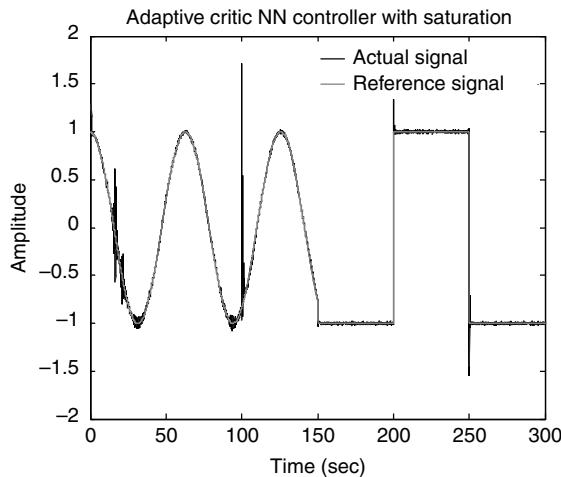


FIGURE 4.10 Performance of the NN controller.

of the disturbance. The tracking error quickly converges close to zero after the application of the disturbance, which indicates that the controller has good disturbance rejection. This phenomenon also demonstrates the good learning ability of the proposed adaptive critic NN controller. The subtle chattering observed in the tracking error, shown in Figure 4.11, is due to the presence of unknown white noise. In fact, the tracking error is close to zero except at some points where the reference signal is discontinuous. Figure 4.12 presents the

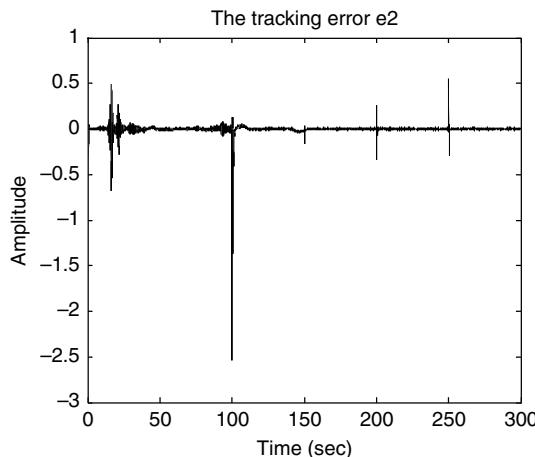


FIGURE 4.11 Tracking error.

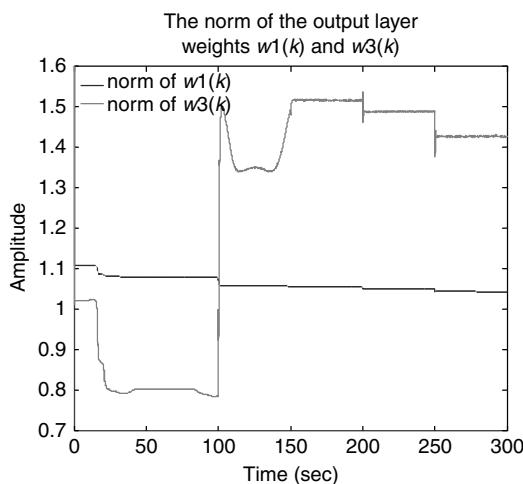


FIGURE 4.12 The norm of the NN output layer weights.

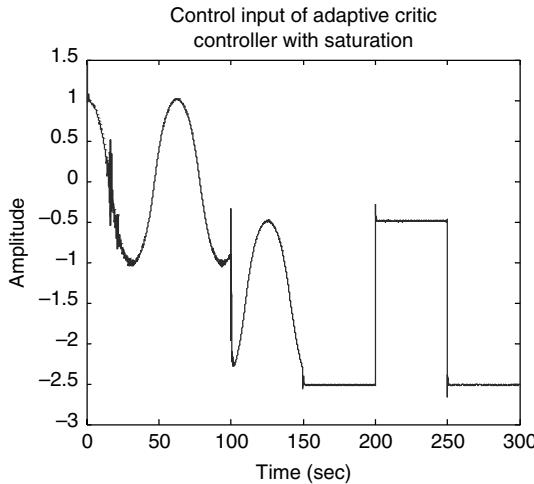


FIGURE 4.13 Control input.

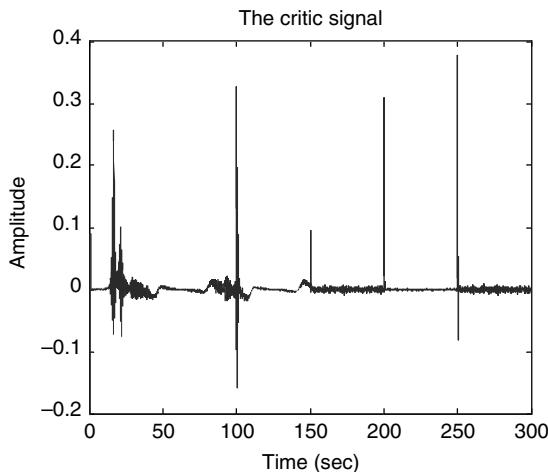


FIGURE 4.14 Critic signal.

boundedness of the norm of the output-layer weights. Figure 4.13 shows the associated NN control input where it is bounded by a magnitude of three. The critic signal is displayed in Figure 4.14.

To show the contribution of the NNs, the controller inner loop (Figure 4.15) is removed and only the outer loop is kept, which now becomes a conventional PD type controller. The outer loop controller parameters were not altered in

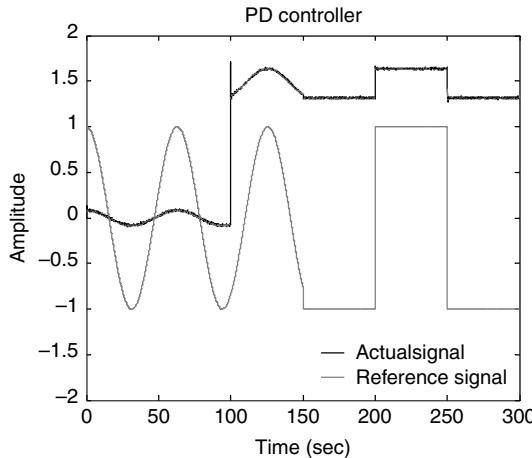


FIGURE 4.15 Controller (PD) performance without NNs.

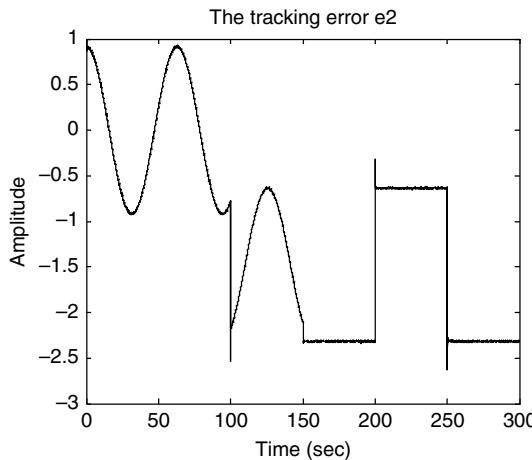


FIGURE 4.16 Tracking error.

both the cases. From Figure 4.16 and Figure 4.17, it is clear that the tracking performance has deteriorated even though the tracking error appears to be bounded. This clearly demonstrates that the NNs are able to compensate for the unknown dynamics by providing an additional compensatory signal. Moreover, the outer-loop PD controller provides a stable closed-loop system initially so that the NNs learn online. The PD control input is depicted in Figure 4.17 where it is bounded as expected.

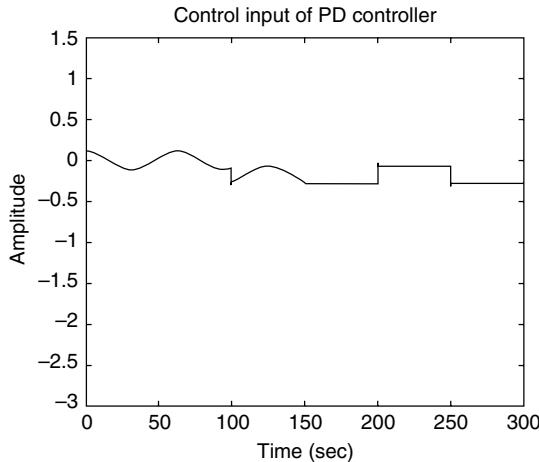


FIGURE 4.17 PD control input.

4.2.6 COMPARISON OF TRACKING ERROR AND REINFORCEMENT LEARNING-BASED CONTROLS DESIGN

Neural network control architectures and learning schemes are discussed in detail in Chapter 1. Here it is important to discuss how standard adaptive critic NN control architectures are modified to suit control purposes. A feedforward NN is used to approximate the dynamics of certain nonlinear functions in (4.10) and the weights are tuned online by using the instantaneous tracking error in the case of tracking error-based adaptive NN controller. By contrast, the critic NN approximates a certain strategic utility function, which is taken as the long-term performance measure of the system. The action NN weights are tuned by both the critic NN signal and the filtered tracking error to minimize the strategic utility function and uncertain system dynamic estimation error so that the action NN can generate a more effective control signal. By contrast, the available critic schemes use a standard Bellman equation. Here the action NN control signal combined with an additional outer-loop conventional control signal is applied as the overall control input to the nonlinear discrete-time system. The outer-loop conventional signal allows the action and critic NN to learn online while making the system stable. It is treated as the additional evaluation feedback signal from the supervisor (Rosenstein and Barto 2004). By selecting appropriate objective functions for both critic and action NNs, the closed-loop stability is inferred.

The proposed critic NN architecture overcomes the limitation of using the tracking error at one step ahead by minimizing a certain long-term performance measure. However, two NN are utilized in adaptive critic NN architectures. In the recent work, the action NN is not used (see Chapter 9), reducing

the number of NN used to one. In any case, feedforward NNs are used as building blocks in both the NN control architectures and gradient-based adaptation is used to derive the weight-updating rules in reinforcement learning-based NN in contrast with the tracking error-based NN controller.

It is very important to note that in the proposed reinforcement learning-based control, the action NN output is added with a standard outer-loop conventional control signal and applied to the system whereas in standard adaptive critic NN control (Prokhorov and Wunsch 1997; Lin and Balakrishnan 2000), the output of the action NN is the actual control signal. This outer loop conventional signal is treated as the evaluation signal from the supervisor to the actor. Moreover, the strategic utility function and tuning rules are different between the proposed reinforcement-based NN controller (He and Jagannathan 2003) and other works (Prokhorov and Wunsch 1997; Lin and Balakrishnan 2000).

Consequently, Lyapunov-based stability analysis is possible for the closed-loop system with the proposed controllers whereas it is difficult to do any type of stability analysis with available adaptive critic NN control architectures (Prokhorov and Wunsch 1997). Out of many available adaptive critic NN works, convergence analysis is given only in Lin and Balakrishnan (2000) and Si and Wang (2001) and that too not complete since in Lin and Balakrishnan, a linear system is considered whereas in Si and Wang, convergence is shown using average sense. Finally, it would be interesting to do a detailed study of using two NNs and the performance improvement using these controllers applied to industrial processes when compared to tracking error-based NN controllers. Next, we present the compensation of unknown deadzone for uncertain nonlinear discrete-time systems while accommodating the actuator constraints.

4.3 UNCERTAIN NONLINEAR SYSTEM WITH UNKNOWN DEADZONE AND SATURATION NONLINEARITIES

Nonlinear systems, for instance robot manipulators and high-power machinery, often have actuator nonlinearities such as deadzones and saturation. Background on the deadzone nonlinearity, which is shown in Figure 4.2, is discussed in Section 4.1. Standard control systems, such as PD, have been observed to result in limit cycles if the actuators have deadzones. The effects of deadzone are deleterious in modern processes where precise motion and extreme speeds are needed. Standard techniques for overcoming deadzones include variable structure control (Utkin 1978) and dithering (Desoer and Shahruz 1986).

Recently, in seminal work, several rigorously derived adaptive control schemes have been given for deadzone compensation (Tao and Kokotovic 1996). Compensation for nonsymmetric deadzones was considered for

unknown linear systems in Tao and Kokotovic (1995) and for nonlinear systems in Brunovsky form in Recker et al. (1991). Nonlinear Brunovsky form systems with unknown dynamics were treated in Tian and Tao (1996), where a backstepping approach was used. All of the known approaches to deadzone compensation using adaptive control techniques assume that the deadzone function can be linearly parameterized using a few parameters such as the deadzone width, slope, and so on. However, deadzones in industrial systems may not be linearly parameterizable.

On the other hand, intelligent control techniques based on NN (Selmic and Lewis 2000) and fuzzy logic systems (Campos and Lewis 1999) have recently shown promise in effectively compensating for the effects of deadzone. NN have been used extensively in feedback control systems. Most applications are ad hoc with no demonstrations of stability. The stability proofs that do exist rely almost invariably on the universal approximation property for NN (Jagannathan and Lewis 1996; Lewis et al. 1999). However, to compensate for deadzone, which is discontinuous at the origin, the deadzone inverse function must be estimated. Typically, smooth activation functions for the many hidden-layer neurons are used to approximate jump functions. Augmented jump functions are used to approximate deadzones in Selmic and Lewis (2000) by assuming that the uncertain nonlinear dynamics are bounded by a known upper bound. Since it is very difficult to accurately obtain the upper bound for many unknown nonlinear systems, this assumption will be relaxed.

In this section, we show how to design an NN controller for deadzone compensation even when the nonlinear dynamics of the system are uncertain (He et al. 2002; He and Jagannathan 2003). Therefore, the proposed work significantly differs from others due to the following reasons: Simultaneous compensation of unknown nonsymmetric time-varying deadzones for uncertain nonlinear systems with magnitude constraints in discrete-time is dealt with in this chapter compared to constant deadzone nonlinearity in Tao and Kokotovic (1996). Since it is very difficult to accurately obtain the upper bound for many unknown nonlinear systems (Lewis et al. 1999), this assumption is relaxed in He and Jagannathan (2003) in contrast with Selmic and Lewis (2000) and Campos and Lewis (1999). Finally, in the proposed NN controller architecture, the future tracking performance is assessed by the critic NN based on a utility function defined using past tracking errors while guaranteeing the performance via Lyapunov stability. No utility function is employed in the past works (Tao and Kokotovic 1994, 1995; Selmic and Lewis 2003).

Neural network architectures and learning methods are discussed in Miller et al. (1991) and White and Sofage (1992). NN learning methods have been divided into three main paradigms: unsupervised learning, supervised learning, and reinforcement learning. The unsupervised learning scheme does not require an external teacher to guide the learning process. Instead, the teacher

is built into the learning method. Unlike the supervised learning scheme, both supervised and reinforcement learning paradigms require an external teacher to provide training signals that guide the learning process. The difference between these two paradigms arises from the kind of information about the local characteristics of the performance surface that is available to the learning system.

In supervised learning, deviation from the acceptable performance is available all the time whereas in reinforcement learning, the role of the teacher is more evaluative than instructional. Instead of receiving detailed deviation of performance during learning, the learning system receives only information about the current value of system performance measure. This measure does not itself indicate how the learning system should change its behavior to improve performance; there is no directed information. Because detailed knowledge of the controlled system and its behavior is not needed, reinforcement learning is potentially one of the most useful NN approaches to feedback control system. An adaptive critic NN controller design using an index similar to the Bellman equation is described in the previous section. In this section, a simplified performance index will be selected.

Adaptive critics have been used in an ad hoc fashion in NN control. No proofs acceptable to the control community have been offered for the performance of this important structure for feedback control. In standard control theory applications with NNs, the NN was used in the action-generating loop (i.e., basic feedback loop) to control a system. Tuning laws for that case were given that guarantee stability and performance. However a high-level critic was not used. Here is offered a rigorous analysis of the adaptive critic structure, including structural modifications and tuning algorithms required to guarantee closed-loop stability.

Therefore, a novel reinforcement learning-based neural network (RLNN) controller in discrete-time (He et al. 2002) is designed to deliver a desired tracking performance for a class of uncertain nonlinear systems plus unknown actuator deadzone with input magnitude constraints. The RLNN controller consists of three NNs: an action-generating NN for compensating the unknown deadzones, a second action-generating NN for compensating the uncertain nonlinear system dynamics, and a critic NN to tune the weights of the action-generating NNs. The magnitude constraints on the input are modeled as saturation nonlinearities and they are dealt with in the Lyapunov-based controller design. The UUB of the closed-loop tracking and the NN weights estimation errors is demonstrated. This clearly demonstrates that multiple nonlinearities can be compensated simultaneously by using several NNs. The NN learning is performed online as the system is controlled, with no off-line learning phase required. Closed-loop performance is guaranteed through the learning algorithms proposed.

4.3.1 NONLINEAR SYSTEM DESCRIPTION AND ERROR DYNAMICS

Consider the following nonlinear system, to be controlled, given in the following form:

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + u(k) + d'(k) \end{aligned} \tag{4.66}$$

where $x(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T \in R^{nm}$ with each $x_i(k) \in R^m$; $i = 1, \dots, n$ is the state at time instant k , $f(x(k)) \in R^m$ is the unknown nonlinear dynamics of the system, $u(k) \in R^m$ is the input, and $d'(k) \in R^m$ is the unknown but bounded disturbance, whose bound is given by $\|d'(k)\| \leq d'_m$.

Definition 4.3.1 (Tracking Errors): Given a desired trajectory, $x_d(k) \in R^m$, and its past values, the tracking errors are defined as

$$e_i(k) = x_i(k) - x_d(k+i-n) \tag{4.67}$$

with each error $e_i(k) \in R^m$. Combining (4.66) and (4.67), the error system is given by

$$\begin{aligned} e_1(k+1) &= e_2(k) \\ &\vdots \\ e_n(k+1) &= f(x(k)) - x_d(k+1) + u(k) + d'(k) \end{aligned} \tag{4.68}$$

4.3.2 DEADZONE COMPENSATION WITH MAGNITUDE CONSTRAINTS

In this section, magnitude constraints are asserted for the actuator and the NN controller is presented next.

4.3.2.1 Deadzone Nonlinearity

The time-varying deadzone nonlinearity is displayed in Figure 4.18. If $\tau(k)$ and $q(k)$ are scalars, the time-varying deadzone nonlinearity is given by

$$q(k) = h(\tau(k)) = \begin{cases} f_1(\tau(k)), & \tau(k) > b_+(k) \\ 0, & -b_-(k) \leq \tau(k) \leq b_+(k) \\ f_2(\tau(k)), & \tau(k) \leq -b_-(k) \end{cases} \tag{4.69}$$

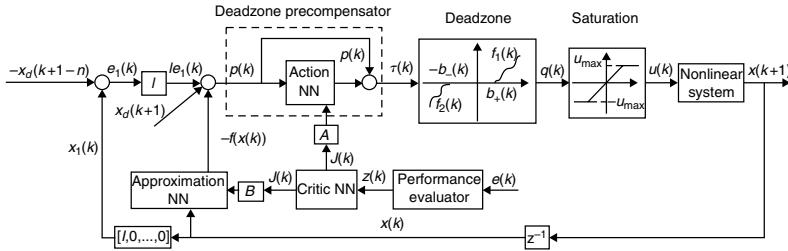


FIGURE 4.18 NN controller with unknown input deadzones and magnitude constraints.

where $b_+(k)$ and $b_-(k)$ are positive time-varying scalars, and $f_1(\tau(k)), f_2(\tau(k))$ are nonlinear functions. To compensate the deadzone nonlinearity, its inverse is required.

Assumption 4.3.1: Both $f_1(\tau(k))$ and $f_2(\tau(k))$ are smooth, continuous, and invertible functions.

Note: The above assumptions imply that $f_1(\tau(k))$ and $f_2(\tau(k))$ are both increasing and decreasing nonlinear functions. In other words, $h(\tau(k))$ is either a nondecreasing or a nonincreasing function.

With Assumption 4.3.1, the inverse time-varying deadzone function, $h^{-1}(q(k))$, is now given by

$$\tau(k) = h^{-1}(q(k)) = \begin{cases} f_1^{-1}(q(k)), & q(k) > 0 \\ 0, & q(k) = 0 \\ f_2^{-1}(q(k)), & q(k) < 0 \end{cases} \quad (4.70)$$

4.3.2.2 Compensation of Deadzone Nonlinearity

To offset the deleterious effects of deadzones, a precompensator displayed in Figure 4.18 is proposed (Tao and Kokotovic 1996). The desired objective of the precompensator is to make the throughput from $p(k)$ to $q(k)$ equal to unity. Here, $p(k) \in R^m$, $\tau(k) \in R^m$, and $q(k) \in R^m$ are vectors.

The precompensator consists of two parts (Tao and Kokotovic 1996): a linear part, $p(k)$, designed to achieve the tracking of the reference signal, and an action-generating NN part, which is used to cancel the deadzone by approximating the nonlinear function $h^{-1}(p(k)) - p(k)$. In other words,

$$h^{-1}(p(k)) - p(k) = w_1^\top \phi_1(v_1^\top p(k)) + \varepsilon_1(p(k)) = w_1^\top \phi_1(p(k)) + \varepsilon_1(p(k)) \quad (4.71)$$

where $w_1 \in R^{n_1 \times m}$ and $v_1 \in R^{m \times n_1}$ are the target weights, $\phi_1(\cdot)$ is the activation function and $\varepsilon_1(k)$ is the NN reconstruction error, with n_1 the number of hidden-layer nodes. According to Igelnik and Pao (1995), a single-layer NN can be used to approximate any nonlinear continuous function over the compact set when the input-layer weights are selected at random and held constant whereas the output-layer weights are only tuned provided sufficiently large number of nodes in the hidden layer is chosen. Therefore, a single-layer NN is employed here whose output is defined as $\hat{w}_1^T(k)\phi_1(v_1^T p(k))$. For simplicity, it is expressed as $\hat{w}_1^T(k)\phi_1(p(k))$, with $\hat{w}_1(k) \in R^{n_1 \times m}$ being the actual output-layer weights. A total of three single-layer NNs will be used whereas the fourth one is only meant for the analysis of the throughput error in Theorem 4.3.1 and it is not used in the NN controller design.

Definition 4.3.2: The weight estimation errors of all the NN are defined as

$$\tilde{w}_i(k) = \hat{w}_i(k) - w_i \quad i = 1, 2, 3, 4 \quad (4.72)$$

Moreover, for convenience, define

$$\xi_i(k) = \hat{w}_i^T(k)\phi_i(k) \quad i = 1, 2, 3, 4 \quad (4.73)$$

The deadzone function, $h(\cdot)$, defined in (4.69), is approximated by using a single-layer NN as

$$h(k) = w_4^T\phi_4(k) + \varepsilon_4(k) \quad (4.74)$$

where $w_4 \in R^{n_4 \times m}$ is the target output-layer weight matrix with n_4 the number of hidden-layer nodes.

Assumption 4.3.2 (Activation Functions): The activation functions of all the NNs are continuously differentiable over a compact set S , and thus its j th derivatives $\phi_i^{(j)}(k)$, $i = 1, 2, 3, 4$ and $j = 0, 1, \dots, n, \dots$, are all bounded over the compact set S .

Assumption 4.3.2 is a mild assumption since many NN activation functions, for instance, hyperbolic tangent sigmoid functions, satisfy it. Using Assumption 4.3.2, the following fact can be stated

$$\|\phi_i(k)\| \leq \phi_{im} \quad i = 1, 2, 3, 4 \quad (4.75)$$

Assumption 4.3.3 (Bounded Ideal Weights): The Frobenius norm (Lewis et al. 1999) of the target weight matrix for all the NNs is bounded by known positive

values so that

$$\|w_i\| \leq w_{im} \quad i = 1, 2, 3, 4 \quad (4.76)$$

The next theorem shows that when the NN weight estimation and the reconstruction errors of the NN precompensator become zero, the throughput error, $q(k) - p(k)$ approaches zero.

Theorem 4.3.1 (Throughput Error): Using Figure 4.18, the throughput of the compensator plus the deadzone is given by

$$\begin{aligned} q(k) &= p(k) + g(k)\xi_1(k) - g(k)\varepsilon_1(p(k)) + O(\xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \\ &\quad - \varepsilon_4(h^{-1}(p(k))) \end{aligned} \quad (4.77)$$

with $\xi_1(k)$ defined in (4.73), $O(\xi_1(k) - \varepsilon_1(p(k)))$ called as the Lagrange remainder after two terms, and $\varepsilon_1(p(k))$, $\varepsilon_4(\tau(k))$, and $\varepsilon_4(h^{-1}(p(k)))$ are the NN reconstruction errors. The $g(k)$ is defined as

$$\begin{aligned} g(k) &= w_4^T \phi'_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) \\ &= \frac{w_4^T d(\phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)))}{d(p(k))} \end{aligned} \quad (4.78)$$

where it is bounded over the compact set, S , whose bound is given by

$$\|g(k)\| \leq g_m \quad (4.79)$$

Proof: The proof is similar to Selmic and Lewis (2000) except the inclusion of higher-order terms in the proof. See Appendix 4.C.

Remarks: Theorem 4.3.1 is of a more general case than that in Selmic and Lewis (2000) where the higher-order terms are ignored. In this chapter, they are considered bounded and incorporated in the proof.

4.3.2.3 Saturation Nonlinearities

The actuator limits can be modeled as saturation nonlinearities with limits defined as u_{\max} . Using Figure 4.18, actuator constraints are expressed as

$$u(k) = \begin{cases} q(k), & \|u(k)\| \leq u_{\max} \\ u_{\max} \text{sgn}(q(k)), & \|u(k)\| > u_{\max} \end{cases} \quad (4.80)$$

with $\text{sgn}(\cdot)$ the sign function.

4.3.3 REINFORCEMENT LEARNING NN CONTROLLER DESIGN

The control objective is to make the system errors, $e_i(k)$, $i = 1, \dots, n$, small with all the internal signals UUB (Lewis et al. 1999). The proposed RLNN controller consists of three NNs: two action-generating NNs with a third critic NN used for tuning the action-generating NNs and its structure is depicted in Figure 4.19.

4.3.3.1 Error Dynamics

Case I: $\|u(k)\| \leq u_{\max}$

Using (4.80), that is, $u(k) = q(k)$ and combining with error systems (4.68) and (4.77), we obtain

$$\begin{aligned} e_n(k+1) &= f(x(k)) - x_d(k+1) + p(k) + g(k)(\xi_1(k) - \varepsilon_1(p(k))) \\ &\quad + O(\xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) - \varepsilon_4(h^{-1}(p(k))) + d'(k) \end{aligned} \quad (4.81)$$

A single-layer NN will be used to approximate the uncertain nonlinear dynamics, $f(x(k))$, as

$$\begin{aligned} f(x(k)) &= w_2^T \phi_2(v_2^T x(k)) + \varepsilon_2(x(k)) \\ \hat{f}(x(k)) &= \hat{w}_2^T(k) \phi_2(v_2^T x(k)) = \hat{w}_2^T(k) \phi_2(k) \end{aligned} \quad (4.82)$$

where $w_2 \in R^{n_2 \times m}$ and $v_2 \in R^{nm \times n_2}$ are the target weights, $\hat{w}_2(k) \in R^{n_2 \times m}$ is the actual weight matrix, with n_2 being the number of hidden-layer nodes. Choose

$$p(k) = le_1(k) - \hat{w}_2^T(k) \phi_2(k) + x_d(k+1) \quad (4.83)$$

with $l \in R^{m \times m}$ being the gain matrix. The error $e_n(k+1)$ can be rewritten using (4.81) through (4.83) as

$$e_n(k+1) = le_1(k) + g(k)\xi_1(k) - \xi_2(k) + d_1(k) \quad (4.84)$$

where $\xi_2(k)$ is defined in (4.73), and $d_1(k)$ is defined as

$$\begin{aligned} d_1(k) &= \varepsilon_2(x(k)) - g(k)\varepsilon_1(p(k)) + O(\xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \\ &\quad - \varepsilon_4(h^{-1}(p(k))) + d'(k) \end{aligned} \quad (4.85)$$

Note in (4.85), $d_1(k)$ is bounded above by d_{1m} in the compact set S due to the fact that $\varepsilon_2(x(k))$, $g(k)$, $O(\xi_1(k) - \varepsilon_1(p(k)))$, $\varepsilon_4(\tau(k))$, $\varepsilon_4(h^{-1}(p(k)))$, and

$d'(k)$ are bounded. The error system (4.68) for Case I can be rewritten as

$$\begin{aligned} e_1(k+1) &= e_2(k) \\ &\vdots \\ e_n(k+1) &= le_1(k) + g(k)\xi_1(k) - \xi_2(k) + d_1(k) \end{aligned} \quad (4.86)$$

Case II: $\|u(k)\| > u_{\max}$

In this case, $p(k)$ is still defined as (4.83). Using (4.80), taking $u(k) = u_{\max}\text{sgn}(q(k))$ and substituting into (4.68)

$$e_n(k+1) = f(x(k)) - x_d(k+1) + u_{\max}\text{sgn}(q(k)) + d'(k) \quad (4.87)$$

Combining (4.82) and (4.87) to get

$$e_n(k+1) = w_2^T \phi_2(v_2^T x(k)) + \varepsilon_2(x(k)) - x_d(k+1) + u_{\max}\text{sgn}(q(k)) + d'(k) \quad (4.88)$$

Let us denote

$$d_2(k) = \varepsilon_2(x(k)) - x_d(k+1) + u_{\max}\text{sgn}(q(k)) + d'(k) \quad (4.89)$$

Equation 4.88 is simplified as

$$e_n(k+1) = w_2^T \phi_2(v_2^T x(k)) + d_2(k) \quad (4.90)$$

where the term $d_2(k)$ is bounded by d_{2m} over the compact set S given the boundedness of $\varepsilon_2(x(k))$, $x_d(k+1)$, u_{\max} , $\text{sgn}(q(k))$, and $d'(k)$. Using (4.90), the error system (4.68) becomes

$$\begin{aligned} e_1(k+1) &= e_2(k) \\ &\vdots \\ e_n(k+1) &= w_2^T \varphi(v_2^T x(k)) + d_2(k) \end{aligned} \quad (4.91)$$

4.3.3.2 Critic NN Design

Define a quadratic performance index in terms of the tracking errors as the critic NN input

$$z(k) = \sum_{j=1}^k (e^T(j)Pe(j)) \quad (4.92)$$

where $e(k) = [e_1(k), \dots, e_n(k)]^T \in R^{nm \times 1}$, $P \in R^{nm \times nm}$ is the positive definite weighting matrix, and $z(k) \in R$ is the input to the critic NN. A choice of the critic NN signal is given by

$$J(k) = \hat{w}_3^T(k)\phi_3(v_3^T z(k)) = \hat{w}_3^T(k)\phi_3(k) \quad (4.93)$$

where $\hat{w}_3(k) \in R^{n_3 \times m}$, and n_3 is the number of hidden layer nodes. This utility function defines the performance of the system over time. The critic signal, $J(k) \in R^m$ by using the index provides an additional corrective action based on current and past performance. This information along with the tracking error in the subsequent step is used to tune the action-generating NNs. The critic NN signal can be viewed as a look-ahead factor, which is determined based on past performance. Future research work will include minimization of the quadratic index in (4.92) by employing the proposed adaptive critic NN controller given in Table 4.3. The structure of the proposed NN controller is depicted in Figure 4.18. The design parameter matrices $A \in R^{m \times m}$ and $B \in R^{m \times m}$ are given in Theorem 4.3.2.

4.3.3.3 Main Result

To prove the next theorem, we need the following assumption.

Assumption 4.3.4 (Bounded Lagrange Remainder): The Lagrange remainder after two terms $O(\xi_1(k) - \varepsilon_1(p(k)))$ is bounded over the compact set S .

Theorem 4.3.2: Consider the system given in (4.66), the input deadzones (4.69), and the input constraints (4.80), and let the Assumptions 4.3.1 through 4.3.4 hold. Let the NN reconstruction errors, $\varepsilon_i(\cdot)$, $i = 1, 2, 3, 4$, disturbances $d'(k)$, the desired trajectory, $x_d(k)$, and its past values be bounded. Let the action NN weight tuning be given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1\phi_1(k)(\hat{w}_1^T(k)\phi_1(k) + Cle_1(k) + AJ(k))^T \quad (4.94)$$

with the action NN weight tuning provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2\phi_2(k)(\hat{w}_2^T(k)\phi_2(k) + Dle_1(k) + BJ(k))^T \quad (4.95)$$

and the critic NN weights be tuned by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3\phi_3(k)(J(k) + Ele_1(k))^T \quad (4.96)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$, $A \in R^{m \times m}$, $B \in R^{m \times m}$, $C \in R^{m \times m}$, $D \in R^{m \times m}$, and $E \in R^{m \times m}$ are design parameters. Consider the deadzone compensator $\tau(k) = p(k) + w_1^T(k)\phi_1(k)$ with $p(k)$ given by (4.83). The tracking errors in (4.68) and the NN weights $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are UUB provided the

TABLE 4.3
Reinforcement Learning NN Controller for Nonlinear Systems with Deadzones

The control input is given by

$$u(k) = \begin{cases} q(k), & \|u(k)\| \leq u_{\max} \\ u_{\max} \text{sgn}(q(k)), & \|u(k)\| > u_{\max} \end{cases}$$

where $q(k)$ = deadzone($\tau(k)$). Consider the deadzone compensator input $\tau(k) = p(k) + w_1^T(k)\phi_1(k)$ with a first action NN used to compensate for the deadzone and $p(k)$ is given by

$$p(k) = le_1(k) - \hat{w}_2^T(k)\phi_2(k) + x_d(k+1)$$

where $l \in R^{m \times m}$ is the gain matrix, and $\hat{f}(x(k))$ is an approximation of the uncertain nonlinear dynamics, $f(x(k))$, of the nonlinear discrete-time system given by a second action NN

$$\hat{f}(x(k)) = \hat{w}_2^T(k)\phi_2(v_2^T x(k)) = \hat{w}_2^T(k)\phi_2(k)$$

Define a quadratic performance index in terms of the tracking errors as the critic NN input

$$z(k) = \sum_{j=1}^k (e^T(j)Pe(j))$$

where $e(k) = [e_1(k), \dots, e_n(k)]^T \in R^{nm \times 1}$, $P \in R^{nm \times nm}$ is the positive definite weighting matrix, and $z(k) \in R$ is the input to the critic NN. A choice of the critic NN signal is given as

$$J(k) = \hat{w}_3^T(k)\phi_3(v_3^T z(k)) = \hat{w}_3^T(k)\phi_3(k)$$

Action and critic NN weight tuning:

Let the action NN weight tuning for the first NN be given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(k)(\hat{w}_1^T(k)\phi_1(k) + Cle_1(k) + AJ(k))^T$$

with the action NN weight tuning for the second NN be provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(k)(\hat{w}_2^T(k)\phi_2(k) + Dle_1(k) + BJ(k))^T$$

and the critic NN weights be tuned by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(k)(J(k) + Ele_1(k))^T$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$, $A \in R^{m \times m}$, $B \in R^{m \times m}$, $C \in R^{m \times m}$, $D \in R^{m \times m}$, and $E \in R^{m \times m}$ are design parameters.

design parameters are selected as

$$(1) 0 < \alpha_i \|\phi_i(k)\|^2 < 1 \quad i = 1, 2, 3 \quad (4.97)$$

$$(2) |l_{\max}| < \min\left(\frac{1}{2\sqrt{2}g_m}, 1\right) \quad (4.98)$$

$$(3) \|A\|^2 + \|B\|^2 < \frac{1}{6}, \quad 3\|C\|^2 + 3\|D\|^2 + 2\|E\|^2 < \frac{1}{2} \quad (4.99)$$

where l_{\max} is the maximum eigenvalue of the gain matrix l .

Proof: See Appendix 4.C.

Remarks:

1. To verify (4.97), let the number of hidden-layer neurons in the first action NN be n_1 with the hyperbolic tangent sigmoid function as its activation function, then it follows that $\|\phi_1(\cdot)\|^2 \leq n_1$. The NN learning rate α_1 can be selected as $0 < \alpha_1 < (1/n_1)$ to satisfy (4.97). A similar analysis can be performed to obtain the NN learning rates α_2 and α_3 .
2. No information is currently available to determine the number of the hidden-layer nodes for a multilayer NN. However, the number of hidden-layer neurons required for suitable approximation can be addressed by using the closed-loop system stability and the NN error bounds. From (4.97) and Remark 1, for closed-loop system stability, the numbers of the hidden-layer nodes have to satisfy $n_1 > 1/\alpha_1$, $n_2 > 1/\alpha_2$, $n_3 > 1/\alpha_3$, once the NN learning rate parameters α_1 , α_2 , and α_3 are selected. With regards to the error bounds for NN, according to Igelnik and Pao (1995), given sufficient number of hidden-layer neurons, n , the reconstruction error, $\varepsilon(k)$, approaches zero. If the continuous function is restricted to satisfy the Lipschitz condition, then the reconstruction error of order $O(C/\sqrt{n})$ achieved with C is independent of n (Igelnik and Pao 1995).
3. Condition (3) in (4.99) can be verified easily.

Simulation Example 4.3.1 (Adaptive NN Controller with Magnitude Constraints and Deadzone Compensator): Consider the following nonlinear system

$$x_1(k+1) = x_2(k), \quad x_2(k+1) = f(x(k)) + u(k) \quad (4.100)$$

where the uncertain nonlinear system dynamic is $f(x(k)) = -5/8[x_1(k)/1 + x_2^2(k)] + 0.65x_2(k)$. The unknown deadzone widths are selected as $b_+(k) = 0.4(1 + 0.1 \sin(kT))$, $b_-(k) = 0.3(1 + 0.1 \cos(kT))$ with the sampling interval, T , given as 50 msec. A reference signal used was selected as

$$x_d(k) = \sin(\omega kT + \zeta) \quad \omega = 0.1, \quad \zeta = \frac{\pi}{2} \quad (4.101)$$

The controller gain, l , is taken as -0.9 . The upper bound on the control input is chosen as $u_{\max} = 0.9$. Parameters are selected as $A = 0.5, B = 0.1, C = 1, D = 1, E = 1$, and $P = I$, where I is the 2×2 identity matrix. The learning rate is selected as $\alpha_1 = \alpha_2 = \alpha_3 = 0.1$. All three NNs each have 11 nodes in the hidden layer so that (4.97) can be satisfied. All the initial weights are

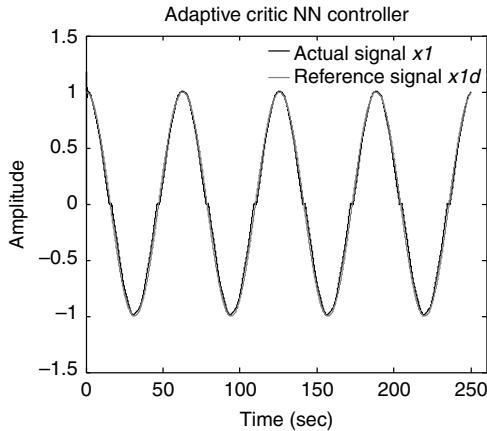


FIGURE 4.19 NN controller with deadzone compensator.

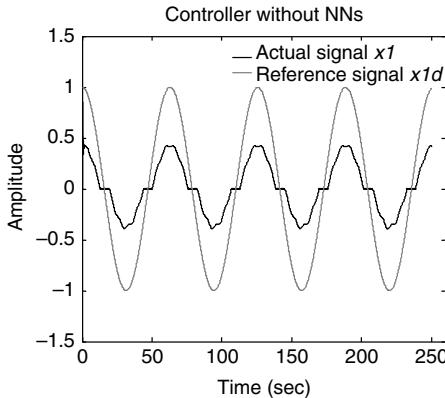


FIGURE 4.20 PD controller without compensator.

selected at random from a normal Gaussian distribution and all the activation functions are hyperbolic tangent sigmoid functions. From Figure 4.19, the tracking performance is quite good with the proposed NN controller. Figure 4.20 illustrates that the system tracking performance is not satisfactory when the NNs are not used. The controller gain l is not changed in both the cases.

4.4 ADAPTIVE NN CONTROL OF NONLINEAR SYSTEM WITH UNKNOWN BACKLASH

Similar to adaptive deadzone compensators, adaptive control approaches for backlash compensation in discrete-time were presented in Grundelius and

Angelli (1996) and Tao and Kokotovic (1995, 1996). These all require an LIP assumption, which may not hold for actual industrial motion systems. Since backlash is a dynamic nonlinearity, a backlash compensator should also be dynamic. We intend to use discrete-time dynamic inversion in this section to design a discrete-time backlash compensator. Dynamic inversion is a form of backstepping. Backstepping was extended in discrete-time by Jagannathan (1997, 2001), Brynes and Lin (1994), and Yeh and Kokotovic (1995). The difficulty with discrete-time dynamic inversion is that a future value of a certain ideal control input is needed. This section presents how this problem is confronted in Lewis et al. (2002) and it is discussed in this section.

In this section, a complete solution for extending dynamic inversion to the discrete-time case by using a filtered prediction approach is presented for the case of nonsymmetric backlash compensation. A rigorous design procedure is discussed that results in a PD tracking loop with an adaptive NN in the feedforward loop for dynamic inversion of the backlash nonlinearity. The NN feedforward compensator is adapted in such a way as to estimate the backlash inverse online. No PE, LIP, and CE assumptions are needed.

4.4.1 NONLINEAR SYSTEM DESCRIPTION

Recall the following nonlinear system, to be controlled, given in the following form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + \tau(k) + d(k) \end{aligned} \tag{4.102}$$

where $x(k) = [x_1(k), x_2(k), \dots, x_n(k)]^T \in \Re^{nm}$ with each $x_i(k) \in \Re^m$; $i = 1, \dots, n$ is the state at time instant k , $f(x(k)) \in \Re^m$ is the unknown nonlinear dynamics of the system, $u(k) \in \Re^m$ is the input, and $d(k) \in \Re^m$ is the unknown but bounded disturbance, whose bound is assumed to be a known constant, $\|d(k)\| \leq d_m$. The actuator output $\tau(k)$ is related to the control input $u(k)$ through the backlash nonlinearity $\tau(k) = \text{Backlash}(u(k))$.

Given a trajectory, $x_{nd}(k) \in \Re^m$, and its past values, define the tracking error

$$e_i(k) = x_i(k) - x_{nd}(k+i-n) \tag{4.103}$$

and the filtered tracking error, $r(k) \in \Re^m$, as

$$r(k) = [\lambda \ I]e(k) \tag{4.104}$$

with $e(k) = [e_1(k), e_2(k), \dots, e_n(k)]^T$, $e_1(k+1) = e_2(k)$, where $e_1(k+1)$ is the next future value for the error $e_1(k)$, $e_{n-1}(k), \dots, e_1(k)$ are past values of the error $e_n(k)$, $I \in \Re^{m \times m}$ is an identity matrix, and $\lambda = [\lambda_{n-1}, \lambda_{n-2}, \dots, \lambda_1] \in \Re^{m \times (n-1)m}$ is a constant diagonal positive definite matrix selected such that the eigenvalues are within a unit disc. Consequently, if the filtered tracking error $r(k)$ tends to zero, then all the tracking errors go to zero. Equation 4.12 can be expressed as

$$r(k+1) = f(x(k)) - x_{nd}(k+1) + \lambda_1 e_n(k) + \dots + \lambda_{n-1} e_2(k) + \tau(k) + d(k) \quad (4.105)$$

The control objective is to make all the tracking errors bounded close to zero and all the internal signals UUB. In order to proceed, the following standard assumptions are needed.

Assumption 4.4.1 (Bounded Estimation Error): The nonlinear function is assumed to be unknown, but a fixed estimate $\hat{f}(x(k))$ is assumed known, such that the function estimation error $\tilde{f}(x(k)) = \hat{f}(x(k)) - f(x(k))$ satisfies $\|\tilde{f}(x(k))\| \leq f_M(x(k))$ for some known bounding function $f_M(x(k))$.

Assumption 4.4.2 (Bounded Desired Trajectory): The desired trajectory is bounded in the sense that

$$\begin{aligned} & \left\| \begin{array}{c} x_{1d}(k) \\ x_{2d}(k) \\ \vdots \\ x_{nd}(k) \end{array} \right\| \leq x_d \end{aligned}$$

for a known bound x_d .

4.4.2 CONTROLLER DESIGN USING FILTERED TRACKING ERROR WITHOUT BACKLASH NONLINEARITY

The discrete-time NN backlash compensator is to be designed using the back-stepping technique. First, we will design a compensator that guarantees system trajectory tracking when there is no backlash. The control input when there is no backlash is defined as $\tau_{des}(k)$.

Define the control input $\tau_{des}(k) \in \Re^m$ as

$$\tau_{des}(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) \quad (4.106)$$

where $\hat{f}(x(k)) \in R^m$ is an estimate of the unknown function $f(x(k))$, and $l_v \in R^{m \times m}$ is a diagonal gain matrix. Then, the closed-loop system becomes

$$r(k+1) = l_v r(k) - \tilde{f}(x(k)) + d(k) \quad (4.107)$$

where the functional estimation error is given by $\tilde{f}(x(k)) = \hat{f}(x(k)) - f(x(k))$.

Equation 4.107 relates the filtered tracking error with the functional estimation error and the filtered tracking error system (4.107) can also be expressed as

$$r(k+1) = l_v r(k) + \delta_0(k) \quad (4.108)$$

where $\delta_0(k) = -\tilde{f}(x(k)) + d(k)$. If the functional estimation error $\tilde{f}(x(k))$ is bounded above such that $\|\tilde{f}(x(k))\| \leq f_M$, for some known bounding function $f_M \in \Re$ then next stability results hold.

Theorem 4.4.1 (*Control Law for Outer Tracking Loop without Backlash*): Consider the system given by (4.102) and assume that there is no backlash nonlinearity. Let the control action be provided by (4.106). Assume the functional estimation error and the unknown disturbance to be bounded. The filtered tracking error system (4.108) is stable provided

$$l_{v \max} < 1 \quad (4.109)$$

where $l_{v \max} \in \Re$ is the maximum eigenvalue of the matrix l_v .

Proof: See proof of Theorem 4.2.1.

4.4.3 BACKLASH COMPENSATION USING DYNAMIC INVERSION

Backstepping technique will be utilized to design the backlash compensation scheme. This is accomplished in two steps. First an ideal control law is designed that renders good performance as if there is no backlash. This control law is given in Theorem 4.4.1. Unfortunately, in the presence of unknown backlash nonlinearity, the desired and actual value of the control signal, $\tau(k)$, will be different.

Under these circumstances, a dynamic inversion technique by NN can be used for the compensation of the inversion error which is derived by using the second step in backstepping. The objective is to make $\tau(k)$ closely follow $\tau_{des}(k)$.

The actuator output given by (4.106) is the desired control signal. The complete error system dynamics can be found by defining the error

$$\tilde{\tau}(k) = \tau_{\text{des}}(k) - \tau(k) \quad (4.110)$$

By substituting the desired control input (4.110) under the presence of unknown backlash the system dynamics (4.107) can be rewritten as

$$r(k+1) = l_v r(k) - \tilde{f}(x(k)) + d(k) - \tilde{\tau}(k) \quad (4.111)$$

Evaluating (4.110) at the subsequent time interval

$$\begin{aligned} \tilde{\tau}(k+1) &= \tau_{\text{des}}(k+1) - \tau(k+1) \\ &= \tau_{\text{des}}(k+1) - B(\tau(k), u(k), u(k+1)) \end{aligned} \quad (4.112)$$

which together with (4.112) represents the complete system error dynamics.

Recall the dynamics of the backlash nonlinearity as

$$\tau(k+1) = \phi(k) \quad (4.113)$$

$$\phi(k) = B(\tau(k), u(k), u(k+1)) \quad (4.114)$$

where $\phi(k)$ is a pseudocontrol input (Leitner et al. 1997). If the backlash is known, then one can select

$$u(k+1) = B^{-1}(\tau(k), u(k), \phi(k)) \quad (4.115)$$

Since the backlash and its inverse are not known beforehand, one can only approximate the backlash inverse as

$$\hat{u}(k+1) = \hat{B}^{-1}(\tau(k), \hat{u}(k), \hat{\phi}(k)) \quad (4.116)$$

The backlash dynamics can now be written as

$$\begin{aligned} \tau(k+1) &= B(\tau(k), \hat{u}(k), \hat{u}(k+1)) \\ &= \hat{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) + \tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) \\ &= \hat{\phi}(k) + \tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) \end{aligned} \quad (4.117)$$

where $\hat{\phi}(k) = \hat{B}(\tau(k), \hat{u}(k), \hat{u}(k+1))$, and therefore its inverse is given by $\hat{u}(k+1) = \hat{B}^{-1}(\tau(k), \hat{u}(k), \hat{\phi}(k))$. The unknown function

$\tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1))$, which represents the backlash inverse error, will be approximated using the NN.

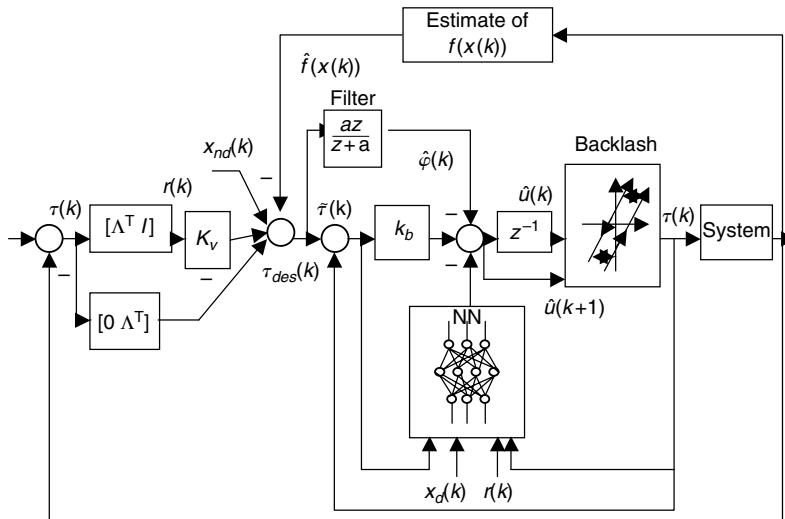
In order to design a stable closed-loop system with backlash compensation, one selects a nominal backlash inverse $\hat{u}(k+1) = \hat{\phi}(k)$ and pseudocontrol input as

$$\hat{\phi}(k) = -l_b \tilde{\tau}(k) + \tau_{\text{filt}}(k) + \hat{W}^T(k) \varphi(V^T x_{\text{NN}}(k)) \quad (4.118)$$

where $l_b > 0$ is a design parameter and τ_{filt} is a discrete-time filtered version of τ_{des} . τ_{filt} is a filtered prediction that approximates $\tau_{\text{des}}(k+1)$ and is obtained using the discrete-time filter $az/(Z+a)$ as shown in Figure 4.21. This is equivalent of using a filtered derivative instead of a pure derivative in continuous-time dynamics inversion, as required in industrial control systems. The filtered dynamics illustrated in Figure 4.21 can be written as

$$\tau_{\text{filt}}(k) = -\frac{\tau_{\text{filt}}(k+1)}{a} + \tau_{\text{des}}(k+1) \quad (4.119)$$

where a is a design parameter. It can be observed that when the filter parameter a is large enough, we have $\tau_{\text{filt}}(k) \approx \tau_{\text{des}}(k+1)$. The mismatch term $(-\tau_{\text{filt}}(k+1)/a)$ can be approximated along with the backlash inversion error using the NN. The complete backlash compensator is given in Table 4.4.



Notes: $\Lambda = [\lambda_{n-1} \lambda_{n-2} \dots \lambda_1]$, $\bar{x}(k) = [x_1(k) \ x_2(k) \ \dots \ x_n(k)]^T$

FIGURE 4.21 NN backlash compensator structure.

TABLE 4.4
NN Backlash Compensator

Define the control input $\tau_{\text{des}}(k) \in \Re^m$ assuming no backlash as

$$\tau_{\text{des}}(k) = x_{nd}(k+1) - \hat{f}(x(k)) + l_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k)$$

where $l_v \in \Re^{m \times m}$ is a diagonal gain matrix. Let the control input be provided by $\hat{u}(k+1) = \hat{\phi}(k)$ where the pseudocontrol input is selected as

$$\hat{\phi}(k) = -l_b \tilde{\tau}(k) + \tau_{\text{filt}}(k) + \hat{W}^T(k) \varphi(V^T x_{\text{NN}}(k))$$

where $\tau_{\text{filt}}(k) = -(\tau_{\text{filt}}(k+1)/a) + \tau_{\text{des}}(k+1)$ and $\tilde{\tau}(k) = \tau_{\text{des}}(k) - \tau(k)$. Let the NN weight tuning be provided by

$$\hat{W}(k+1) = \hat{W}(k) + \alpha \varphi(k) (r(k+1) + \tilde{\tau}(k+1))^T - \Gamma \|I - \alpha \varphi(k) \varphi^T(k)\| \hat{W}(k)$$

where $\alpha > 0$ is a constant learning rate parameter or adaptation gain, $\Gamma > 0$ is another design parameter.

Based on the NN approximation property, the backlash inversion plus the filtered error dynamics can be represented as

$$\tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) + \frac{\tau_{\text{filt}}(k+1)}{a} = W^T(k) \varphi(V^T x_{\text{NN}}(k)) + \varepsilon(k) \quad (4.120)$$

where the NN input vector is chosen to be $x_{\text{NN}}(k) = [1 \ r^T(k) \ x_d^T(k) \ \tilde{\tau}^T(k) \ \tau^T(k)]^T$ and $\varepsilon(k)$ represents the NN approximation error. It is important to note that the input-to-hidden-layer weights are selected at random initially to provide a basis (Igelnik and Pao 1995) and held constant throughout the tuning process.

Define the NN weight estimation error as

$$\tilde{W}(k) = \hat{W}(k) - W \quad (4.121)$$

where $\hat{W}(k)$ is the estimate of the target weights $W(k)$. Using the proposed controller shown in Figure 4.21, the error dynamics can be written as

$$\begin{aligned} \tilde{\tau}(k+1) &= \tau_{\text{des}}(k+1) - \hat{\phi}(k) + \tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) \\ &= -l_b \tilde{\tau}(k) + \frac{\tau_{\text{filt}}(k+1)}{a} - \hat{W}^T(k) \varphi(V^T x_{\text{NN}}(k)) \\ &\quad + \tilde{B}(\tau(k), \hat{u}(k), \hat{u}(k+1)) \\ &= -l_b \tilde{\tau}(k) - \hat{W}^T(k) \varphi(V^T x_{\text{NN}}(k)) + W^T(k) \varphi(V^T x_{\text{NN}}(k)) + \varepsilon(k) \end{aligned} \quad (4.122)$$

Using (4.121),

$$\tilde{\tau}(k+1) = l_b \tilde{\tau}(k) + \tilde{W}^T(k)\varphi(k) + \varepsilon(k) \quad (4.123)$$

where $\varphi(k) = \varphi(V^T x_{\text{NN}}(k))$.

The next theorem presents how to tune the NN weights so that the tracking error, $r(k)$, and the backlash estimation error, $\tilde{\tau}(k)$, achieve small values, while the NN estimation errors $\tilde{W}(k)$ are bounded.

Theorem 4.4.2 (Control Law for Backstepping Loop): Consider the system given by (4.102). Let the Assumptions 4.4.1 and 4.4.2 hold with the disturbance bound d_m a known constant. Let the control action $\hat{\phi}(k)$ be provided by (4.118), with $l_b > 0$ being a design parameter. Let the control input be provided by $\hat{u}(k+1) = \hat{\phi}(k)$ and the NN weights are tuned by

$$\begin{aligned} \hat{W}(k+1) = & \hat{W}(k) + \alpha \varphi(k)(r(k+1) + \tilde{\tau}(k+1))^T \\ & - \Gamma \|I - \alpha \varphi(k) \varphi^T(k)\| \hat{W}(k) \end{aligned} \quad (4.124)$$

where $\alpha > 0$ is a constant learning rate parameter or adaptation gain, $\Gamma > 0$ is another design parameter. Then the filtered tracking error $r(k)$, the backlash estimation error $\tilde{\tau}(k)$, and the NN weight estimation error $\hat{W}(k)$ are UUB.

Proof: See Lewis et al. (2002).

Remarks:

1. It is important to note that in this theorem there are no CE and LIP assumptions for the NN controller, in contrast to standard work in discrete-time adaptive control (Åström and Wittenmark 1989; Kanellakopolous 1994). In the proof, the Lyapunov function weights the filtered tracking error, the NN estimation errors for the controller $\tilde{W}(k)$ and the backlash error $\tilde{\tau}(k)$. The Lyapunov function used to show the boundedness of all closed-loop signals is of the form

$$\begin{aligned} J(k) = & (r(k) + \tilde{\tau}(k))^T(r(k) + \tilde{\tau}(k)) + r^T(k)r(k) \\ & + \frac{1}{\alpha} \text{tr}\{\tilde{W}^T(k)\tilde{W}(k)\} > 0 \end{aligned} \quad (4.125)$$

which weights the tracking error $r(k)$, the backlash estimation error $\tilde{\tau}(k)$, and the NN weight estimation error $\tilde{W}(k)$. The proof is exceedingly complex due to the presence of several different variables.

However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner. Additional complexities that arise in the proof due to the fact that the backlash compensator NN system is in the feedforward loop and not in the feedback loop are taken care of.

2. The NN weight-updating rule (4.124) is quite similar to Jagannathan and Lewis (1996) since it includes an extra term, referred to as discrete-time ε -modification (Jagannathan and Lewis 1996; Jagannathan 2001), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms. This is referred to as the “forgetting term” in NN weight-tuning algorithms and it is added to prevent over-training of weights. The Lyapunov proof demonstrates that the additional term in the weight tuning is required. In addition, a quadratic term of the backlash estimation error is also added in the weight tuning.
3. In the above theorem, the dynamics are approximated by a known term and the error in approximation is considered bounded but its bound is assumed to be unknown. One can add another NN to approximate the nonlinear system dynamics similar to the case of the deadzone compensator with actuator constraints described in the previous section. Additionally, one can use reinforcement-based learning instead of the filtered tracking error and backlash estimation at the subsequent time instant to tune the NN weights.

Simulation Example 4.4.1 (Adaptive NN Controller for Backlash Compensation): Consider the following nonlinear system described in (3.151) with the uncertain nonlinear system dynamics given by $f(x(k)) = -5/8[(x_1(k))/(1 + x_2^2(k))] + x_2(k)$. The process input $\tau(k)$ is related to the control input $u(k)$ through the backlash nonlinearity. The deadband widths of the unknown backlash nonlinearity are selected as $d_+(k) = 0.4(1 + 0.1 \sin(kT))$, $d_-(k) = -0.3(1 + 0.1 \cos(kT))$, and the slope as $m = 0.5$ with the sampling interval, T , given as 50 msec. A reference signal used was selected as

$$x_d(k) = \sin(\omega kT + \zeta) \quad \omega = 0.1, \quad \zeta = \frac{\pi}{2} \quad (4.126)$$

The controller gain, l_b , is taken as -0.9 .

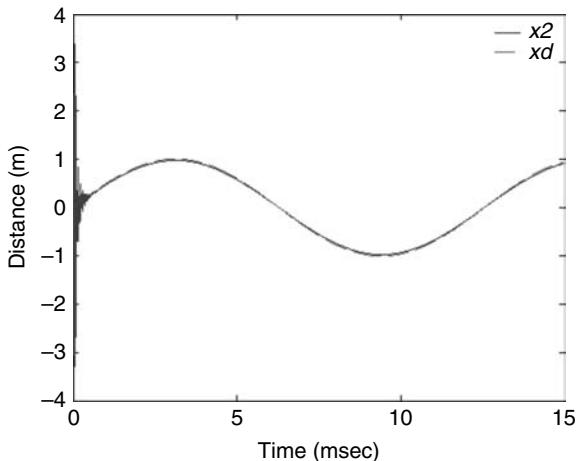


FIGURE 4.22 PD controller without backlash.

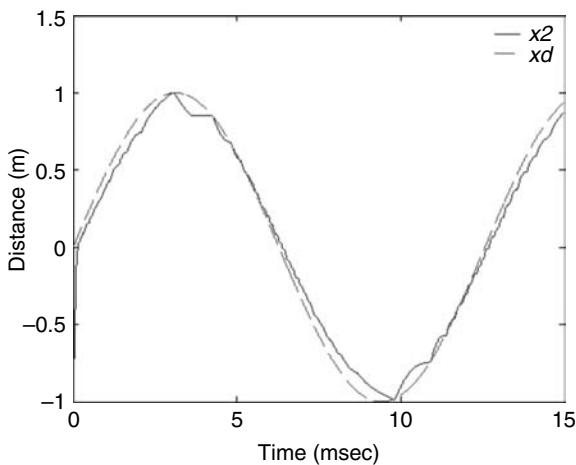


FIGURE 4.23 PD controller with backlash compensator.

The MATLAB[®] code is given in Appendix D. Figure 4.22 shows the system response without the backlash using the standard PD controller. The PD controller does a good job on the tracking, which is achieved at about 2 sec even though considerable overshoots and undershoots are observed during initial stages. Figure 4.23 illustrates the tracking response using only a PD controller

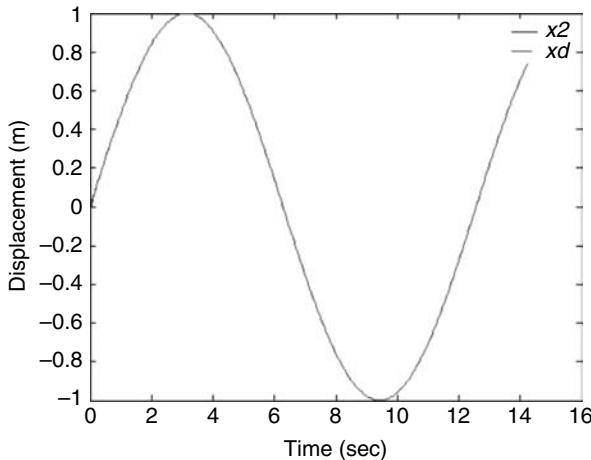


FIGURE 4.24 Proposed NN backlash compensator.

with input backlash. The system backlash destroys the tracking performance, and the controller is unable to compensate for it.

Now we have added the NN backlash compensator prescribed in Table 4.4. Figure 4.24 depicts the results using the discrete-time NN backlash compensator. The backlash compensator takes care of the system backlash and tracking is achieved in less than 0.5 sec. Thus the NN compensator significantly improves the performance of the system in the presence of backlash.

4.5 CONCLUSIONS

This chapter proposes an adaptive critic NN-based controller for a class of nonlinear systems in the presence of magnitude constraint due to the actuator, which is represented and incorporated in the closed-loop system as saturation nonlinearity. This adaptive NN-based approach does not require the information about the system dynamics. Both tracking error and reinforcement learning-based NN methodology is presented. The adaptive critic NN controller includes an action NN for compensating the unknown dynamics, a critic signal for approximating the strategic utility function, and an outer PD control loop for tracking. The tuning of the action-generating NN is performed online without an explicit off-line learning phase. The strategic utility function resembles the Bellman equation.

The input magnitude constraint is modeled as saturation nonlinearity and it was treated by converting the nonlinearity into an input disturbance, which was suitably accommodated by the adaptive NN weight-tuning law. The proposed adaptive critic NN controller was applied to a nonlinear system with and without saturation and the controller performance was demonstrated. Results demonstrate that the Lyapunov-based adaptive critic design renders a satisfactory performance while ensuring closed-loop stability.

Subsequently, in the next section, an RLNN was developed for a class of uncertain discrete-time nonlinear systems with unknown actuator deadzones and magnitude constraints. The magnitude constraints are manifested in the controller design as saturation nonlinearities. The proposed RLNN controller consisting of two action-generating NN and a third critic NN renders a satisfactory tracking performance. Lyapunov analysis ensures the boundedness of all the closed-loop signals in the presence of multiple nonlinearities.

Finally, in the last section, the NN backlash compensator is presented by using a well-known backstepping technique. The general case of nonsymmetric time varying backlash is treated. A rigorous design procedure is given that results in a PD tracking loop with an adaptive NN in the feedforward loop for dynamic inversion of the backlash nonlinearity. The NN feedforward compensator is adapted in such a way as to estimate online the backlash inverse. Simulation results concur with the theoretical results.

REFERENCES

- Armstrong-Helouvry, B., Dupont, P., and Canudas De Wit, C., A survey of models, analysis tools and compensation methods for the control of machines with friction, *Automatica*, 30, 1391–1413, 1994.
- Åström, K.J. and Wittenmark, B., *Adaptive Control*, Addison-Wesley, Reading, MA, 1989.
- Barto, A.G., Reinforcement learning and adaptive critic methods, *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 65–90, 1992.
- Bertsekas, D.P. and Tsitsiklis, J.N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- Byrnes, C.I. and Lin, W., Losslessness, feedback equivalence, and the global stabilization of discrete-time nonlinear systems, *IEEE Trans. Autom. Contr.*, 39, 83–98, 1994.
- Calise, A.J., Neural networks in nonlinear aircraft flight control, *IEEE Aerospace Electron. Syst. Mag.*, 11, 5–10, 1996.
- Campos, J. and Lewis, F.L., Deadzone compensation in discrete-time using adaptive fuzzy logic, *IEEE Trans. Fuzzy Syst.*, 7, 697–707, 1999.
- Canudas de Wit, C., Olsson, H., Åström, K.J., and Lischinsky, P., A new model for control of systems with friction, *IEEE Trans. Autom. Contr.*, 40, 419–425, 1995.

- Desoer, C. and Shahruz, S.M., Stability of dithered nonlinear systems with backlash or hysteresis, *Int. J. Contr.*, 43, 1045–1060, 1986.
- Grundelius, M. and Angelli, D., Adaptive control of systems with backlash acting on the input, *Proc. IEEE Conf. Decis. Contr.*, 4, 4689–4694, 1996.
- He, P. and Jagannathan, S., Adaptive critic neural network-based controller for nonlinear systems with input constraints, *Proc. IEEE Conf. Decis. Contr.*, 6, 5709–5714, 2003.
- He, P., Jagannathan, S., and Balakrishnan, S., Adaptive critic-based neural network controller for nonlinear systems with unknown deadzones, *Proc. IEEE Conf. Decis. Contr.*, 1, 955–960, 2002.
- Igelnik, B. and Pao, Y.H., Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, 6, 1320–1329, 1995.
- Jagannathan, S., Robust backstepping control of nonlinear systems using multilayered neural networks, *Proc. IEEE Conf. Decis. Contr.*, 1, 480–485, 1997.
- Jagannathan, S., Robust backstepping control of robotic systems using neural networks, *Proc. IEEE Conf. Decis. Contr.*, 943–948, 1998.
- Jagannathan, S., Control of a class of nonlinear discrete-time systems using multi layer neural networks, *IEEE Trans. Neural Netw.*, 12, 1113–1120, 2001.
- Jagannathan, S. and Galan, G., Adaptive critic neural network-based object grasping controller using a three-fingered gripper, *IEEE Trans. Neural Netw.*, 15, 395–407, 2004.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Autom. Contr.*, 41, 1693–1699, 1996.
- Kanellakopoulos, I., A discrete-time adaptive nonlinear system, *IEEE Trans. Automat. Contr.*, 39, 2362–2365, 1994.
- Karason, S.P. and Annaswamy, A.M., Adaptive control in the presence of input constraints, *IEEE Trans. Automat. Contr.*, 39, 2325–2330, 1994.
- Leitner, J., Calise, A., and Prasad, J.V.R., Analysis of adaptive neural networks for helicopter flight control, *J. Guidance Contr. Dynam.*, 20, 972–979, 1997.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.
- Lewis, F.L., Jagannathan, S., and Yesilderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, UK, 1999.
- Lewis, F.L., Campos J., and Selmic, R., *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society for Industrial and Applied Mathematics, Philadelphia, 2002.
- Lin, X. and Balakrishnan, S.N., Convergence analysis of adaptive critic based optimal control, *Proc. Am. Contr. Conf.*, 3, 1929–1933, 2000.
- Luenberger, D.G., *Introduction to Dynamic Systems*, John Wiley & Sons, New York, 1979.
- Miller, W.T. III, Sutton, R.S., and Werbos, P.J., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1991.

- Murray, J.J., Cox, C., Lendaris, G.G., and Saeks, R., Adaptive dynamic programming, *IEEE Trans. Syst., Man, Cybern.*, 32, 140–153, 2002.
- Narendra, K.S. and Parthasarathy, K.S., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, 4–27, 1990.
- Prokhorov, D.V. and Feldkamp, L.A., Analyzing for Lyapunov stability with adaptive critics, *Proc. of the IEEE Conf. on Systems, Man and Cybernetics*, San Diego, CA, pp. 1658–1661, 1998.
- Prokhorov, D.V. and Wunsch, D.C., Adaptive critic designs, *IEEE Trans. Neural Netw.*, 8, 997–1007, 1997.
- Recker, P., Kokotovic, V., Rhode, D., and Winkelman, J., Adaptive nonlinear control of systems containing a deadzone, *Proc. IEEE Conf. Decis. Contr.*, 2111–2115, 1991.
- Selmic, R.R. and Lewis, F.L., Deadzone compensation in motion control systems using neural networks, *IEEE Trans. Autom. Contr.*, 45, 602–613, 2000.
- Shervais, S., Shannon, T.T., and Lendaris, G.G., Intelligent supply chain management using adaptive critic learning, *IEEE Trans. Syst., Man, Cybern.*, 33, 235–244, 2003.
- Si, J., *NSF Workshop on Learning and Approximate Dynamic Programming*, Playacar, Mexico, 2002. Available: <http://ebrains.la.asu.edu/~nsfadp/>
- Si, J. and Wang, Y.T., On-line learning control by association and reinforcement, *IEEE Trans. Neural Netw.*, 12, 264–276, 2001.
- Tao, G. and Kokotovic, P.V., Discrete-time adaptive control of systems with unknown deadzones, *Int. J. Contr.*, 61, 1–17, 1995.
- Tao, G. and Kokotovic, P.V., *Adaptive Control of Systems with Actuator and Sensor Nonlinearities*, John Wiley & Sons, New York, 1996.
- Tian, M. and Tao, G., Adaptive control of a class of nonlinear systems with unknown deadzones, *Proceedings of the IFAC World Congress*, San Francisco, pp. 209–214, 1996.
- Utkin, V.I., *Sliding Modes and Their Applications in Variable Structure Systems*, Mir Publishers, Moscow, pp. 55–63, 1978.
- von Bertalanffy, L., *General Systems Theory*, Braziller, New York, 1968.
- Werbos, P.J., A menu of designs for reinforcement learning over time, In *Neural Networks for Control*, Miller, W.T., Sutton, R.S., and Werbos, P.J., Eds., MIT Press, Cambridge, pp. 67–95, 1991.
- Werbos, P.J., Neurocontrol and supervised learning: an overview and evaluation, In *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 65–90, 1992.
- White, D.A. and Sofge, D.A., Eds., *Handbook of Intelligent Control*, Van Nostrand Reinhold, New York, 1992.
- Whitehead, A.N., *Science and the Modern World*, Lowell Lectures (1925), Macmillan, New York, 1953.
- Yeh, P.C. and Kokotovic, P.V., Adaptive output feedback design for a class of nonlinear discrete-time systems, *IEEE Trans. Autom. Contr.*, 40, 1663–1668, 1995.

PROBLEMS

SECTION 4.2

4.2-1: A nonlinear system is described by

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \quad (4.127)$$

where $f(x(k)) = -3/16[(x_1(k))/(1+x_2^2(k))] + 0.3x_2(k)$. The objective is to make the state $x_2(k)$ track a reference signal using the proposed adaptive critic NN controller with input saturation. The reference signal used was selected as

$$x_{2d} = \begin{cases} \sin(\omega kT + \xi), & \omega = 0.1, \quad \xi = \frac{\pi}{2}, \quad 0 \leq k \leq 3000 \\ -1, & 3000 < k \leq 4000 \text{ or } 5000 < k \leq 6000 \\ 1, & 4000 < k \leq 5000 \end{cases} \quad (4.128)$$

where the desired signal is in part a sine wave and in part a unit step signal. The two different reference signals are used to evaluate the learning ability of the adaptive critic NN controller. Take a sampling interval T to be 50 msec and add white Gaussian noise with the standard deviation of 0.005. The time duration is taken to be 300 sec. Consider the proposed disturbance acting on the system as

$$d(k) = \begin{cases} 0, & k < 2000 \\ 1.5, & 6000 \geq k \geq 2000 \end{cases} \quad (4.129)$$

Design the adaptive NN controller by taking the actuator limits at 2.0 and 3.0 and by appropriately selecting the number of neurons in the hidden layer.

4.2-2: The nonlinear system is described by

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ x_2(k+1) &= f(x(k)) + u(k) + d(k) \end{aligned} \quad (4.130)$$

where $f(x(k)) = -3/16[x_1(k)x_2(k)/(1+x_2^2(k))] + 0.3x_1(k)$. The objective is to make the state $x_2(k)$ track a reference signal using the proposed adaptive critic NN controller with input saturation. The reference signal used was selected as

$$x_{2d} = \sin(\omega kT + \xi) \quad \omega = 0.1 \quad \xi = \frac{\pi}{2} \quad 0 \leq k \leq 6000 \quad (4.131)$$

where the desired signal is a sine wave. Take a sampling interval T to be 10 msec and add white Gaussian noise with the standard deviation of 0.005. The time duration is taken to be 300 sec. Design the adaptive NN controller by taking the actuator limits at 2.0 and 3.0 and by appropriately selecting the number of neurons in the hidden layer.

SECTION 4.3

4.3-1: Consider the following nonlinear system:

$$x_1(k+1) = x_2(k) \quad x_2(k+1) = f(x(k)) + u(k) \quad (4.132)$$

where the uncertain nonlinear system dynamic is $f(x(k)) = -3/16[x_2(k)/(1+x_2^2(k))] + x_1(k)$. The unknown deadzone widths are selected as $b_+(k) = 0.4(1+0.1 \sin(kT))$, $b_-(k) = 0.3(1+0.1 \cos(kT))$ with the sampling interval, T , given as 50 msec. A reference signal used was selected as

$$x_d(k) = \sin(\omega kT + \zeta) \quad \omega = 0.1 \quad \zeta = \pi/2 \quad (4.133)$$

The controller gain, l , is taken as -0.9 . The upper bound on the control input is chosen as $u_{\max} = 0.9$. Design the NN controller.

SECTION 4.4

4.4-1: Design a reinforcement learning-based NN controller for the nonlinear system presented in (4.102) with backlash nonlinearity.

4.4-2: Prove the UUB of the closed-loop system by using the NN weight-update law presented in (4.124).

4.4-3: Consider the following nonlinear system

$$x_1(k+1) = x_2(k), \quad x_2(k+1) = f(x(k)) + u(k) \quad (4.134)$$

where the uncertain nonlinear system dynamic is $f(x(k)) = -3/16[x_2(k)/(1+x_2^2(k))] + x_1(k)$. The process input $\tau(k)$ is related to the control input $u(k)$ through the backlash nonlinearity. The deadband widths of the unknown backlash nonlinearity are selected as $d_+(k) = 0.4(1+0.1 \sin(kT))$, $d_-(k) = 0.3(1+0.1 \cos(kT))$, and the slope as $m = 0.5$ with the sampling interval, T , given as 50 msec. The sampling interval, T , is given as 50 msec. A reference

signal is selected as

$$x_d(k) = \sin(\omega kT + \zeta) \quad \omega = 0.1 \quad \zeta = \pi/2 \quad (4.135)$$

Design the NN controller.

APPENDIX 4.A

Proof of Theorem 4.2.2: Define the Lyapunov function candidate

$$\begin{aligned} J(k) &= \frac{1}{\gamma_1} r^T(k)r(k) + \frac{1}{\alpha_1} \text{tr}(\tilde{w}_1^T(k)\tilde{w}_1(k)) + \frac{1}{\gamma_2} \|\zeta_1(k-1)\|^2 \\ &\quad + \frac{1}{\gamma_3 \alpha_2} \text{tr}(\tilde{w}_2^T(k)\tilde{w}_2(k)) \end{aligned} \quad (4.A.1)$$

where $\zeta_1(k-1) = (\hat{w}_1(k-1) - w_1)^T \phi_1(k-1) = \tilde{w}_1^T(k-1) \phi_1(k-1)$ and $0 < \gamma_i, i = 1, 2, 3$.

The first difference of the Lyapunov function is calculated as

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) \quad (4.A.2)$$

The $\Delta J_1(k)$ is obtained using the filtered tracking error dynamics (3.321) as

$$\begin{aligned} \Delta J_1(k) &= \frac{1}{\gamma_1} (r^T(k+1)r(k+1) - r^T(k)r(k)) \\ &= \frac{1}{\gamma_1} ((l_v r(k) - \zeta_2(k) + \varepsilon_2(x(k)) + d(k))^T (l_v r(k) - \zeta_2(k) \\ &\quad + \varepsilon_2(x(k)) + d(k)) - r^T(k)r(k)) \\ &\leq \frac{3}{\gamma_1} \left(\left(l_{v \max}^2 - \frac{1}{3} \right) \|r(k)\|^2 + \|\zeta_2(k)\|^2 + \|\varepsilon_2(k) + d(k)\|^2 \right) \end{aligned} \quad (4.A.3)$$

where the $l_{v \max} \in R$ is the maximum eigenvalue of the matrix $l_v \in R^{m \times m}$. Now taking the second term in the first difference of (4.A.2) and rewriting as

$$\Delta J_2(k) = \frac{1}{\alpha_1} \text{tr}[\tilde{w}_1^T(k+1)\tilde{w}_1(k+1) - \tilde{w}_1^T(k)\tilde{w}_1(k)] \quad (4.A.4)$$

Substituting the NN weight updates from (4.29) yields

$$\begin{aligned}\tilde{w}_1(k+1) = & (I - \alpha_1 \phi_1(k) \phi_1^T(k)) \tilde{w}_1(k) - \alpha_1 \phi_1(k) (w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) \\ & - \alpha \hat{w}_1^T(k-1) \phi_1(k-1))^T\end{aligned}\quad (4.A.5)$$

Now substituting (4.A.5) into (4.A.4) and combining them to get

$$\begin{aligned}\Delta J_2(k) \leq & -(1 - \alpha_1 \phi_1^T(k) \phi_1(k)) \\ & \times \|\zeta_1(k) + w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1) \phi_1(k-1)\|^2 \\ & - \|\zeta_1(k)\|^2 + 2 \|w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha w_1^T \phi_1(k-1)\|^2 \\ & + 2\alpha^2 \|\zeta_1(k-1)\|^2\end{aligned}\quad (4.A.6)$$

Now taking the third term in (4.A.2) to get

$$\Delta J_3(k) = \frac{1}{\gamma_2} (\|\zeta_1(k)\|^2 - \|\zeta_1(k-1)\|^2) \quad (4.A.7)$$

The fourth term in (4.A.2) is expanded as

$$\Delta J_3(k) = \frac{1}{\gamma_3 \alpha_2} \text{tr}[\tilde{w}_2^T(k+1) \tilde{w}_2(k+1) - \tilde{w}_2^T(k) \tilde{w}_2(k)] \quad (4.A.8)$$

Substituting the weight updates for the NN (3.314) and simplifying to get

$$\begin{aligned}\Delta J_4(k) \leq & \frac{1}{\gamma_3} \left\{ -(1 - \alpha_2 \phi_2^T(k) \phi_2(k)) \|\zeta_2(k) + \hat{w}_1^T(k) \phi_1(k) \right. \\ & \left. - (\varepsilon_2(x(k)) + d(k))\|^2 - \|\zeta_2(k)\|^2 \right\} + \frac{2}{\gamma_3} \left\{ \|w_1^T(k) \phi_1(k) \right. \\ & \left. - (\varepsilon_2(x(k)) + d(k))\|^2 + \|\zeta_1(k)\|^2 \right\}\end{aligned}\quad (4.A.9)$$

Combining (4.A.3), (4.A.6), (4.A.8), and (4.A.9) to get the first difference of the Lyapunov Equation 4.A.2

$$\begin{aligned}
\Delta J(k) \leq & \frac{-1}{\gamma_1} (1 - 3l_{v \max}^2) \|r(k)\|^2 - \left(1 - \frac{1}{\gamma_2} - \frac{2}{\gamma_3}\right) \|\zeta_1(k)\|^2 - \left(\frac{1}{\gamma_3} - \frac{3}{\gamma_1}\right) \\
& \times \|\zeta_2(k)\|^2 - \left(\frac{1}{\gamma_2} - 2\alpha^2\right) \|\zeta_1(k-1)\|^2 - (1 - \alpha_1 \phi_1^T(k) \phi_1(k)) \\
& \times \|\zeta_1(k) + w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1) \phi_1(k-1)\|^2 \\
& - \frac{1}{\gamma_3} \{(1 - \alpha_2 \phi_2^T(k) \phi_2(k)) \|\zeta_2(k) + \hat{w}_1^T(k) \phi_1(k) - (\varepsilon_2(x(k)) \\
& + d(k))\|^2\} + 2 \|w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha w_1^T \phi_1(k-1)\|^2 \\
& + \frac{2}{\gamma_3} \{\|w_1^T(k) \phi_1(k) - (\varepsilon_2(x(k)) + d(k))\|^2\} + \frac{3}{\gamma_1} \|\varepsilon_2(k) + d(k)\|^2
\end{aligned} \tag{4.A.10}$$

Choose

$$\begin{aligned}
\gamma_1 &> 3\gamma_3 \\
\gamma_2 &= \frac{\sqrt{2}}{2}\alpha \\
\gamma_3 &> \frac{2}{1 - 2\alpha^2}
\end{aligned} \tag{4.A.11}$$

and define

$$\begin{aligned}
D^2 = & 2 \|w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha w_1^T \phi_1(k-1)\|^2 \\
& + \frac{2}{\gamma_3} \{\|w_1^T(k) \phi_1(k) - (\varepsilon_2(x(k)) + d(k))\|^2\} + \frac{3}{\gamma_1} \|\varepsilon_2(k) + d(k)\|^2
\end{aligned} \tag{4.A.12}$$

The upper bound, D_m , for D is

$$D^2 \leq D_m^2 = 6 \left(1 + \alpha^2 + \frac{1}{\gamma_3}\right) w_{1m}^2 \phi_{1m}^2 + 6 \left(\frac{1}{\gamma_1} + \frac{1}{\gamma_3}\right) (\varepsilon_{2m}^2 + d_m^2) \tag{4.A.13}$$

Using (4.A.11) and (4.A.12) in (4.A.10) and rewriting

$$\begin{aligned}
 \Delta J(k) \leq & \frac{-1}{\gamma_1} (1 - 3l_{v\max}^2) \|r(k)\|^2 - \left(1 - \frac{1}{\gamma_2} - \frac{2}{\gamma_3}\right) \|\zeta_1(k)\|^2 \\
 & - \left(\frac{1}{\gamma_3} - \frac{3}{\gamma_1}\right) \|\zeta_2(k)\|^2 - (1 - \alpha_1 \phi_1^T(k) \phi_1(k)) \\
 & \times \|\zeta_1(k) + w_1^T(k) \phi_1(k) + \alpha^{N+1} p(k) - \alpha \hat{w}_1^T(k-1) \phi_1(k-1)\|^2 \\
 & - \frac{1}{\gamma_3} \{(1 - \alpha_2 \phi_2^T(k) \phi_2(k)) \|\zeta_2(k) + \hat{w}_1^T(k) \phi_1(k) \\
 & - (\varepsilon_2(x(k)) + d(k))\|^2\} + D^2
 \end{aligned} \tag{4.A.14}$$

This further implies that the first difference $\Delta J(k) \leq 0$ as long as (4.46) through (4.49) hold and

$$\|r(k)\| > \sqrt{\frac{\gamma_1}{1 - 3l_{v\max}^2}} D_m \tag{4.A.15}$$

or

$$\|\zeta_1(k)\| > \frac{D_m}{\sqrt{1 - 1/\gamma_2 - 2/\gamma_3}} \tag{4.A.16}$$

or

$$\|\zeta_2(k)\| > \frac{D_m}{\sqrt{1/\gamma_3 - 3/\gamma_1}} \tag{4.A.17}$$

According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the filtered tracking error and the error in weight estimates are UUB. The boundedness of $\|\zeta_1(k)\|$ and $\|\zeta_2(k)\|$ implies that $\|\tilde{w}_1(k)\|$ and $\|\tilde{w}_2(k)\|$ are bounded, and this further implies that the weight estimates $\hat{w}_1(k)$ and $\hat{w}_2(k)$ are bounded.

Note: Condition (4.A.11) is easy to check. For instance, select the parameters $\alpha = \frac{1}{2}$, $\gamma_1 = 16$, $\gamma_2 = \frac{\sqrt{2}}{4}$, $\gamma_3 = 5$ to satisfy (4.A.11).

APPENDIX 4.B

Proof of Theorem 4.2.3: Define the Lyapunov function candidate as

$$\begin{aligned} J(k) = & \frac{1}{\gamma_1} e_u^T(k) e_u(k) + \frac{1}{\alpha_1} \text{tr}(\tilde{w}_1^T(k) \tilde{w}_1(k)) + \frac{1}{\gamma_2} \|\xi_1(k)\|^2 \\ & + \frac{1}{\gamma_3 \alpha_3} \text{tr}(\tilde{w}_3^T(k) \tilde{w}_3(k)) \end{aligned} \quad (4.B.1)$$

The proof follows in similar steps as that of Theorem 4.2.2, so it is omitted. The first difference $\Delta J(k) \leq 0$ as long as (4.46) through (4.49), (4.62), and (4.A.11) is satisfied and

$$\|e_u(k)\| > \sqrt{\frac{\gamma_1}{1 - 3l_{v \max}^2}} D_m \quad (4.B.2)$$

or

$$\|\xi_1(k)\| > \frac{D_m}{\sqrt{1 - 1/\gamma_2 - 2/\gamma_3}} \quad (4.B.3)$$

or

$$\|\xi_3(k)\| > \frac{D_m}{\sqrt{1/\gamma_3 - 3/\gamma_1}} \quad (4.B.4)$$

where

$$\xi_3(k) = (\hat{w}_3(k) - w_3)^T \phi_3(k) = \tilde{w}_3^T(k) \phi_3(k) \quad (4.B.5)$$

According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the auxiliary error and the error in weight estimates are UUB. The boundedness of $\|\xi_1(k)\|$ and $\|\xi_3(k)\|$ implies that $\|\tilde{w}_1(k)\|$ and $\|\tilde{w}_3(k)\|$ are bounded, and this further implies that the weight estimates $\hat{w}_1(k)$ and $\hat{w}_3(k)$ are bounded.

The next step is to show the filtered tracking error, $r(k)$, is bounded. Here, two cases are being discussed. The first is when $\|v(k)\| \leq u_{\max}$, and the second is when $\|v(k)\| > u_{\max}$.

Case I: $\|v(k)\| \leq u_{\max}$

If $\|v(k)\| \leq u_{\max}$, then $u(k) = v(k)$. Recall the closed-loop error system from (4.45) as

$$r(k+1) = l_v r(k) - \zeta_2(k) + \varepsilon_2(x(k)) + d(k) \quad (4.B.6)$$

This is a linear system driven by function estimation error and disturbances. Since the disturbances are bounded and the weight estimation error is shown to be bounded above, the stable filtered tracking error system is driven by bounded inputs. Therefore the filtered tracking error is bounded and hence all the tracking errors are bounded.

Case II: $\|v(k)\| > u_{\max}$

If $\|v(k)\| > u_{\max}$, then $u(k) = u_{\max} \text{sgn}(v(k))$. For the nonlinear system (4.10), the tracking error should be in the form of:

$$\begin{aligned} e_n(k+1) &= x_n(k+1) - x_{nd}(k+1) \\ &= f(x(k)) + u_{\max} \text{sgn}(v(k)) + d(k) - x_{nd}(k+1) \end{aligned} \quad (4.B.7)$$

Over a compact set, the smooth function is bounded by F_{\max} and the desired trajectory is bounded by $x_{d\max}$. Then, we obtain the upper bound of $e_n(k)$:

$$\|e_n(k)\| \leq F_{\max} + u_{\max} + d_M + x_{d\max} \quad (4.B.8)$$

Based on the definition of filtered tracking error of (4.12) and $e_n(k)$ having an upper bound, in this case, the filtered tracking error is UUB. Considering Cases I and II, the proof of the UUB of filtered tracking error is complete.

APPENDIX 4.C

Proof of Theorem 4.3.1: The following fact will be used in the proof

$$h(h^{-1}(p(k))) = p(k) \quad (4.C.1)$$

where $h^{-1}(p(k))$ is the inverse function of $h(k)$.

From (4.74) and (4.C.1), $p(k)$ is rewritten as

$$p(k) = w_4^T \phi_4(h^{-1}(p(k))) + \varepsilon_4(h^{-1}(p(k))) \quad (4.C.2)$$

Equation 4.C.2 can be expressed as

$$p(k) = w_4^T \phi_4(h^{-1}(p(k)) - p(k) + p(k)) + \varepsilon_4(h^{-1}(p(k))) \quad (4.C.3)$$

Combining (4.71) with (4.C.3) to get

$$p(k) = w_4^T \phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) + \varepsilon_4(h^{-1}(p(k))) \quad (4.C.4)$$

Equation 4.C.4 is a critical equation to be used in the following proof. From (4.69), we have

$$q(k) = h(\tau(k)) \quad (4.C.5)$$

Using an action-generating NN to approximate the deadzone function (see Equation 4.69)

$$q(k) = w_4^T \phi_4(\tau(k)) + \varepsilon_4(\tau(k)) \quad (4.C.6)$$

From Figure 4.19,

$$\tau(k) = p(k) + \hat{w}_1^T \phi_1(p(k)) \quad (4.C.7)$$

Based on Definition 4.3.2, (4.C.7) can be further written as

$$\begin{aligned} \tau(k) &= p(k) + w_1^T \phi_1(p(k)) + \tilde{w}_1^T \phi_1(p(k)) \\ &= p(k) + w_1^T \phi_1(p(k)) + \xi_1(k) \end{aligned} \quad (4.C.8)$$

Combining (4.C.6) and (4.C.8) to get

$$\begin{aligned} q(k) &= w_4^T \phi_4(p(k)) + w_1^T \phi_1(p(k)) + \xi_1(k) + \varepsilon_4(\tau(k)) \\ &= w_4^T \phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k) + \xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \end{aligned} \quad (4.C.9)$$

Using the Taylor series expansion we have

$$\begin{aligned} q(k) &= w_4^T \phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) + w_4^T \phi'_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) \\ &\quad + p(k))(\xi_1(k) - \varepsilon_1(p(k))) + O(\xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \end{aligned} \quad (4.C.10)$$

where $O(\xi_1(k) - \varepsilon_1(p(k)))$ is the Lagrange remainder after two terms, and $w_4^T \phi'_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k))$ is defined as

$$\begin{aligned} &w_4^T \phi'_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) \\ &= \frac{w_4^T d(\phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)))}{d(p(k))} \end{aligned} \quad (4.C.11)$$

By applying Assumptions 4.3.2 and 4.3.3, we define

$$g(k) = w_4^T \phi'_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) \quad (4.C.12)$$

where $g(k)$ is bounded over the compact set, S , whose bound is given by

$$\|g(k)\| \leq g_m \quad (4.C.13)$$

Simplifying (4.C.10) using (4.C.4) and (4.C.12), we obtain

$$\begin{aligned} q(k) &= w_4^T \phi_4(w_1^T \phi_1(p(k)) + \varepsilon_1(p(k)) + p(k)) + \varepsilon_4(h^{-1}(p(k))) \\ &\quad + g(k)(\xi_1(k) - \varepsilon_1(p(k))) - \varepsilon_4(h^{-1}(p(k))) + O(\xi_1(k) \\ &\quad - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \end{aligned} \quad (4.C.14)$$

that is,

$$\begin{aligned} q(k) &= p(k) + g(k)\xi_1(k) - g(k)\varepsilon_1(p(k)) + O(\xi_1(k) - \varepsilon_1(p(k))) + \varepsilon_4(\tau(k)) \\ &\quad - \varepsilon_4(h^{-1}(p(k))) \end{aligned} \quad (4.C.15)$$

From (4.C.15), it can be concluded that when the NN weight estimation and the NN reconstruction errors go to zero, the throughput error $q(k) - p(k)$ approaches zero. This makes the deadzone precompensator plus the deadzone equal to unity. It also implies that the effect of deadzone is overcome by the proposed NN precompensator.

Remarks: In Selmic and Lewis (2000), the Lagrange remainder after three terms is completely ignored to prove that the throughput error is bounded. This may not be a reasonable approach due to (1) the Lagrange remainder after three terms does exist and (2) the Lagrange remainder becomes infinity in Selmic and Lewis (1999) due to an unbounded derivative. In particular in Selmic and Lewis (1999), jump basis functions

$$\varphi_k(x) = \begin{cases} 0, & \text{for } x < 0 \\ (1 - e^{-x})^k, & \text{for } x \geq 0 \end{cases} \quad (4.B.16)$$

are employed as the activation functions ($\phi_4(k)$ in this case) to approximate the deadzone inverse nonlinear function. However, jump basis functions are not continuously differentiable at the origin. Therefore, the $(k+1)$ th derivative of (4.B.16) at the origin does not exist. Instead, in this Appendix, this problem

is confronted both using Assumption 4.3.4 and employing sufficient number of sigmoidal activation functions since sigmoidal functions are smooth and differentiable. The need for a large number of smooth activation functions for suitable NN approximation is a mild assumption.

Proof of Theorem 4.3.2: The proof of this theorem is given in two cases.

Case I: $\|u(k)\| \leq u_{\max}$

$$1. \|g(k)\| \leq g_m \leq 1$$

Define the Lyapunov function candidate

$$J(k) = \frac{1}{8} \sum_{i=1}^n e_i^T(k) e_i(k) + \sum_{i=1}^3 \frac{1}{\alpha_i} \text{tr}(\tilde{w}_i^T(k) \tilde{w}_i(k)) \quad (4.B.17)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, and $\alpha_3 \in R$ are design parameters (see Theorem 4.3.1). The first difference of Lyapunov function is given by

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) \quad (4.B.18)$$

The term $\Delta J_1(k)$ is obtained using (4.86) as

$$\begin{aligned} \Delta J_1(k) &= \frac{1}{8} (l e_1(k) + g(k) \xi_1(k) - \xi_2(k) + d_1(k))^T (l e_1(k) \\ &\quad + g(k) \xi_1(k) - \xi_2(k) + d_1(k)) - \frac{1}{8} \|e_1(k)\|^2 \\ &\leq \frac{1}{2} l_{\max}^2 \|e_1(k)\|^2 + \frac{1}{2} \|\xi_1(k)\|^2 + \frac{1}{2} \|\xi_2(k)\|^2 \\ &\quad + \frac{1}{2} \|d_1(k)\|^2 - \frac{1}{8} \|e_1(k)\|^2 \end{aligned} \quad (4.B.19)$$

$$2. 1 \leq \|g(k)\| \leq g_m$$

Define the Lyapunov function candidate

$$J(k) = \frac{1}{8g_m^2} \sum_{i=1}^n e_i^T(k) e_i(k) + \sum_{i=1}^3 \frac{1}{\alpha_i} \text{tr}(\tilde{w}_i^T(k) \tilde{w}_i(k)) \quad (4.B.20)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, and $\alpha_3 \in R$ are design parameters (see Theorem 4.3.2). The term $\Delta J_1(k)$ is obtained using (4.86) as

$$\begin{aligned}\Delta J_1(k) &= \frac{1}{8g_m^2} (le_1(k) + g(k)\xi_1(k) - \xi_2(k) + d_1(k))^T (le_1(k) + g(k)\xi_1(k) \\ &\quad - \xi_2(k) + d_1(k)) - \frac{1}{8g_m^2} \|e_1(k)\|^2 \\ &\leq \frac{1}{2g_m^2} (l_{\max}^2 \|e_1(k)\|^2 + \|\xi_2(k)\|^2 + \|d_1(k)\|^2) + \frac{1}{2} \|\xi_1(k)\|^2 \\ &\quad - \frac{1}{8g_m^2} \|e_1(k)\|^2\end{aligned}\tag{4.B.21}$$

Combining (4.B.19) and (4.B.21), in both cases: $\|g(k)\| \leq g_m \leq 1$ and $1 \leq \|g(k)\| \leq g_m$, we have

$$\begin{aligned}\Delta J_1(k) &\leq \frac{1}{2} l_{\max}^2 \|e_1(k)\|^2 + \frac{1}{2} \|\xi_1(k)\|^2 + \frac{1}{2} \|\xi_2(k)\|^2 + \frac{1}{2} \|d_1(k)\|^2 \\ &\quad - \frac{1}{8g_m^2} \|e_1(k)\|^2\end{aligned}\tag{4.B.22}$$

Substituting (4.86), (4.94), through (4.96) in (4.B.22) to obtain

$$\begin{aligned}\Delta J(k) &\leq \frac{1}{2} l_{\max}^2 \|e_1(k)\|^2 - \frac{1}{2} \|\xi_1(k)\|^2 - \frac{1}{2} \|\xi_2(k)\|^2 - \|\xi_3(k)\|^2 + \frac{1}{2} \|d_1(k)\|^2 \\ &\quad - \frac{1}{8g_m^2} \|e_1(k)\|^2 - (1 - \alpha_1 \|\phi_1(k)\|^2) \\ &\quad \times \|\xi_1(k) + (w_1^T \phi_1(k) + Cle_1(k) + AR(k))\|^2 \\ &\quad - (1 - \alpha_2 \|\phi_2(k)\|^2) \|\xi_2(k) + (w_2^T \phi_2(k) + Dle_1(k) + BR(k))\|^2 \\ &\quad - (1 - \alpha_3 \|\phi_3(k)\|^2) \|\xi_3(k) + (w_3^T \phi_3(k) + Ele_1(k))\|^2 \\ &\quad + \|(w_1^T \phi_1(k) + Cle_1(k) + AR(k))\|^2 + \|w_2^T \phi_2(k) + Dle_1(k) \\ &\quad + BR(k)\|^2 + \|w_3^T \phi_3(k) + Ele_1(k)\|^2\end{aligned}\tag{4.B.23}$$

Equation 4.B.23 can be rewritten as

$$\begin{aligned}
 \Delta J(k) &\leq \frac{1}{2}l_{\max}^2 \|e_1(k)\|^2 - \frac{1}{2}\|\xi_1(k)\|^2 - \frac{1}{2}\|\xi_2(k)\|^2 - \|\xi_3(k)\|^2 + \frac{1}{2}\|d_1(k)\|^2 \\
 &\quad - \frac{1}{8g_m^2} \|e_1(k)\|^2 - (1 - \alpha_1 \|\phi_1(k)\|^2) \\
 &\quad \times \|\xi_1(k) + (w_1^T \phi_1(k) + Cle_1(k) + AR(k))\|^2 \\
 &\quad - (1 - \alpha_2 \|\phi_2(k)\|^2) \|\xi_2(k) + (w_2^T \phi_2(k) + Dle_1(k) + BR(k))\|^2 \\
 &\quad - (1 - \alpha_3 \|\phi_3(k)\|^2) \|\xi_3(k) + (w_3^T \phi_3(k) + Ele_1(k))\|^2 \\
 &\quad + 3\|w_1^T \phi_1(k) + Aw_3^T \phi_3(k)\|^2 + 3l_{\max}^2 \|C\|^2 \|e_1(k)\|^2 + 3\|A\|^2 \|\xi_3(k)\|^2 \\
 &\quad + 3\|w_2^T \phi_2(k) + Bw_3^T \phi_3(k)\|^2 + 3l_{\max}^2 \|D\|^2 \|e_1(k)\|^2 + 3\|B\|^2 \|\xi_3(k)\|^2 \\
 &\quad + 2\|w_3^T \phi_3(k)\|^2 + 2l_{\max}^2 \|E\|^2 \|e_1(k)\|^2
 \end{aligned} \tag{4.B.24}$$

Choose $\|A\|^2 + \|B\|^2 < \frac{1}{6}$, $3\|C\|^2 + 3\|D\|^2 + 2\|E\|^2 < \frac{1}{2}$, and define

$$\begin{aligned}
 D_{1M}^2 &= \frac{1}{2}\|d_1(k)\|^2 + 3\|w_1^T \phi_1(k) + Aw_3^T \phi_3(k)\|^2 + 3\|w_2^T \phi_2(k) + Bw_3^T \phi_3(k)\|^2 \\
 &\quad + 2\|w_3^T \phi_3(k)\|^2
 \end{aligned} \tag{4.B.25}$$

Equation 4.B.24 is expressed as

$$\begin{aligned}
 \Delta J(k) &\leq \left(l_{\max}^2 - \frac{1}{8g_m^2} \right) \|e_1(k)\|^2 - \frac{1}{2}\|\xi_1(k)\|^2 - \frac{1}{2}\|\xi_2(k)\|^2 - \frac{1}{2}\|\xi_3(k)\|^2 \\
 &\quad + D_{1M}^2 - (1 - \alpha_1 \|\phi_1(k)\|^2) \|\xi_1(k) + (w_1^T \phi_1(k) + Cle_1(k) + AR(k))\|^2 \\
 &\quad - (1 - \alpha_2 \|\phi_2(k)\|^2) \|\xi_2(k) + (w_2^T \phi_2(k) + Dle_1(k) + BR(k))\|^2 \\
 &\quad - (1 - \alpha_3 \|\phi_3(k)\|^2) \|\xi_3(k) + (w_3^T \phi_3(k) + Ele_1(k))\|^2
 \end{aligned} \tag{4.B.26}$$

This implies that $\Delta J \leq 0$ as long as (4.97) through (4.98) hold and

$$\|e_n(k)\| > \frac{D_{1M}}{\sqrt{\frac{1}{8g_m^2} - l_{\max}^2}} \tag{4.B.27}$$

or

$$\|\xi_1(k)\| > \sqrt{2}D_{1M} \tag{4.B.28}$$

or

$$\|\xi_2(k)\| > \sqrt{2}D_{1M} \quad (4.B.29)$$

or

$$\|\xi_3(k)\| > \sqrt{2}D_{1M} \quad (4.B.30)$$

Case II: $\|u(k)\| > u_{\max}$

Define the Lyapunov function candidate

$$J(k) = \sum_{i=1}^n e_i^T(k) e_i(k) + \sum_{i=1}^3 \frac{1}{\alpha_i} \text{tr}(\tilde{w}_i^T(k) \tilde{w}_i(k)) \quad (4.B.31)$$

The proof is similar to Case I where by considering error dynamics (4.91) and the weight-updating rules (4.94) through (4.96) into the Lyapunov function, the first difference is given by

$$\begin{aligned} \Delta J(k) \leq & \|w_2^T \varphi(v_2^T x(k)) + d_2(k)\|^2 - \|e_1(k)\|^2 - \|\xi_1(k)\|^2 - \|\xi_2(k)\|^2 \\ & - \|\xi_3(k)\|^2 - (1 - \alpha_1 \|\phi_1(k)\|^2) \|\xi_1(k) + (w_1^T \phi_1(k) + Cle_1(k) \\ & + AR(k))\|^2 - (1 - \alpha_2 \|\phi_2(k)\|^2) \|\xi_2(k) + (w_2^T \phi_2(k) + Dle_1(k) \\ & + BR(k))\|^2 - (1 - \alpha_3 \|\phi_3(k)\|^2) \|\xi_3(k) + (w_3^T \phi_3(k) + Ele_1(k))\|^2 \\ & + 3\|w_1^T \phi_1(k) + Aw_3^T \phi_3(k)\|^2 + 3l_{\max}^2 \|C\|^2 \|e_1(k)\|^2 \\ & + 3\|A\|^2 \|\xi_3(k)\|^2 + 3\|w_2^T \phi_2(k) + Bw_3^T \phi_3(k)\|^2 \\ & + 3l_{\max}^2 \|D\|^2 \|e_1(k)\|^2 + 3\|B\|^2 \|\xi_3(k)\|^2 + 2\|w_3^T \phi_3(k)\|^2 \\ & + 2l_{\max}^2 \|E\|^2 \|e_1(k)\|^2 \end{aligned} \quad (4.B.32)$$

Define

$$\begin{aligned} D_{2M}^2 = & 2\|d_2(k)\|^2 + 2\|w_2^T \varphi(v_2^T x(k))\|^2 + 3\|w_1^T \phi_1(k) + Aw_3^T \phi_3(k)\|^2 \\ & + 3\|w_2^T \phi_2(k) + Bw_3^T \phi_3(k)\|^2 + 2\|w_3^T \phi_3(k)\|^2 \end{aligned} \quad (4.B.33)$$

Choose the matrices A, B, C, D , and E using $\|A\|^2 + \|B\|^2 < 1/6$, $3\|C\|^2 + 3\|D\|^2 + 2\|E\|^2 < 1$. Equation 4.B.32 can be rewritten as

$$\begin{aligned}\Delta J(k) \leq & - (1 - l_{\max}^2) \|e_1(k)\|^2 - \|\xi_1(k)\|^2 - \|\xi_2(k)\|^2 - \frac{1}{2} \|\xi_3(k)\|^2 + D_{2M}^2 \\ & - (1 - \alpha_1 \|\phi_1(k)\|^2) \|\xi_1(k) + (w_1^T \phi_1(k) + Cle_1(k) + AR(k))\|^2 \\ & - (1 - \alpha_2 \|\phi_2(k)\|^2) \|\xi_2(k) + (w_2^T \phi_2(k) + Dle_1(k) + BR(k))\|^2 \\ & - (1 - \alpha_3 \|\phi_3(k)\|^2) \|\xi_3(k) + (w_3^T \phi_3(k) + Ele_1(k))\|^2\end{aligned}\quad (4.B.34)$$

This implies that $\Delta J \leq 0$ as long as (4.41) through (4.43) hold and

$$\|e_n(k)\| > \frac{D_{2M}}{\sqrt{1 - l_{\max}^2}} \quad (4.B.35)$$

or

$$\|\xi_1(k)\| > D_{2M} \quad (4.B.36)$$

or

$$\|\xi_2(k)\| > D_{2M} \quad (4.B.37)$$

or

$$\|\xi_3(k)\| > \sqrt{2}D_{2M} \quad (4.B.38)$$

In both Case I and II, $\Delta J \leq 0$ for all k greater than zero. According to the standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that $e_1(k)$ and the weight estimation errors are UUB. The boundedness of $e_1(k)$ implies that all the tracking errors are bounded from the error system (4.12). The boundedness of $\|\xi_1(k)\|$, $\|\xi_2(k)\|$, and $\|\xi_3(k)\|$ implies that $\|\tilde{w}_1(k)\|$, $\|\tilde{w}_2(k)\|$, and $\|\tilde{w}_3(k)\|$ are bounded, and this further implies that the weight estimates $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded. Therefore all the signals in the closed-loop system are bounded.

Remark: Note in this proof of Case 1, we have selected a nonstandard Lyapunov candidate due to the nature of the error system (4.86), that is, the term of $g(k)\xi_1(k)$ makes the error system different. Conditions (4.97) to (4.99) and (4.B.28) or (4.B.29) or (4.B.30) or (4.B.31) assure that the first difference of both Lyapunov candidates is less than zero.

APPENDIX 4.D

```
% backSim.m
% main file for simulation for backslash controller
% design for discrete-time nonlinear system
% programmed by Qinmin Yang at UMR
% Aug. 23, 2005

clc; clear all; close all;

% system parameter setting
T = 15;
T_step = 0.05;
maxStep = T/T_step;
RAN_MAX = 32676.0;
V_min = -0.1;
V_max = 0.1;

amplitude = 1.0;
period = 4*pi;
w = 0.5;

% control parameter definition
lamda1 = 0.25;
lamda2 = 1/16;
Kv = 0.285;
Kb = 2.0;
alpha = 0.1;
gamma = 0.2;
a = 0.45;
L_km1 = 0;
m = 0.5;
d_plus = 0.2;
d_minus = -0.2;
tau = 0.01;
u = 0.2;
u_b = 0;

% NN parameter initialization
numInput = 7;
numNeuron = 10;
numOutput = 1;
```

```
V = rand(numInput, numNeuron);
W0 = rand(numNeuron, numOutput);
W = W0;
% system initialization
x1_v = zeros(maxStep, 1);
x2_v = zeros(maxStep, 1);
e_v = zeros(maxStep, 1);
xd_v = zeros(maxStep, 1);
time = zeros(maxStep, 1);
t = 0;

x1_v(1) = 0;
x2_v(1) = 0;
x1_b = 0;
x2_b = 0;

% closed-loop system
for step = 1:maxStep
    t = t + T_step;

    d_plus = 0.4*(1+0.1*sin(t));
    d_minus = -0.3*(1+0.1*sin(t));

    x1 = x1_v(step);
    x2 = x2_v(step); % -0.1875*x1(step)/(1+x2(step)^2)
                      + x2(step) + tau;

    xd1 = amplitude*cos(w*(t-T_step));
    xd2 = amplitude*cos(w*t);
    xd_v(step) = xd2;

    xd1_b = amplitude*cos(w*(t-2*T_step));
    xd2_b = amplitude*cos(w*(t-T_step));
    xd1_kp1 = amplitude*cos(w*t);
    xd2_kp1 = amplitude*cos(w*(t+T_step));

    e1_b = x1_b - xd1_b;
    e2_b = x2_b - xd2_b;
    e1 = x1 - xd1;
    e2 = x2 - xd2;
    e_v(step) = e2;
```

```

% e1_kp1 = x1_kp1 - xd1_kp1;
% e2_kp1 = x2_kp1 - xd2_kp1;

r = e2 + lamda1*e1; % + lamda2*e_v(abs(step-3)+1);
r_b = e2_b +lamda1*e1_b;

f = -0.1875*x1/(1+x2^2) + x2;
taud = Kv*r - f + xd2_kp1 - lamda1*e1;
% tau_kp1 = tau; %u_kp1;
L = a*L_km1 + a*taud;
inputNN = [x1;x2;r;xd1;xd2;tau;taud-tau];
NN = W'*tansig(V'*inputNN);

% u_kp1 = 1.3*Kb*r;
u = -Kb*(taud - tau) + L + NN %;

if ((u-u_b>0) & (tau <=m*u_b-m*d_plus))
    taud = m*u-m*d_plus;
elseif ((u-u_b<0) & (tau>=m*u_b-m*d_minus))
    taud = m*u-m*d_minus;
else
    taud = taud;
end
% tau = u; % no backlash

I = ones(numNeuron, numNeuron);
W = W + alpha*tansig(V'*inputNN)*(r+taud-tau)
    - gamma*norm(I-alpha*tansig
        (V'*inputNN)*tansig((V'*inputNN)'))*W;

u_b = u;

% tau = tau_kp1;

x1_v(step+1) = x2;
x2_v(step+1) = -0.1875*x1/(1+x2^2) + x2 + taud;
x1_b = x1;
x2_b = x2;
% u = u_kp1;

L_km1 = L;

```

```
time(step) = t;
end

% simulation results
figure;
plot(time, x2_v(1:maxStep), 'b-', time, xd_v
(1:maxStep), 'r--');
legend('x2','xd');
xlabel('Time (in seconds)'); ylabel('Displacement
(in m)');

figure;
plot(time,e_v);
title('tracking error');
xlabel('Time (in seconds)');
ylabel('tracking error');
legend('error');
```

5 Output Feedback Control of Strict Feedback Nonlinear MIMO Discrete-Time Systems

In Chapter 4, state feedback control of a class of nonlinear system with actuator nonlinearities was presented by assuming that the states are available for measurement. In many industrial applications, all the states are normally not measurable. Only certain outputs are measured and they have to be used as feedback variables. Under those circumstances, one has to design an observer to estimate the states from the measured outputs and then a controller using the estimated states. Therefore, output feedback controller schemes are necessary when certain states of the plant become unavailable for measurement. In this chapter, the output feedback control of a class of multi-input and multi-output (MIMO) nonlinear discrete-time system is presented.

Design of an output feedback controller is very difficult for general nonlinear discrete-time systems. Unless carefully designed, an exponentially decaying state estimation error can lead to system instability in a finite time (Krstic et al. 1995) referred to as escape time for nonlinear systems. Moreover, the separation principle that is normally used to design output feedback control schemes for linear systems does not hold for nonlinear systems. These make the output feedback controller design quite challenging.

Several output feedback controller designs in discrete-time are proposed for single-input-single-out (SISO) nonlinear systems (Chen and Khalil 1995; Yeh and Kokotovic 1995; Ge et al. 2003). In particular, a backstepping-based adaptive output feedback controller scheme is presented

in Yeh and Kokotovic (1995) for the control of a class of strict feedback nonlinear systems, where a rank condition is required to ensure the boundedness of all signals. In Chen and Khalil (1995), a discrete-time NN output feedback controller is designed for a class of nonlinear systems expressed in input–output fashion, where the system is assumed to be minimum phase. A deadzone algorithm is used to develop a well-defined controller. Two output feedback control schemes are given in discrete time based on a causal input–output representation and by using an adaptive NN observer, respectively (Ge et al. 2003). The semiglobal UUB of the closed-loop system is demonstrated. However, SISO controller designs cannot be directly extended to the proposed MIMO case.

In this chapter, the output feedback controller design from He and Jagannathan (2005) using an adaptive critic neural network (NN) architecture is presented for an unknown MIMO nonlinear discrete system. As presented in Chapter 4, the reinforcement learning-based adaptive critic NN approach (Werbos 1992; Bertsekas and Tsitsiklis 1996; Lin and Balakrishnan 2000; Murray et al. 2002) has emerged as a promising tool to develop suboptimal NN controllers due to its potential in finding approximate solutions to dynamic programming, where a strategic utility function, which is considered as the long-term system performance measure, can be optimized. The adaptive critic output feedback NN controller (He and Jagannathan 2005) consists of (1) an NN observer to estimate the system states by using the input–output data, (2) an action NN to drive the output to track the reference signal and to minimize both the strategic utility function and the unknown dynamics estimation errors, and (3) an adaptive critic NN to approximate certain strategic utility function and to tune the weights of the action NN. The strategic utility function is similar to the standard Bellman equation. As explained in the previous chapter, the adaptive critic controller can be viewed quite similar to supervised actor-critic reinforcement learning method. With incomplete information of the system states and dynamics, an approximate optimization is accomplished using the proposed controller. Further, the actuator magnitude constraints are incorporated as saturation nonlinearities during the controller development in contrast to other works where no explicit magnitude constraints are treated.

Besides optimization and the incorporation of the input constraints, contributions of this paper (He and Jagannathan 2005) to the literature includes (1) the demonstration of the UUB of the overall system in the presence of NN approximation errors and bounded unknown disturbances unlike in the existing adaptive critic works (Werbos 1992; Bertsekas and Tsitsiklis 1996; Lin and Balakrishnan 2000; Murray et al. 2002) where the convergence is given under ideal circumstances, (2) the NN weights are tuned online instead of offline training that is commonly employed in adaptive critic

design (Werbos 1992, Prokhorov and Wunsch 1997), and (3) the linear in the parameter assumption (LIP) is overcome along with the persistent excitation (PE) condition requirement (Ge et al. 2003) both in NN observer and controller designs.

5.1 CLASS OF NONLINEAR DISCRETE-TIME SYSTEMS

Consider the following nonlinear system, to be controlled, given in the following form

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_n(k+1) &= f(x(k)) + g(x(k))u(k) + d'(k) \end{aligned} \tag{5.1}$$

$$y(k) = x_1(k) \tag{5.2}$$

with state $x(k) = [x_1^T(k), x_2^T(k), \dots, x_n^T(k)]^T \in R^{nm}$, and each $x_i(k) \in R^m$, $i = 1, \dots, n$ is the state at time instant k , $f(x(k)) \in R^m$ is the unknown nonlinear function vector, $g(x(k)) \in R^{m \times m}$ is a diagonal matrix of unknown nonlinear functions, $u(k) \in R^m$ is the control input vector, and $d'(k) \in R^m$ is the unknown but bounded disturbance vector, whose bound is assumed to be a known constant, $\|d'(k)\| \leq d'_m$ where the Frobenius norm (Lewis et al. 1999) will be used throughout this paper. It is assumed that the output, $y(k) \in R^m$, is known at the k th time instant and the state vector $x_i(k) \in R^m$, $i = 2, \dots, n$ is considered to be unavailable at the k th step.

Assumption 5.1.1: Let the diagonal matrix $g(x(k)) \in R^{m \times m}$ be a positive definite matrix for each $x(k) \in R^{nm}$, with $g_{\min} \in R^+$ and $g_{\max} \in R^+$ representing the minimum and maximum eigenvalues of the matrix $g(x(k)) \in R^{m \times m}$, respectively, such that $0 < g_{\min} < g_{\max}$.

5.2 OUTPUT FEEDBACK CONTROLLER DESIGN

We will first design an observer and then a controller. Since the separation principle that is commonly applied to linear systems design is no longer valid for nonlinear systems, the Lyapunov function candidate should consist of quadratic error terms resulting from the observer and the controller. This makes the output feedback controller design and overall stability analysis difficult.

5.2.1 OBSERVER DESIGN

For the system described by (5.1) and (5.2), we use the following state observer to estimate the state $x(k)$.

$$\begin{aligned}\hat{x}_1(k) &= \hat{x}_2(k-1) \\ &\vdots \\ \hat{x}_n(k) &= \hat{w}_1^T(k-1)\phi_1(v_1^T\hat{z}_1(k-1)) = \hat{w}_1^T(k-1)\phi_1(\hat{z}_1(k-1))\end{aligned}\tag{5.3}$$

where $\hat{x}_i(k) \in R^m$ is the estimated state of $x_i(k) \in R^m$ with $i = 1, \dots, n$ and $\hat{z}_1(k-1) = [\hat{x}_1^T(k-1), \dots, \hat{x}_n^T(k-1), u^T(k-1)]^T \in R^{(n+1)m}$ is the input vector to the NN observer at the k th instant, $\hat{w}_1(k-1) \in R^{n_1 \times m}$ and $v_1 \in R^{(n+1)m \times n_1}$ denote the output- and hidden-layer weight matrices, and n_1 is the number of the hidden-layer nodes. For simplicity, the hidden-layer activation function vector $\phi_1(v_1^T\hat{z}_1(k-1)) \in R^{n_1}$ is written as $\phi_1(\hat{z}_1(k-1))$. It is demonstrated in Igelnik and Pao (1995) that, if the hidden-layer weight matrix, v_1 , is chosen initially at random and kept constant and n_1 is sufficiently large, the NN approximation error can be made arbitrarily small since the hidden-layer activation function vector forms a basis.

Define the state estimation error by

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k) \quad i = 1, \dots, n \tag{5.4}$$

where $\tilde{x}_i(k) \in R^m$, $i = 1, \dots, n$, is the state estimation error. In fact, the NN observer approximates the nonlinear function given by $f(x(k-1)) + g(x(k-1))u(k-1)$. This nonlinear function can be expressed as

$$\begin{aligned}f(x(k-1)) + g(x(k-1))u(k-1) &= w_1^T\phi_1(v_1^Tz_1(k-1)) + \varepsilon_1(z_1(k-1)) \\ &= w_1^T\phi_1(z_1(k-1)) + \varepsilon_1(z_1(k-1))\end{aligned}\tag{5.5}$$

where $w_1 \in R^{n_1 \times m}$ is the target NN weight matrix, $\varepsilon_1(z_1(k-1))$ is the NN approximation error, and the NN input is given by $z_1(k-1) = [x_1^T(k-1), \dots, x_n^T(k-1), u^T(k-1)]^T \in R^{(n+1)m}$. For convenience, the hidden-layer activation function vector $\phi_1(v_1^Tz_1(k-1)) \in R^{n_1}$ is written as $\phi_1(z_1(k-1))$.

Combining (5.3) to (5.5) we get

$$\begin{aligned}
\tilde{x}_n(k) &= \hat{x}_n(k) - x_n(k) = \hat{x}_n(k) - f(x(k-1)) \\
&\quad + g(x(k-1))u(k-1) - d'(k-1) \\
&= \hat{w}_1^T(k-1)\phi_1(\hat{z}_1(k-1)) - w_1^T\phi_1(z_1(k-1)) \\
&\quad - \varepsilon_1(z_1(k-1)) - d'(k-1) \\
&= (\hat{w}_1^T(k-1) - w_1^T)\phi_1(\hat{z}_1(k-1)) + w_1^T(\phi_1(\hat{z}_1(k-1)) \\
&\quad - \phi_1(z_1(k-1))) - \varepsilon_1(z_1(k-1)) - d'(k-1) \\
&= \tilde{w}_1^T(k-1)\phi_1(\tilde{z}_1(k-1)) + w_1^T\phi_1(\tilde{z}_1(k-1)) \\
&\quad - \varepsilon_1(z_1(k-1)) - d'(k-1) \\
&= \xi_1(k-1) + d_1(k-1)
\end{aligned} \tag{5.6}$$

where

$$\tilde{w}_1(k-1) = \hat{w}_1(k-1) - w_1 \tag{5.7}$$

$$\xi_1(k-1) = \tilde{w}_1^T(k-1)\phi_1(\tilde{z}_1(k-1)) \tag{5.8}$$

$$\phi_1(\tilde{z}_1(k-1)) = \phi_1(\hat{z}_1(k-1)) - \phi_1(z_1(k-1)) \tag{5.9}$$

$$d_1(k-1) = w_1^T\phi_1(\tilde{z}_1(k-1)) - (\varepsilon_1(z_1(k-1)) + d'(k-1)) \tag{5.10}$$

The dynamics of the estimation error using (5.4) and (5.6) are obtained as

$$\begin{aligned}
\tilde{x}_1(k) &= \tilde{x}_2(k-1) \\
&\vdots \\
\tilde{x}_n(k) &= \xi_1(k-1) + d_1(k-1)
\end{aligned} \tag{5.11}$$

5.2.2 NN CONTROLLER DESIGN

Our objective is to design an adaptive critic NN output feedback controller by incorporating the magnitude constraints for the system (5.1) and (5.2) (*note:* the states $x_i(k)$, $i = 2, \dots, n$ are not measurable at the k th time instant) such that (1) all the signals in the closed-loop system remain UUB; (2) the state $x(k)$ follows a desired trajectory $Y_d(k) = [y_d^T(k), \dots, y_d^T(k+n-1)]^T \in R^{nm}$, with $y_d(k) \in R^m$ and $y_d(k+i)$ representing the future value of $y_d(k)$, $i = 1, \dots, n-1$; and (3) a long-term system performance index similar to the HJB equation, is optimized.

Assumption 5.2.1: The desired trajectory, $Y_d(k)$, is a smooth bounded function over the compact subset of R^{nm} .

5.2.2.1 Auxiliary Controller Design

Define the tracking error between actual and desired trajectory as

$$e_i(k+1) = x_i(k+1) - y_d(k+i) \quad i = 1, \dots, n \quad (5.12)$$

Equation 5.1 can be rewritten as

$$e_n(k+1) = f(x(k)) + g(x(k))u(k) + d'(k) - y_d(k+n) \quad (5.13)$$

Define the desired auxiliary control signal as

$$v_d(k) = g^{-1}(x(k))(-f(x(k)) + y_d(k+n) + l_1 e_n(k)) \quad (5.14)$$

where $l_1 \in R^{m \times m}$ is a design matrix selected such that the tracking error, $e_n(k)$, is bounded.

Since $f(x(k))$ and $g(x(k))$ are unknown smooth functions, the desired auxiliary feedback control input $v_d(k)$ cannot be implemented in practice. From (5.14) and using Assumptions 5.1.1 and 5.2.1, $v_d(k)$ can be approximated by the action NN as

$$v_d(k) = w_2^T \phi_2(v_2^T s(k)) + \varepsilon_2(s(k)) = w_2^T \phi_2(s(k)) + \varepsilon_2(s(k)) \quad (5.15)$$

where $s(k) = [x^T(k), e_n^T(k)]^T \in R^{(n+1)m}$ is the NN input vector, $w_2 \in R^{n_2 \times m}$ and $v_2 \in R^{(n+1)m \times n_2}$ denote the output- and hidden-layer target weights, respectively, $\varepsilon_2(s(k))$ is the action NN approximation error, and n_2 is the number of the nodes in the hidden-layer. For the purpose of simplicity, the hidden-layer activation function vector $\phi_2(v_2^T s(k)) \in R^{n_2}$ is written as $\phi_2(s(k))$.

Since the states $x_i(k)$, $i = 2, \dots, n$ are not measurable at the k th time instant, replacing the actual states with their estimated values, (5.15) can be expressed as

$$v(k) = \hat{w}_2^T(k) \phi_2(v_2^T \hat{s}(k)) = \hat{w}_2^T(k) \phi_2(\hat{s}(k)) \quad (5.16)$$

where $\hat{w}_2(k) \in R^{n_2 \times m}$ is the actual weight matrix, $\hat{s}(k) = [\hat{x}^T(k), \hat{e}_n^T(k)]^T \in R^{(n+1)m}$ is the action NN input, with $\hat{e}_n(k) \in R^m$ referred to as the modified tracking error, which is defined between the estimated state vector and the desired trajectory as

$$\hat{e}_i(k+1) = \hat{x}_i(k+1) - y_d(k+i) \quad i = 1, \dots, n \quad (5.17)$$

and

$$\hat{e}(k) = \begin{bmatrix} \hat{e}_1(k) \\ \vdots \\ \hat{e}_n(k) \end{bmatrix} = \begin{bmatrix} \hat{x}_1(k) - y_d(k) \\ \vdots \\ \hat{x}_n(k) - y_d(k+n-1) \end{bmatrix} \quad (5.18)$$

5.2.2.2 Controller Design with Magnitude Constraints

By applying the magnitude constraints, the actual control input $u(k) \in R^m$ is now given by

$$u(k) = \begin{cases} v(k), & \text{if } \|v(k)\| \leq u_{\max} \\ u_{\max} \operatorname{sgn}(v(k)), & \text{if } \|v(k)\| \geq u_{\max} \end{cases} \quad (5.19)$$

where u_{\max} is the upper limit defined by the actuator. Now the design has to be done in two cases.

Case I: $\|v(k)\| \leq u_{\max}$

In this case, the control input $u(k) = v(k)$. Substituting (5.16) through (5.18) into (5.13) yields

$$\begin{aligned} e_n(k+1) &= f(x(k)) + g(x(k))v(k) + d'(k) - y_d(k+n) \\ &= f(x(k)) + g(x(k))(v_d(k) + v(k) - v_d(k)) + d'(k) - y_d(k+n) \\ &= l_1 e_n(k) + g(x(k))(v(k) - v_d(k)) + d'(k) \\ &= l_1 e_n(k) + g(x(k))(\hat{w}_2^T(k)\phi_2(\hat{s}(k)) \\ &\quad - w_2^T\phi_2(s(k)) + \varepsilon_2(s(k))) + d'(k) \\ &= l_1 e_n(k) + g(x(k))((\hat{w}_2^T(k) - w_2^T)\phi_2(\hat{s}(k)) \\ &\quad + w_2^T(\phi_2(\hat{s}(k)) - \phi_2(s(k))) + \varepsilon_2(s(k))) + d'(k) \\ &= l_1 e_n(k) + g(x(k))\zeta_2(k) + g(x(k))(w_2^T\phi_2(\tilde{s}(k)) \\ &\quad + \varepsilon_2(s(k))) + d'(k) \\ &= l_1 e_n(k) + g(x(k))\zeta_2(k) + d_2(k) \end{aligned} \quad (5.20)$$

where

$$\tilde{w}_2(k) = \hat{w}_2(k) - w_2 \quad (5.21)$$

$$\zeta_2(k) = \tilde{w}_2^T(k)\phi_2(\hat{s}(k)) \quad (5.22)$$

$$\phi_2(\tilde{s}(k)) = \phi_2(\hat{s}(k)) - \phi_2(s(k)) \quad (5.23)$$

$$d_2(k) = g(x(k))(w_2^T\phi_2(\tilde{s}(k)) - \varepsilon_2(s(k))) + d'(k) \quad (5.24)$$

Thus, the tracking error dynamics is given by

$$\begin{aligned} e_1(k+1) &= e_2(k) \\ &\vdots \\ e_n(k+1) &= l_1 e_n(k) + g(x(k)) \zeta_2(k) + d_2(k) \end{aligned} \quad (5.25)$$

Case II: $\|v(k)\| \geq u_{\max}$

In this case, the control input $u(k) = u_{\max} \text{sgn}(v(k))$. Combining (5.13) through (5.16) to get

$$\begin{aligned} e_n(k+1) &= f(x(k)) + g(x(k))u(k) + d'(k) - y_d(k+n) \\ &= f(x(k)) + g(x(k))(u(k) + v_d(k) - v_d(k)) + d'(k) - y_d(k+n) \\ &= l_1 e_n(k) + g(x(k))(u_{\max} \text{sgn}(v(k)) - w_2^T \phi_2(s(k)) - \varepsilon_2(s(k))) + d'(k) \\ &= l_1 e_n(k) + d'_2(k) \end{aligned} \quad (5.26)$$

where

$$d'_2(k) = g(x(k))(u_{\max} \text{sgn}(v(k)) - w_2^T \phi_2(s(k)) - \varepsilon_2(s(k))) + d'(k) \quad (5.27)$$

Therefore, for Case II, the tracking error dynamics are written as

$$\begin{aligned} e_1(k+1) &= e_2(k) \\ &\vdots \\ e_n(k+1) &= l_1 e_n(k) + d'_2(k) \end{aligned} \quad (5.28)$$

5.3 WEIGHT UPDATES FOR GUARANTEED PERFORMANCE

The next step is to design the observer, action, and critic NNs using Lyapunov analysis.

5.3.1 WEIGHTS UPDATING RULE FOR THE OBSERVER NN

The weight update for the observer NN is driven by the state estimation error $\tilde{x}_1(k)$ and it is given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(\hat{z}_1(k)) (\hat{w}_1^T(k) \phi_1(\hat{z}_1(k)) + l_2 \tilde{x}_1(k))^T \quad (5.29)$$

where $l_2 \in R^{m \times m}$ is a design matrix, and $\alpha_1 \in R^+$ is the adaptation gain of the NN observer.

5.3.2 STRATEGIC UTILITY FUNCTION

The utility function $p(k) = [p_i(k)]_{i=1}^m \in R^m$ is defined based on the modified tracking error $\hat{e}_n(k)$ and it is given by

$$p_i(k) = \begin{cases} 0, & \text{if } \text{abs}(e_n^i(k)) \leq c \\ 1, & \text{otherwise} \end{cases} \quad i = 1, 2, \dots, m \quad (5.30)$$

where $e_n^i(k) \in R$ is the i th element of vector $e_n(k)$, $\text{abs}(e_n^i(k))$ is the absolute value of $e_n^i(k)$, $c \in R^+$ is a predefined threshold. The utility function $p(k)$ is viewed as the current system performance index: $p_i(k) = 0$ and $p_i(k) = 1$ refer to good and unacceptable tracking performances, respectively.

The strategic utility function $Q(k) \in R^m$, is defined as

$$Q^T(k) = \alpha^N p(k+1) + \alpha^{N-1} p(k+2) + \dots + \alpha^{k+1} p(N) + \dots \quad (5.31)$$

where $\alpha \in R$ is a design parameter, $0 < \alpha < 1$, and N is the stage number. The term $Q^T(k)$ is viewed here as the long-term performance measure since it is the sum of all future system performance indices. Equation 5.31 can also be expressed as $Q^T(k) = \min_{u(k)} \{\alpha Q(k-1) - \alpha^{N+1} p(k)\}$. This measure is quite similar to the standard Bellman equation (Prokhorov and Wunsch 1997; Si and Wang 2001).

5.3.3 CRITIC NN DESIGN

The critic NN is employed to approximate the strategic utility function $Q(k)$, since $Q(k)$ is unavailable at the k th time instant. The critic signal is then used to tune the action NN to minimize $Q(k)$. The prediction error is defined as

$$e_c(k) = \hat{Q}(k) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)) \quad (5.32)$$

where the subscript c stands for the critic. The critic signal $\hat{Q}(k) \in R^m$ is given by

$$\hat{Q}(k) = \hat{w}_3^T(k) \phi_3(v_3^T \hat{x}(k)) = \hat{w}_3^T(k) \phi_3(\hat{x}(k)) \quad (5.33)$$

$\hat{w}_3(k) \in R^{n_3 \times m}$ and $v_3 \in R^{nm \times n_3}$ represent the matrices of weight estimates, n_3 is the number of the nodes in the hidden layer, and the state estimate $\hat{x}(k) = [\hat{x}_1^T(k), \dots, \hat{x}_n^T(k)]^T \in R^{nm}$ is the critic NN input. The activation function vector

of the hidden layer $\phi_3(v_3^T \hat{x}(k)) \in R^{n_3}$ is written as $\phi_3(\hat{x}(k))$. Now the objective function to be minimized by the critic NN is defined as

$$E_c(k) = \frac{1}{2} e_c^T(k) e_c(k) \quad (5.34)$$

The weight-update rule for the critic NN is a gradient-based adaptation, which is given by

$$\hat{w}_3(k+1) = \hat{w}_3(k) + \Delta \hat{w}_3(k) \quad (5.35)$$

where

$$\Delta \hat{w}_3(k) = \alpha_3 \left[-\frac{\partial E_c(k)}{\partial \hat{w}_3(k)} \right] \quad (5.36)$$

with $\alpha_3 \in R$ is the adaptation gain. Before we proceed further, the following Lemma is required.

Lemma 5.3.1: Given the matrices $A \in R^{m \times m}$, $X \in R^{n \times m}$ and vectors $b \in R^n$ and $q \in R^m$, the derivative of the following quadratic term with respect to the matrix X is given by

$$\frac{d((AX^T b + q)^T(AX^T b + q))}{dX} = 2b(A^T(AX^T b + q))^T \quad (5.37)$$

where the matrix A , vectors b and q are independent of the matrix X .

Proof: See Appendix 5.A.

Combining (5.32) to (5.34) with (5.36), the critic NN weight-updating rule is obtained as

$$\begin{aligned} \Delta \hat{w}_3(k) &= -\frac{1}{2} \alpha_3 \left[\frac{\partial e_c^T(k) e_c(k)}{\partial \hat{w}_3(k)} \right] \\ &= -\frac{1}{2} \alpha_3 [\partial[(\hat{Q}(k) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)))^T(\hat{Q}(k) - \alpha(\hat{Q}(k-1) \\ &\quad - \alpha^N p(k)))](\partial \hat{w}_3(k))^{-1}] \\ &= -\frac{1}{2} \alpha_3 [\partial[(\hat{w}_3^T(k) \phi_3(\hat{x}(k)) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)))^T(\hat{w}_3^T(k) \phi_3(\hat{x}(k)) \\ &\quad - \alpha(\hat{Q}(k-1) - \alpha^N p(k)))](\partial \hat{w}_3(k))^{-1}] \end{aligned} \quad (5.38)$$

Using *Lemma 5.3.1* (*note*: in this case, A is an identity matrix), (5.38) can be simplified as

$$\begin{aligned}\Delta \hat{w}_3(k) &= -\alpha_3 \phi_3(\hat{x}(k)) (\hat{w}_3^T(k) \phi_3(\hat{x}(k)) - \alpha(\hat{Q}(k-1) - \alpha^N p(k)))^T \\ &= -\alpha_3 \phi_3(\hat{x}(k)) (\hat{Q}(k) + \alpha^{N+1} p(k) - \alpha \hat{Q}(k-1))^T\end{aligned}\quad (5.39)$$

Thus the critic NN weight-updating rule is obtained as

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(\hat{x}(k)) (\hat{Q}(k) + \alpha^{N+1} p(k) - \alpha \hat{Q}(k-1))^T \quad (5.40)$$

5.3.4 WEIGHT-UPDATING RULE FOR THE ACTION NN

The action NN weights $\hat{w}_2^T(k)$ are tuned by using the functional estimation error, $\zeta_2(k)$ and the error between the desired strategic utility function $Q_d(k) \in R^m$ and the critic signal $\hat{Q}(k)$. Define

$$e_a(k) = \sqrt{g(x(k))} \zeta_2(k) + (\sqrt{g(x(k))})^{-1} (\hat{Q}(k) - Q_d(k)) \quad (5.41)$$

where $\zeta_2(k)$ is defined in (5.22), $\sqrt{g(x(k))} \in R^{m \times m}$ is the principle square root of the diagonal positive definite matrix $g(x(k))$, that is, $(\sqrt{g(x(k))})^2 = g(x(k))$, and $(\sqrt{g(x(k))})^T = (\sqrt{g(x(k))})$, $e_a(k) \in R^m$, and the subscript a stands for the action NN.

The desired strategic utility function $Q_d(k)$ is considered to be zero (0) (Si and Wang 2001) to indicate that at every step the nonlinear system can track the reference signal well. Thus, (5.41) becomes

$$e_a(k) = \sqrt{g(x(k))} \zeta_2(k) + (\sqrt{g(x(k))})^{-1} \hat{Q}(k) \quad (5.42)$$

The objective function to be minimized is given by

$$E_a(k) = \frac{1}{2} e_a^T(k) e_a(k) \quad (5.43)$$

Combining (5.22), (5.42) with (5.43), we get

$$\begin{aligned}
\Delta \hat{w}_2(k) &= -\frac{1}{2} \alpha_2 \left[\frac{\partial e_a^T(k) e_a(k)}{\partial \hat{w}_2(k)} \right] \\
&= -\frac{1}{2} \alpha_2 \left[\partial \left[(\sqrt{g(x(k)))} \zeta_2(k) + (\sqrt{g(x(k)))}^{-1} \hat{Q}(k))^T (\sqrt{g(x(k)))} \zeta_2(k) \right. \right. \\
&\quad \left. \left. + (\sqrt{g(x(k)))}^{-1} \hat{Q}(k)) \right] (\partial \hat{w}_2(k))^{-1} \right] \\
&= -\frac{1}{2} \alpha_2 \left[\partial \left[(\sqrt{g(x(k)))} (\hat{w}_2^T(k) \phi_2(\hat{s}(k))) + ((\sqrt{g(x(k)))}^{-1} \hat{Q}(k) \right. \right. \\
&\quad \left. \left. - \sqrt{g(x(k)))} (w_2^T \phi_2(\hat{s}(k)))) \right)^T (\sqrt{g(x(k)))} (\hat{w}_2^T(k) \phi_2(\hat{s}(k))) \right. \\
&\quad \left. + ((\sqrt{g(x(k)))}^{-1} \hat{Q}(k) - \sqrt{g(x(k)))} (w_2^T \phi_2(\hat{s}(k)))) \right] (\partial \hat{w}_2(k))^{-1} \right] \\
&\quad (5.44)
\end{aligned}$$

Using Lemma 5.3.1, (5.44) is simplified as

$$\begin{aligned}
\Delta \hat{w}_2(k) &= -\alpha_2 \phi_2(\hat{s}(k)) \left((\sqrt{g(x(k)))}^T (\sqrt{g(x(k)))} (\hat{w}_2^T(k) \phi_2(\hat{s}(k))) \right. \\
&\quad \left. + ((\sqrt{g(x(k)))}^{-1} \hat{Q}(k) - \sqrt{g(x(k)))} (w_2^T \phi_2(\hat{s}(k)))) \right)^T \\
&= -\alpha_2 \phi_2(\hat{s}(k)) \left((\sqrt{g(x(k)))} (\sqrt{g(x(k)))} (\hat{w}_2^T(k) \phi_2(\hat{s}(k))) \right. \\
&\quad \left. + ((\sqrt{g(x(k)))}^{-1} \hat{Q}(k) - \sqrt{g(x(k)))} (w_2^T \phi_2(\hat{s}(k)))) \right)^T \\
&= -\alpha_2 \phi_2(\hat{s}(k)) (g(x(k))) \zeta_2(k) + \hat{Q}(k) \\
&\quad (5.45)
\end{aligned}$$

Equation 5.45 can be further expressed using (5.25) as

$$\Delta \hat{w}_2(k) = -\alpha_2 \phi_2(\hat{s}(k)) (e_n(k+1) - l_1 e_n(k) - d_2(k) + \hat{Q}(k))^T \quad (5.46)$$

The weight-updating rule for the action NN is given by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{s}(k)) (e_n(k+1) - l_1 e_n(k) - d_2(k) + \hat{Q}(k))^T \quad (5.47)$$

where $\alpha_2 \in R^+$ is the adaptation gain of the action NN. Since $e_n(k+1)$ and $e_n(k)$ are unavailable, the modified tracking errors $\hat{e}_n(k+1)$ and $\hat{e}_n(k)$,

respectively, are used instead. In the ideal case, we take the disturbance $d_2(k)$ as zero to obtain the action NN $\hat{w}_2^T(k)\phi_2(\hat{s}(k))$ weight-updating rule as

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2\phi_2(\hat{s}(k))(\hat{e}_n(k+1) - l_1\hat{e}_n(k) + \hat{Q}(k))^T \quad (5.48)$$

Assumption 5.3.1: Let w_1 , w_2 , and w_3 be the unknown output-layer target weights for the observer, action, and critic NN, and assume that they are bounded above so that

$$\|w_1\| \leq w_{1m} \quad \|w_2\| \leq w_{2m} \quad \text{and} \quad \|w_3\| \leq w_{3m} \quad (5.49)$$

where $w_{1m} \in R^+$, $w_{2m} \in R^+$, and $w_{3m} \in R^+$ represent the bounds on the unknown target weights.

Fact 5.3.1: The activation functions are bounded by known positive values so that

$$\|\phi_i(k)\| \leq \phi_{im} \quad i = 1, 2, 3 \quad (5.50)$$

where $\phi_{im} \in R^+$, $i = 1, 2, 3$ is the upper bound for $\phi_i(k)$, $i = 1, 2, 3$.

Assumption 5.3.2: The NN approximation errors $\varepsilon_1(z_1(k))$ and $\varepsilon_2(s(k))$ are bounded over the compact set $S \subset R^m$ by ε_{1m} and ε_{2m} , respectively (Lewis et al. 1999).

Fact 5.3.2: With the Assumptions (5.3.1), (5.3.2), and Fact 5.3.1, the terms $d_1(k)$ (Equation 5.10), $d_2(k)$ (Equation 5.24), and $d'_2(k)$ (Equation 5.27) are bounded over the compact set $S \subset R^m$ by

$$\|d_1(k)\| \leq d_{1m} = 2w_{1m}\phi_{1m} + d'_m + \varepsilon_{1m} \quad (5.51)$$

$$\|d_2(k)\| \leq d_{2m} = 2g_{\max}w_{1m}\phi_{1m} + \varepsilon_{2m} + d'_m \quad (5.52)$$

and

$$\|d'_2(k)\| \leq d'_{2m} = g_{\max}(u_{\max} + w_{2m}\phi_{2m} + \varepsilon_{2m}) + d'_m \quad (5.53)$$

where $d_{1m} \in R^+$, $d_{2m} \in R^+$, and $d'_{2m} \in R^+$ are the upper bounds for $d_1(k)$, $d_2(k)$, and $d'_2(k)$, respectively.

Theorem 5.3.1 (NN Output Feedback Controller): Consider the system given by (5.1) and (5.2). Let the Assumptions 5.1.1, 5.2.1, 5.3.1, and 5.3.2 hold

with the disturbance bound d'_m a known constant. Let the state estimate vector and control input be provided by the observer (5.3) and (5.19), respectively. Let the NN observer, $\hat{w}_1^T(k)\phi_1(\hat{z}_1(k))$, action NN, $\hat{w}_2^T(k)\phi_2(\hat{s}(k))$, and the critic NN $\hat{w}_3^T(k)\phi_3(\hat{x}(k))$ weight tuning be given by (5.29), (5.40), and (5.48), respectively, as shown in Table 5.1. Then the state estimation error $\tilde{x}_i(k)$, the tracking error $e_i(k)$, and the NN weight estimates, $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are UUB, with the bounds specifically given by (5.B.14) through (5.B.18) provided the controller design parameters are selected as:

$$(a) \quad 0 < \alpha_1 \|\phi_1(\hat{z}_1(k))\|^2 < 1 \quad (5.54)$$

$$(b) \quad 0 < \alpha_2 \|\phi_2(k)\|^2 < \min\left(\frac{g_{\min}}{g_{\max}^2}, \frac{1}{g_{\min}}\right) \quad (5.55)$$

$$(c) \quad 0 < \alpha_3 \|\phi_3(\hat{x}(k))\|^2 < 1 \quad (5.56)$$

$$(d) \quad 0 < \alpha < \frac{\sqrt{2}}{2} \quad (5.57)$$

where α_1 , α_2 , and α_3 are NN adaptation gains, and α is a parameter in the strategic utility function.

Proof: See Appendix 5.B.

Remarks:

1. The separation principle is not used in the above design since estimation errors from the observer are included as part of the same Lyapunov function candidate that is used for controller analysis.
2. It is important to note that in this theorem certainty equivalence (CE), persistence of excitation condition (PE), and LIP assumptions are relaxed for the NN controller, in contrast to standard works in discrete-time adaptive output feedback control (Yeh and Kokotovic 1995). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption, and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking

TABLE 5.1
Adaptive NN Output Feedback Controller

The actual control input $u(k) \in R^m$ is given by

$$u(k) = \begin{cases} v(k), & \text{if } \|v(k)\| \leq u_{\max} \\ u_{\max} \operatorname{sgn}(v(k)), & \text{if } \|v(k)\| \geq u_{\max} \end{cases}$$

where u_{\max} is the upper limit defined by the actuator. The auxiliary control signal is defined as

$$v(k) = \hat{w}_2^T(k) \phi_2(v_2^T \hat{s}(k)) = \hat{w}_2^T(k) \phi_2(\hat{s}(k))$$

where the action NN input is given by

$$\hat{s}(k) = [\hat{x}^T(k), \hat{e}_n^T(k)]^T \in R^{(n+1)m},$$

with $\hat{e}_n(k) \in R^m$ referred to as the modified tracking error, which is defined as

$$\hat{e}_i(k+1) = \hat{x}_i(k+1) - y_d(k+i), \quad i = 1, \dots, n$$

and

$$\hat{e}(k) = \begin{bmatrix} \hat{e}_1(k) \\ \vdots \\ \hat{e}_n(k) \end{bmatrix} = \begin{bmatrix} \hat{x}_1(k) - y_d(k) \\ \vdots \\ \hat{x}_n(k) - y_d(k+n-1) \end{bmatrix}$$

The utility function $p(k) = [p_i(k)]_{i=1}^m \in R^m$ is defined as

$$p_i(k) = \begin{cases} 0, & \text{if } \operatorname{abs}(e_n^i(k)) \leq c \\ 1, & \text{otherwise} \end{cases} \quad i = 1, 2, \dots, m$$

where $e_n^i(k) \in R$ is the i th element of the vector $e_n(k)$, $\operatorname{abs}(e_n^i(k))$ is the absolute value of $e_n^i(k)$, $c \in R^+$ is a predefined threshold. The critic NN signal is given by $\hat{Q}(k) = \hat{w}_3^T(k) \phi_3(v_3^T \hat{x}(k)) = \hat{w}_3^T(k) \phi_3(\hat{x}(k))$.

The observer NN weight update is given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(\hat{z}_1(k)) (\hat{w}_1^T(k) \phi_1(\hat{z}_1(k)) + l_2 \tilde{x}_1(k))^T$$

where $l_2 \in R^{m \times m}$ is a design matrix. The action NN weight-updating law is derived as

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{s}(k)) (\hat{e}_n(k+1) - l_1 \hat{e}_n(k) + \hat{Q}(k))^T$$

and the critic NN weight-updating rule is obtained as

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(\hat{x}(k)) (\hat{Q}(k) + \alpha^{N+1} p(k) - \alpha \hat{Q}(k-1))^T$$

where $\alpha_1 \in R^+, \alpha_2 \in R^+, \alpha_3 \in R$ are adaptation gains, $0 < \alpha < 1$ is a design parameter, with N the final time.

performance. By contrast in our proof, the Lyapunov function shown in the Appendix is of the form (5.B.1), which weights the estimation errors of the observer, filtered tracking errors, the NN estimation errors for the controller, $\tilde{w}_2(k)$ and $\tilde{w}_3(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner. Here the weight-tuning schemes derived from minimizing certain quadratic objective functions are used in the Lyapunov proof.

3. The NN weight-updating rules (5.29), (5.40), and (5.48) are much simpler than in Jagannathan and Lewis (1996) since they do not include an extra term, referred to as discrete-time e-mod (Jagannathan and Lewis 1996; Jagannathan 2001), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required. As a result, the complexity of the proof as well as the computational overhead is reduced significantly without the PE condition.
4. The NN weights tuning rules (5.29), (5.40), and (5.48) are updated online in contrast to the off-line training in previous adaptive critic works. The weights of the observer, action, and critic NNs can be initialized at zero or random. This means that there is no explicit off-line learning phase needed.
5. The proposed scheme results in a well-defined controller by avoiding the problem of $\hat{g}(x(k))$ becoming zero.
6. Conditions (5.54) through (5.56) can be verified easily. For instance, the hidden layer of the critic NN consists of n_1 nodes with the hyperbolic tangent sigmoid function as its activation function, then $\|\phi_1(\cdot)\|^2 \leq n_1$. The NN learning rate α_1 can be selected as $0 < \alpha_1 < 1/n_1$ to satisfy (5.56). A similar analysis can be performed to obtain the NN learning rate α_2 .
7. There is no information currently available to decide the number of hidden-layer neurons for a multilayer NN. However, the number of hidden-layer neurons required for suitable approximation can be addressed by using the stability of the closed-loop system and the error bounds of the NNs. With regards to the error bounds of NN, in this chapter, a single-layer NN is used to approximate the continuous functions on a compact set S . According to Igelnik and Pao (1995), if the hidden-layer nodes are large enough, the reconstruction error $\varepsilon(k)$ approaches zero. If the continuous function is restricted to satisfy the

Lipschitz condition, then the reconstruction error of order $O(C/\sqrt{n})$ is achieved, where n is the number of hidden-layer nodes, and C is independent of n .

Example 5.3.1 (Output Feedback NN Control of MIMO Nonlinear Systems): The MIMO nonlinear system is described by

$$x_1(k+1) = x_3(k) \quad (5.58)$$

$$x_2(k+1) = x_4(k) \quad (5.59)$$

$$x_3(k+1) = -\frac{5}{8} \frac{x_1(k)}{(1+x_3^2(k))} + x_1(k) + \frac{1}{(10+x_1^2(k)+x_3^2(k))} u_1(k) \quad (5.60)$$

$$x_4(k+1) = -\frac{9}{16} \frac{x_2(k)}{(1+x_4^2(k))} + x_2(k) + \frac{1}{(1+x_2^2(k)+x_4^2(k))} u_2(k) \quad (5.61)$$

$$y(k) = [x_1(k), x_2(k)]^T \quad (5.62)$$

where $x_i(k) \in R$, $i = 1, \dots, 4$ is the state, $u_1(k) \in R$ and $u_2(k) \in R$ are the control inputs, and $y(k) \in R^2$ is the system output. The objective is to track a reference signal using the proposed adaptive NN output feedback controller. The reference signal used was selected as $Y_d(k) = [\sin(\omega kT + \xi), \sin(\omega kT + \xi + \pi)]^T$, $\omega = 0.1$, $\xi = \pi/2$, with a sampling interval of $T = 50$ msec. The total simulation time is taken as 200 sec. The actuator constraint is taken as 4.0 for $u_1(k)$ and $u_2(k)$. All the three NNs have ten nodes each in the hidden layer. For weight updating, the adaptation gain is selected as $\alpha_1 = \alpha_2 = \alpha_3 = 0.1$. The parameter α is taken as 0.5 and c is taken as 0.1. Both l_1 and l_2 are selected as $0.06I$, where I is a 2×2 identity matrix. All the initial hidden-layer weights and the action NN $\hat{w}_2^T(k)\phi_2(\hat{s}(k))$ output layer weights are chosen at random in the interval $[0, 1]$. The observer NN $\hat{w}_1^T(k)\phi_1(\hat{z}_1(k))$ and the critic NN $\hat{w}_3^T(k)\phi_3(\hat{x}(k))$ output layer weights are chosen as zeros. All the activation functions are selected as hyperbolic tangent sigmoid functions.

Figure 5.1 and Figure 5.2 illustrate the tracking performance of the adaptive output feedback NN controller without and with saturation, respectively. Figure 5.3 and Figure 5.4 depict the associated control inputs. Without the saturation constraint, a considerable overshoot was observed during initial transient

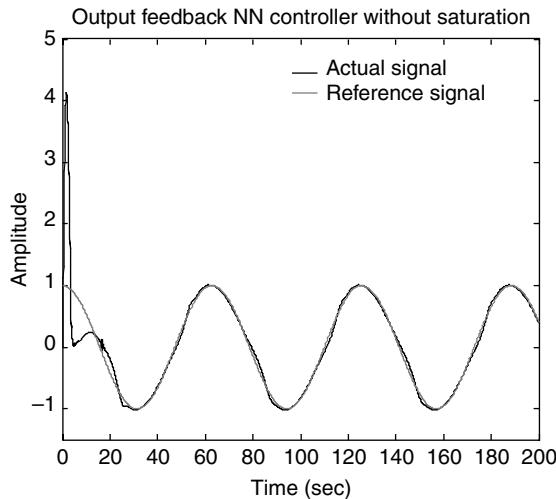


FIGURE 5.1 State $x_1(k)$ without saturation.

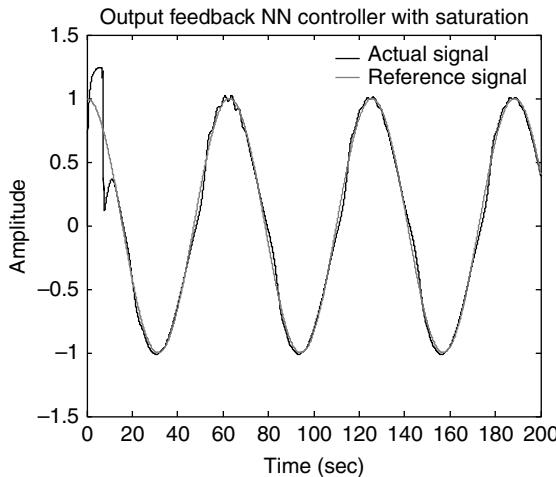


FIGURE 5.2 State $x_1(k)$ with saturation.

phase as illustrated in Figure 5.1. With the input constraints imposed, the system appears to have a better transient response. By properly selecting the upper limits of the actuators, one could avoid unexpected damage to the system without sacrificing the tracking performance.

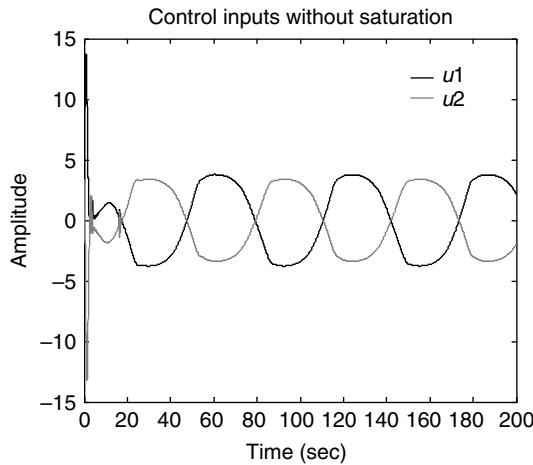


FIGURE 5.3 Control inputs without saturation.

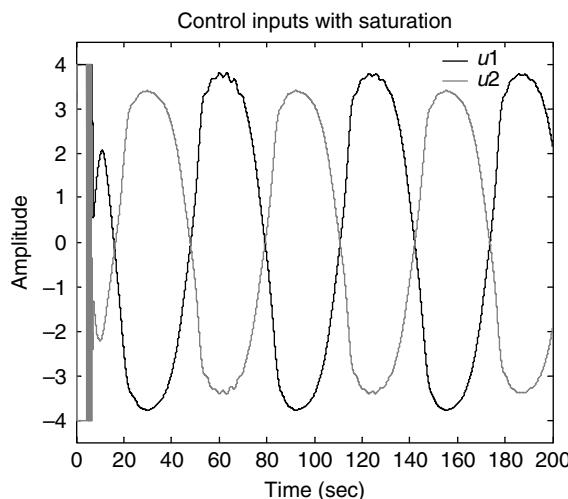


FIGURE 5.4 Control inputs with saturation.

5.4 CONCLUSIONS

A novel adaptive critic NN-based output feedback controller with magnitude constraints is designed to deliver a desired tracking performance for a class of MIMO strict feedback nonlinear discrete-time systems. The adaptive critic NN structure optimizes a certain strategic utility function, which is quite similar to

Bellman equation. Magnitude constraints on the control input allow the designer to meet the physical limits of the actuator while meeting the closed-loop stability and tracking performance requirements. The UUB of the closed-loop tracking, the estimation errors, and NN weight estimates was demonstrated.

REFERENCES

- Bertsekas, D.P. and Tsitsiklis, J.N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- Chen, F.C. and Khalil, H.K., Adaptive control of a class of nonlinear discrete-time systems using neural networks, *IEEE Trans. Automat. Contr.*, 40, 791–801, 1995.
- Ge, S.S., Lee, T.H., Li, G.Y., and Zhang, J., Adaptive NN control for a class of discrete-time nonlinear systems, *Int. J. Contr.*, 76, 334–354, 2003.
- He, P. and Jagannathan, S., Reinforcement-based neuro-output feedback control of discrete-time systems with input constraints, *IEEE Trans. Syst., Man Cybern., Part B*, 35, 150–154, 2005.
- Igelnik, B. and Pao, Y.H., Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, 6, 1320–1329, 1995.
- Jagannathan, S., Control of a class of nonlinear discrete-time systems using multi layer neural networks, *IEEE Trans. Neural Netw.*, 12, 1113–1120, 2001.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Automat. Contr.*, 41, 1693–1699, 1996.
- Krstic, M., Kanellakopoulos, I., and Kokotovic, P.V., *Nonlinear and Adaptive Control Design*, John Wiley & Sons, Inc., New York, 1995.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.
- Lewis, F.L., Jagannathan, S., and Yesilderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, UK, 1999.
- Lin, X. and Balakrishnan, S.N., Convergence analysis of adaptive critic based optimal control, *Proceedings of American Control Conference*, vol. 3, pp. 1929–1933, 2000.
- Murray, J.J., Cox, C., Lendaris, G.G., and Saeks, R., Adaptive dynamic programming, *IEEE Trans. Syst., Man, Cybern.*, 32, 140–153, 2002.
- Prokhorov, D.V. and Wunsch, D.C., Adaptive critic designs, *IEEE Trans. Neural Netw.*, 8, 997–1007, 1997.
- Si, J. and Wang, Y.T., On-line learning control by association and reinforcement, *IEEE Trans. Neural Netw.*, 12, 264–272, 2001.
- Werbos, P.J., Neurocontrol and supervised learning: an overview and evaluation, in *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 65–90, 1992.
- Yeh, P.C. and Kokotovic, P.V., Adaptive output feedback design for a class of nonlinear discrete-time systems, *IEEE Trans. Automat. Contr.*, 40, 1663–1668, 1995.

PROBLEMS

SECTION 5.6

5.6-1: The MIMO nonlinear system is described by

$$x_1(k+1) = x_3(k) \quad (5.63)$$

$$x_2(k+1) = x_4(k) \quad (5.64)$$

$$x_3(k+1) = -\frac{1}{8} \frac{x_1(k)}{(1+x_3^2(k))} + x_1(k) + \frac{1}{(10+x_1^2(k)+x_4^2(k))} u_1(k) \quad (5.65)$$

$$x_4(k+1) = -\frac{5}{16} \frac{x_2(k)}{(1+x_4^2(k))} + x_2(k) + \frac{1}{(1+x_2^2(k)+x_3^2(k))} u_2(k) \quad (5.66)$$

$$y(k) = [x_1(k), x_2(k)]^T \quad (5.67)$$

where $x_i(k) \in R$, $i = 1, \dots, 4$ is the state, $u_1(k) \in R$ and $u_2(k) \in R$ are the control inputs, and $y(k) \in R^2$ is the system output. The objective is to track a reference signal using the proposed adaptive NN output feedback controller. The reference signal used was selected as $Y_d(k) = [\cos(\omega kT + \xi), \cos(\omega kT + \xi + \pi)]^T$, $\omega = 0.1$, $\xi = \pi/2$, with a sampling interval of $T = 50$ msec. Design and simulate the output feedback controller using MATLAB®. Plot the time histories of the desired and actual states. Plot the error signals.

5.6-2: Consider the following MIMO nonlinear system described by

$$x_1(k+1) = x_3(k) \quad (5.68)$$

$$x_2(k+1) = x_4(k) \quad (5.69)$$

$$x_3(k+1) = -\frac{x_1(k)}{(1+x_3^2(k))} + x_1(k) + \frac{1}{(1+x_1^2(k)+x_3^2(k))} u_1(k) \quad (5.70)$$

$$x_4(k+1) = -\frac{x_2(k)}{(1+x_4^2(k))} + x_2(k) + u_2(k) \quad (5.71)$$

$$y(k) = [x_1(k), x_2(k)]^T \quad (5.72)$$

where $x_i(k) \in R$, $i = 1, \dots, 4$ is the state, $u_1(k) \in R$ and $u_2(k) \in R$ are the control inputs, and $y(k) \in R^2$ is the system output. The reference signal used was selected as $Y_d(k) = [\cos(\omega kT + \xi), \cos(\omega kT + \xi + \pi)]^T$, $\omega = 0.1$, $\xi = \pi/2$, with a sampling interval of $T = 10$ msec. Design and simulate the output feedback controller using MATLAB. Plot the time histories of the desired and actual states. Plot the error signals.

APPENDIX 5.A

Definition 5.A.1: Suppose $y \in R$ and $X \in R^{n \times m}$, then dy/dX is a matrix whose (i,j) element is dy/dx_{ij} , where $x_{ij} \in R$, $x_{ij} \in X$, $i = 1, \dots, n$, and $j = 1, \dots, m$.

Proof of Lemma 5.3.1: Define the matrices $A \in R^{m \times m}$, $X \in R^{n \times m}$, and vectors $b \in R^n$ and $q \in R^m$ as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} \quad (5.A.1)$$

Writing $X^T b \in R^m$ in a compact form as

$$X^T b = \begin{bmatrix} x_{11}b_1 + x_{21}b_2 + x_{31}b_3 + \cdots + x_{n1}b_n \\ x_{12}b_1 + x_{22}b_2 + x_{32}b_3 + \cdots + x_{n2}b_n \\ \vdots \\ x_{1m}b_1 + x_{2m}b_2 + x_{3m}b_3 + \cdots + x_{nm}b_n \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} \quad (5.A.2)$$

where $\beta_i \in R$, $i = 1, \dots, m$, $\beta_1 = x_{11}b_1 + x_{21}b_2 + x_{31}b_3 + \cdots + x_{n1}b_n$, and $\beta_m = x_{1m}b_1 + x_{2m}b_2 + x_{3m}b_3 + \cdots + x_{nm}b_n$.

Then $AX^T b + q$ is rewritten as

$$\begin{aligned} AX^T b + q &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mm} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} \\ &= \begin{bmatrix} a_{11}\beta_1 + a_{12}\beta_2 + \cdots + a_{1m}\beta_m + q_1 \\ a_{21}\beta_1 + a_{22}\beta_2 + \cdots + a_{2m}\beta_m + q_2 \\ \vdots \\ a_{m1}\beta_1 + a_{m2}\beta_2 + \cdots + a_{mm}\beta_m + q_m \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \end{bmatrix} \quad (5.A.3) \end{aligned}$$

where $\gamma_i \in R$, $i = 1, \dots, m$, $\gamma_1 = a_{11}\beta_1 + a_{12}\beta_2 + \cdots + a_{1m}\beta_m + q_1$, and $\gamma_m = a_{m1}\beta_1 + a_{m2}\beta_2 + \cdots + a_{mm}\beta_m + q_m$.

Then utilizing (5.A.2), we find that $(AX^T b + q)^T(AX^T b + q)$ is given by

$$(AX^T b + q)^T(AX^T b + q) = [\gamma_1, \gamma_2, \dots, \gamma_m] \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_m \end{bmatrix} = \sum_{i=1}^m \gamma_i^2 \quad (5.A.4)$$

Combining Definition 5.A.1, we rewrite the derivative of $(AX^T b + q)^T(AX^T b + q)$ with respect to the matrix X as

$$\begin{aligned} \frac{d((AX^T b + q)^T(AX^T b + q))}{dX} \\ = \begin{bmatrix} \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{11}} & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{12}} & \cdots & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{1m}} \\ \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{21}} & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{22}} & \cdots & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{n1}} & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{n2}} & \cdots & \sum_{i=1}^m \frac{d\gamma_i^2}{dx_{nm}} \end{bmatrix} \quad (5.A.5) \end{aligned}$$

Substituting (5.A.1), (5.A.2), and (5.A.4) into (5.A.5), we obtain Lemma 5.1

$$\begin{aligned}
 & \frac{d((AX^T b + q)^T (AX^T b + q))}{dX} \\
 &= 2 \begin{bmatrix} b_1 \sum_{i=1}^m \gamma_i a_{i1} & b_1 \sum_{i=1}^m \gamma_i a_{i2} & \cdots & b_1 \sum_{i=1}^m \gamma_i a_{im} \\ b_2 \sum_{i=1}^m \gamma_i a_{i1} & b_2 \sum_{i=1}^m \gamma_i a_{i2} & \cdots & b_2 \sum_{i=1}^m \gamma_i a_{im} \\ \vdots & \vdots & \vdots & \vdots \\ b_n \sum_{i=1}^m \gamma_i a_{i1} & b_n \sum_{i=1}^m \gamma_i a_{i2} & \cdots & b_n \sum_{i=1}^m \gamma_i a_{im} \end{bmatrix} \\
 &= 2 \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \begin{bmatrix} \sum_{i=1}^m \gamma_i a_{i1} & \sum_{i=1}^m \gamma_i a_{i2} & \cdots & \sum_{i=1}^m \gamma_i a_{im} \end{bmatrix} \\
 &= 2 \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} [\gamma_1 \quad \gamma_2 \quad \cdots \quad \gamma_m] A \\
 &= 2b(AX^T b + q)^T A = 2b(A^T(AX^T b + q))^T
 \end{aligned} \tag{5.A.6}$$

This completes the proof.

APPENDIX 5.B

Proof of Theorem 5.3.1:

Case I: $\|v(k)\| \leq u_{\max}$. Define the Lyapunov function as

$$\begin{aligned}
 J(k) &= \frac{\gamma_1}{2} \sum_{i=1}^n \|\tilde{x}_i(k-1)\|^2 + \frac{\gamma_2}{2} \sum_{i=1}^n \|\tilde{x}_i(k)\|^2 + \frac{\gamma_3}{3} \sum_{i=1}^n \|e_i(k)\|^2 \\
 &\quad + \frac{\gamma_4}{3} \sum_{i=1}^n \|e_n(k)\|^2 + \frac{\gamma_5}{\alpha_1} \text{tr}(\tilde{w}_1^T(k-1)\tilde{w}_1(k-1)) + \frac{\gamma_6}{\alpha_1} \text{tr}(\tilde{w}_1^T(k)\tilde{w}_1(k)) \\
 &\quad + \frac{\gamma_7}{\alpha_2} \text{tr}(\tilde{w}_2^T(k)\tilde{w}_2(k)) + \frac{\gamma_8}{\alpha_3} \text{tr}(\tilde{w}_3^T(k)\tilde{w}_3(k)) + \gamma_9 \|\xi_3(k)\|^2
 \end{aligned} \tag{5.B.1}$$

where $\gamma_i \in R^+, i = 1, \dots, 9$ are design parameters.

The first difference of Lyapunov function is given by

$$\Delta J(k) = \sum_{i=1}^9 \Delta J_i(k) \quad (5.B.2)$$

The term $\Delta J_1(k)$ is obtained using (5.11) as

$$\begin{aligned} \Delta J_1(k) &= \frac{\gamma_1}{2} \sum_{i=1}^n \|\tilde{x}_i(k)\|^2 - \frac{\gamma_1}{2} \sum_{i=1}^n \|\tilde{x}_i(k-1)\|^2 \\ &\leq \gamma_1 \|\zeta_1(k-1)\|^2 + \gamma_1 \|d_1(k-1)\|^2 - \frac{\gamma_1}{2} \|\tilde{x}_1(k-1)\|^2 \end{aligned} \quad (5.B.3)$$

Now taking the second term in the first difference (5.A.2) and substituting (5.39) into (5.B.4) to get

$$\begin{aligned} \Delta J_2(k) &= \frac{\gamma_2}{2} \sum_{i=1}^n \|\tilde{x}_i(k+1)\|^2 - \frac{\gamma_2}{2} \sum_{i=1}^n \|\tilde{x}_i(k)\|^2 \\ &\leq \gamma_2 \|\zeta_1(k)\|^2 + \gamma_2 \|d_1(k)\|^2 - \frac{\gamma_2}{2} \|\tilde{x}_1(k)\|^2 \end{aligned} \quad (5.B.4)$$

Taking the third term in (5.B.2) and substituting (5.53) into (5.B.2) and simplifying, we get

$$\begin{aligned} \Delta J_3(k) &= \frac{\gamma_3}{3} \sum_{i=1}^n \|e_i(k+1)\|^2 - \frac{\gamma_3}{3} \sum_{i=1}^n \|e_i(k)\|^2 \\ &\leq \gamma_3 l_{1\max}^2 \|e_n(k)\|^2 + \gamma_3 g_{\max}^2 \|\zeta_2(k)\|^2 + \gamma_3 \|d_2(k)\|^2 - \frac{\gamma_3}{3} \|e_1(k)\|^2 \end{aligned} \quad (5.B.5)$$

where $l_{1\max} \in R$ is the maximum eigenvalue of matrix l_1 .

Similarly, using (5.53) $\Delta J_4(k)$ is obtained as

$$\begin{aligned} \Delta J_4(k) &= \frac{\gamma_4}{3} \|e_n(k+1)\|^2 - \frac{\gamma_4}{3} \|e_n(k)\|^2 \\ &\leq \gamma_4 l_{1\max}^2 \|e_n(k)\|^2 + \gamma_4 g_{\max}^2 \|\zeta_2(k)\|^2 + \gamma_4 \|d_2(k)\|^2 - \frac{\gamma_4}{3} \|e_n(k)\|^2 \end{aligned} \quad (5.B.6)$$

Utilizing the observer NN weight-updating rule (5.57), we derive $\Delta J_5(k)$ as

$$\begin{aligned}\Delta J_5(k) \leq & -\gamma_5(1-\alpha_1\|\phi_1(k-1)\|^2)\|\zeta_1(k-1)+l_2\tilde{x}_1(k-1)+w_1^T\phi_1(k-1)\|^2 \\ & -\gamma_5\|\zeta_1(k-1)\|^2+2\gamma_5l_{2\max}^2\|\tilde{x}_1(k-1)\|^2+2\gamma_5w_{1m}^2\phi_{1m}^2\end{aligned}\quad (5.B.7)$$

where $\zeta_1(k-1) \in R^m$ is defined in (5.36), $l_{2\max} \in R$ is the maximum eigenvalue of matrix l_2 . Similarly, $\Delta J_6(k)$ is obtained using (5.57) as

$$\begin{aligned}\Delta J_6(k) \leq & -\gamma_6(1-\alpha_1\|\phi_1(k)\|^2)\|\zeta_1(k)+l_2\tilde{x}_1(k)+w_1^T\phi_1(k)\|^2-\gamma_6\|\zeta_1(k)\|^2 \\ & +2\gamma_6l_{2\max}^2\|\tilde{x}_1(k)\|^2+2\gamma_6w_{1m}^2\phi_{1m}^2\end{aligned}\quad (5.B.8)$$

Combining (5.45), (5.53) with (5.48), $\Delta J_7(k)$ is obtained as

$$\begin{aligned}\Delta J_7(k) = & \gamma_7 \left[-g_{\min}\|\zeta_2(k)\|^2 - (g_{\min} - \alpha_2\|\phi_2(k)\|^2)g_{\max}^2 \right. \\ & \times \left. \left\| \zeta_2(k) + \frac{(I - \alpha_2\phi_2^2(k)g(x(k)))\theta(k)}{g_{\min} - \alpha_2\|\phi_2(k)\|^2g_{\max}^2} \right\|^2 \right] \\ & + \gamma_7'(\|w_3\phi_3(k) - d_2(k)\|^2 + 2\|\zeta_1(k)\|^2 + 2\|d_1(k)\|^2 \\ & + 2l_{1\max}^2(\|\zeta_1(k-1)\|^2 + \|d_1(k-1)\|^2) + \|\zeta_3(k)\|^2)\end{aligned}\quad (5.B.9)$$

In addition, $\Delta J_8(k)$ can be rewritten as

$$\begin{aligned}\Delta J_8(k) = & -\gamma_8(1-\alpha_3\|\phi_3(k)\|^2)\|\zeta_3(k)+w_3^T\phi_3(k)+\alpha^{N+1}p(k)-\alpha\hat{Q}(k-1)\|^2 \\ & -\gamma_8\|\zeta_3(k)\|^2+2\gamma_8\alpha^2\|\zeta_3(k-1)\|^2 \\ & +2\gamma_8\|w_3^T(\phi_3(k)-\alpha\phi_3(k-1))+\alpha^{N+1}p(k)\|^2\end{aligned}\quad (5.B.10)$$

On the other hand, $\Delta J_9(k)$ can be expressed as

$$\Delta J_9(k) = \gamma_9\|\zeta_3(k)\|^2 - \gamma_9\|\zeta_3(k-1)\|^2\quad (5.B.11)$$

Taking $\gamma_9 = 2\gamma_8\alpha^2$, and combining (5.B.3) through (5.B.11) to define the first difference of the Lyapunov function, $\Delta J(k)$, as

$$\begin{aligned}
 \Delta J(k) = & -\frac{1}{2}(\gamma_1 - 4\gamma_5 l_{2\max}^2) \|\tilde{x}_1(k-1)\|^2 - \frac{1}{2}(\gamma_2 - 4\gamma_6 l_{2\max}^2) \|\tilde{x}_1(k)\|^2 \\
 & - \frac{\gamma_3}{3} \|e_1(k)\|^2 - \frac{1}{3}(\gamma_4 - 3(\gamma_3 + \gamma_4)l_{1\max}^2) \|e_n(k)\|^2 - \gamma_6(1 - \alpha_1 \|\phi_1(k)\|^2) \\
 & \times \|\zeta_1(k) + l_2 \tilde{x}_1(k) + w_1^T \phi_1(k)\|^2 - (\gamma_6 - \gamma_2 - 2\gamma'_7) \|\zeta_1(k)\|^2 \\
 & - \gamma_5(1 - \alpha_1 \|\phi_1(k-1)\|^2) \|\zeta_1(k-1) + l_2 \tilde{x}_1(k-1) + w_1^T \phi_1(k-1)\|^2 \\
 & - (\gamma_5 - \gamma_1 - 2\gamma'_7 l_{2\max}^2) \|\zeta_1(k-1)\|^2 - \gamma_7(g_{\min} - \alpha_2 \|\phi_2(k)\|^2 g_{\max}^2) \\
 & \times \left\| \zeta_2(k) + \frac{(I - \alpha_2 \|\phi_2(k)\|^2 g(x(k)) \beta(k))}{g_{\min} - \alpha_2 \|\phi_2(k)\|^2 g_{\max}^2} \right\|^2 - (\gamma_7 g_{\min} - \gamma_3 g_{\max}^2 \\
 & - \gamma_4 g_{\max}^2) \|\zeta_2(k)\|^2 - \gamma_8(1 - \alpha_3 \|\phi_3(k)\|^2) \|\zeta_3(k) + w_3^T \phi_3(k)\|^2 \\
 & + \alpha^{N+1} p(k) - \alpha \hat{Q}(k-1) \|^2 - (\gamma_8 - 2\gamma_8\alpha^2 - \gamma'_7) \|\zeta_3(k)\|^2 + D_M^2
 \end{aligned} \tag{5.B.12}$$

where

$$\begin{aligned}
 D_M^2 = & (\gamma_1 + \gamma_2 + 2(1 + l_{2\max}^2)\gamma'_7) d_{1m}^2 + (\gamma_3 + \gamma_4 + 2\gamma'_7) d_{2m}^2 + 2\gamma_5 w_{1m}^2 \phi_{1m}^2 \\
 & + 2\gamma_6 w_{1m}^2 \phi_{1m}^2 + 6\gamma_8 + 2(\gamma'_7 + 3\gamma_8(1 + \alpha^2)) w_{3m}^2 \phi_{3m}^2
 \end{aligned} \tag{5.B.13}$$

This implies that $\Delta J(k) \leq 0$ as long as (5.54) through (5.57) are satisfied and the following conditions hold:

$$\|\tilde{x}_1(k)\| \geq \frac{\sqrt{2}D_M}{\sqrt{\gamma_2 - 4\gamma_6 l_{2\max}^2}} \tag{5.B.14}$$

or

$$\|e_n(k)\| \geq \frac{\sqrt{3}D_M}{\sqrt{\gamma_4 - 3(\gamma_3 + \gamma_4)l_{1\max}^2}} \tag{5.B.15}$$

or

$$\|\zeta_1(k)\| \geq \frac{D_M}{\sqrt{\gamma_6 - \gamma_2 - 2\gamma'_7}} \tag{5.B.16}$$

or

$$\|\zeta_2(k)\| \geq \frac{D_M}{\sqrt{\gamma_7 g_{\min} - \gamma_3 g_{\max}^2 - \gamma_4 g_{\max}^2}} \quad (5.B.17)$$

or

$$\|\zeta_3(k)\| \geq \frac{D_M}{\sqrt{\gamma_8 - 2\gamma_8\alpha^2 - \gamma_7'}} \quad (5.B.18)$$

Case II: $\|v(k)\| > u_{\max}$

The proof is similar to that in *Case I* and it is omitted.

For both Case I and Case II, $\Delta J(k) \leq 0$ for all k is greater than zero. According to the standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that $\tilde{x}_1(k)$, $e_1(k)$ and the weight estimation errors are UUB. The boundedness of $\|\zeta_1(k)\|$, $\|\zeta_2(k)\|$, and $\|\zeta_3(k)\|$ implies that $\|\tilde{w}_1(k)\|$, $\|\tilde{w}_2(k)\|$, and $\|\tilde{w}_3(k)\|$ and weight estimates $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded. Since $\tilde{x}_1(k)$ is bounded, from the estimation errors system given by (5.39), it implies that all the estimation errors are bounded. Similarly, bounded $e_n(k)$ implies that all the tracking errors are bounded from (5.25) and (5.28). Therefore all the signals in the observer-controller system are bounded.

6 Neural Network Control of Nonstrict Feedback Nonlinear Systems

In Chapter 5, the adaptive neural network (NN) control of a class of strict feedback nonlinear discrete-time systems was presented using NN. Although Lyapunov stability analysis was discussed in detail, the analysis was limited to a restrictive class of nonlinear systems in strict feedback form. The dynamics of several industrial systems, for example, spark ignition (SI) engine (Daw et al. 1997) during lean operation and with high exhaust gas recirculation (EGR), can be represented only in nonstrict feedback form. Moreover, the controller design presented in Chapter 5 for strict feedback nonlinear systems is not applicable to nonstrict feedback nonlinear discrete-time systems. Therefore, in this chapter, we initially treat the design of the controller by assuming that the states of the nonlinear discrete-time systems in nonstrict feedback form are available for measurement (He and Jagannathan 2003a) and later relax this assumption (He and Jagannathan 2004) by using an NN observer.

6.1 INTRODUCTION

In this section, we describe the class of systems to be dealt with in this chapter and the backstepping methodology that is used to arrive at the controller.

6.1.1 NONLINEAR DISCRETE-TIME SYSTEMS IN NONSTRICT FEEDBACK FORM

Consider the following nonstrict feedback nonlinear system to be controlled, described by

$$\begin{aligned}x_1(k+1) &= f_1(x_1(k), x_2(k)) + g_1(x_1(k), x_2(k))x_2(k) + d'_1(k) \\x_2(k+1) &= f_2(x_1(k), x_2(k)) + g_2(x_1(k), x_2(k))u(k) + d'_2(k)\end{aligned}\tag{6.1}$$

where $x_i(k) \in R$; $i = 1, 2$ are states, $u(k) \in R$ is the system input, and $d'_1(k) \in R$ and $d'_2(k) \in R$ are unknown but bounded disturbances whose bounds are given by $\|d'_1(k)\| \leq d_{1M}$ and $\|d'_2(k)\| \leq d_{2M}$. The nonlinear discrete-time systems in (6.1) in nonstrict feedback form have a peculiar characteristic because the nonlinear functions, $f_1(\cdot)$ and $g_1(\cdot)$, are a function of both the system states, $x_1(k)$ and $x_2(k)$, unlike in the case of strict feedback nonlinear system, where $f_1(\cdot)$ and $g_1(\cdot)$ are only a function of the state $x_1(k)$. Moreover, in this chapter, $f_i(k)$ and $g_i(k)$; $i = 1, 2$; are considered as unknown smooth functions for the controller design.

Note that there is no general approach to analyze this class of nonlinear systems. Further, an adaptive backstepping design (He and Jagannathan 2003a) from Chapter 5 cannot be applied directly to (6.1) since it results in a noncausal closed-loop system. Moreover, adaptive backstepping control, for instance, needs an additional linear in the unknown parameter (LIP) assumption (Kanellakopoulos 1991; Kokotovic 1992; Yeh and Kokotovic 1995) and the nonlinear functions, $f_i(\cdot)$ and $g_i(\cdot)$; $i = 1, 2$, normally may not be satisfied. Therefore, in Chen and Khalil (1995), a one-step predictor was used to convert the strict feedback nonlinear system into a causal system and then a controller design was discussed. However, for nonstrict feedback nonlinear systems (6.1), a one-step predictor when directly applied to an n th order system does not work since $f_1(\cdot)$ and $g_1(\cdot)$ are functions of both $x_1(k)$ and $x_2(k)$. This problem is confronted in He and Jagannathan (2003a) by using the well-known NN approximation property (Barron 1991) with some mild assumptions as explained in Section 6.2.1 since an NN acts as a one-step nonlinear predictor for the second-order nonlinear discrete-time systems in nonstrict feedback form.

For simplicity, let us denote $f_i(k)$ for $f_i(x_1(k), x_2(k))$, $g_i(k)$ for $g_i(x_1(k), x_2(k))$, $\forall i = 1, 2$. The system under consideration (6.1) can be rewritten as

$$\begin{aligned} x_1(k+1) &= f_1(k) + g_1(k)x_2(k) + d'_1(k) \\ x_2(k+1) &= f_2(k) + g_2(k)u(k) + d'_2(k) \end{aligned} \tag{6.2}$$

Our objective is to design an NN controller for the nonlinear system (6.1) such that (1) all the signals in the closed-loop remain uniformly ultimately bounded (UUB) while (2) the state $x_1(k)$ follows a desired trajectory $x_{1d}(k)$. To meet the objective, a suite of adaptive NN controllers will be presented in Section 6.2 by using the tracking error and more complex adaptive critic NN control architectures under the assumption that all the system states are available for measurement. Finally in Section 6.3, an output feedback design is covered by assuming that the state $x_2(k)$ is unavailable. In both state and output feedback controller designs the backstepping design

methodology from Kokotovic (1992) is utilized and therefore it is discussed briefly next.

6.1.2 BACKSTEPPING DESIGN

The backstepping methodology is a potential solution for controlling a larger class of nonlinear systems. By using NNs in each stage of the backstepping procedure to estimate certain nonlinear functions (Jagannathan 2001), a more suitable control law can be designed without using the LIP assumption and the need for a regression matrix. Recently, the adaptive NN control of nonlinear systems using backstepping approach in both continuous (Kuljaca et al. 2003) and discrete-time (Jagannathan 2001) has been dealt with in several works.

The discrete-time backstepping-based NN control design (Jagannathan 2001) is far more complex than continuous-time due primarily to the fact that discrete-time Lyapunov derivatives are quadratic in the state and not linear as in the continuous-time case. In Jagannathan (2001), a multilayer NN backstepping controller is proposed for discrete-time feedback system in strict feedback form, where $f_i(\cdot)$, $i = 1, \dots, n$ are considered as unknown smooth functions whereas $g_i(\cdot)$, $i = 1, \dots, n$ are assumed to be unknown constants. By contrast, in this chapter both $f_i(\cdot)$, $i = 1, \dots, n$ and $g_i(\cdot)$, $i = 1, \dots, n$ are considered unknown.

The backstepping approach (Kokotovic 1992) is widely used to control strict feedback nonlinear systems (He and Jagannathan, 2003b, 2003c). Tracking error is used to tune NN weights with online learning. No performance measure is used in the controller design. We also use an adaptive NN backstepping approach for developing our controller. For details on standard backstepping, refer to Kokotovic (1992). The NN controller design presented in this chapter overcomes several deficiencies that currently exist in the previous works such as (1) the need to know the signs of certain unknown nonlinear functions is relaxed; (2) a well-defined controller is presented by using a single NN (Lewis et al. 1999); and (3) the NN weight-tuning rules were simplified in this work in comparison to that of Jagannathan and Lewis (1996).

An adaptive critic-based NN control scheme with reinforcement learning capability would be very useful for complex nonlinear discrete-time systems (6.1). However, adaptive critic NN control architecture using reinforcement learning is more complex than traditional tracking error based with online learning type control architectures in terms of computational overhead, though the system performance can be optimized by using the critic. In this chapter, both unsupervised learning-based tracking error and reinforcement learning-based adaptive critic NN control designs are covered. Though originally the adaptive critic NN architecture is primarily proposed when reinforcement learning is utilized to train NN, later the NN architecture is utilized for generating nearly

optimal control inputs. Such adaptive critic NN control schemes are developed in Chapter 4 and Chapter 5. By contrast, in this chapter, reinforcement learning is employed to tune the NN weights whereas optimization using the Bellman equation is not carried out. Such development is still valuable since past values of tracking error are used to tune the action NN weights instead of using tracking error from the current step as is typically seen in other nonadaptive critic NN schemes.

Two feedforward NNs with online learning capability are used to approximate the dynamics of certain nonlinear functions of (6.1) and their weights are tuned online by using tracking error information in the case of tracking error-based control schemes. On the other hand, in the case of an adaptive critic NN control architecture, two action-generating NNs approximate the dynamics of the nonlinear system (6.1) and their weights are tuned based on a signal from a third critic NN. The critic NN uses the weighted tracking error signal or a signal from a standard quadratic performance criterion, as a utility function and generates a suitable output. Note that in the controller design (He et al. 2005), a single critic NN signal is used to tune the weights of two action-generating NNs, unlike in the literature where a single critic NN is used for an action-generating NN (Werbos 1991, 1992). Feedforward NNs are normally used as building blocks in both the NN control architectures.

Lyapunov-based analysis is used to derive the novel NN weight updates. No preliminary learning phase is needed since the NN weights are tuned online and an underlying conventional controller, for instance, a proportional or a proportional plus derivative (PD) is used to maintain the stability of the closed-loop system until the NN learns. The UUB of the closed-loop tracking error is demonstrated no matter which NN control architecture is employed. A comparison in terms of their online learning and computational overhead is highlighted in Section 6.2.3. Finally, the need for the availability of all the states of the nonlinear discrete-time system is relaxed in Section 6.3 by using an NN observer.

6.2 ADAPTIVE NN CONTROL DESIGN USING STATE MEASUREMENTS

In this section, two-layer NNs are used to approximate certain nonlinear functions. The input to the hidden-layer weights for all the NNs will be selected at random and held constant throughout the simulation, whereas the hidden-to-output layer weights will be tuned online in order for a two-layer NN to approximate the nonlinear function (Igelnik and Pao 1995). First, we present a tracking error-based adaptive NN controller with an online weight-tuning scheme. Subsequently, an adaptive critic NN-based control

design with reinforcement learning scheme is introduced. Lyapunov-based analysis is performed for both control schemes. To proceed, the following mild assumptions are required.

6.2.1 TRACKING ERROR-BASED ADAPTIVE NN CONTROLLER DESIGN

Assumption 6.2.1: The desired trajectory $x_{1d}(k)$ is a smooth function and it is bounded.

Assumption 6.2.2: The unknown smooth functions, $g_i(k)$, $\forall i = 1, 2$ are bounded above within a certain compact set such that $g_{1M} > |g_1(k)| > 0$ and $g_{2M} > |g_2(k)| > 0$ holds.

Next, the adaptive backstepping NN control design is described in a step-by-step manner.

6.2.1.1 Adaptive NN Backstepping Controller Design

Step 1: Virtual controller design

Define the tracking error between actual and desired trajectory as

$$e_1(k) = x_1(k) - x_{1d}(k) \quad (6.3)$$

where $x_{1d}(k)$ is the desired trajectory. Hence, (6.3) can be rewritten as

$$\begin{aligned} e_1(k+1) &= x_1(k+1) - x_{1d}(k+1) = f_1(k) + g_1(k)x_2(k) \\ &\quad - x_{1d}(k+1) + d'_1(k) \end{aligned} \quad (6.4)$$

By viewing $x_2(k)$ as a virtual control input, a desired feedback control signal can be designed as

$$x_{2d}(k) = \frac{1}{g_1(k)}(-f_1(k) + x_{1d}(k+1)) + l_1 e_1(k) \quad (6.5)$$

where l_1 is a design constant selected, such that the tracking error, $e_1(k)$, is asymptotically stable (AS). The term, $l_1 e_1(k)$, can be viewed as a proportional controller in the outer loop. A PD controller results by adding a delayed value of $e_1(k)$ in (6.5).

Since $f_1(k)$ and $g_1(k)$ are unknown smooth functions, the desired feedback control $x_{2d}(k)$ cannot be implemented in practice. From (6.5), it can be seen that the unknown part $(1/g_1(k))(-f_1(k) + x_{1d}(k+1))$ is a smooth function

of $x_1(k)$, $x_2(k)$, and $x_{1d}(k+1)$. Therefore, a single NN (Ge et al. 2001) can be employed to approximate the entire unknown part, which consists of two nonlinear functions thus saving computational complexity. By contrast, in the past literature (Lewis et al. 1999), it is common to use a two-layer NN to approximate each of the nonlinear functions, $f_1(k)$ and $g_1(k)$. Then, one has to ensure that the NN estimate for $g_1(k)$ is bounded away from zero in order to design a well-defined controller. This problem is overcome by utilizing a single NN to approximate this unknown part, $x_{2d}(k)$, which can be expressed as

$$x_{2d}(k) = w_1^T(k)\phi(v_1^T z_1(k)) + \varepsilon_1(k) + l_1 e_1(k) = w_1^T(k)\phi(k) + \varepsilon_1(k) + l_1 e_1(k) \quad (6.6)$$

where $w_1(k) \in R^{n_1 \times 1}$ denotes the constant ideal weights, $v_1 \in R^{3 \times n_1}$ is the weights of the hidden layer, n_1 is the node's number of hidden-layer, $\phi(k)$ is the hidden-layer activation function, and $\varepsilon_1(k)$ is the approximation error. The NN input is selected as $z_1(k) = [x_1(k), x_2(k), x_{1d}(k+1)]^T$ for suitable approximation. Here we consider that the input $z_1(k)$ is restricted to a compact set and in that case the upper bound on the approximation error is a constant, which is given by $|\varepsilon_1(k)| \leq \varepsilon_{1N}$.

The virtual control input is given as

$$\hat{x}_{2d}(k) = \hat{w}_1^T(k)\phi(k) + l_1 e_1(k) \quad (6.7)$$

where $\hat{w}_1(k) \in R^{n_1 \times 1}$ is the actual NN weight vector, which needs to be tuned. Define the error in weights during estimation by

$$\tilde{w}_1(k) = \hat{w}_1(k) - w_1(k) \quad (6.8)$$

Define the error between $x_2(k)$ and $\hat{x}_{2d}(k)$ as

$$e_2(k) = x_2(k) - \hat{x}_{2d}(k) \quad (6.9)$$

Equation 6.6 can be expressed using (6.9) for $x_2(k)$ as

$$e_1(k+1) = f_1(k) + g_1(k)(e_2(k) + \hat{x}_{2d}(k)) - x_{1d}(k+1) + d'_1(k) \quad (6.10)$$

or equivalently

$$e_1(k+1) = g_1(k)(l_1 e_1(k) + e_2(k) + \xi_1(k) + d_1(k)) \quad (6.11)$$

where

$$\xi_1(k) = \tilde{w}_1^T(k)\phi(k) \quad (6.12)$$

$$d_1(k) = \frac{d'_1(k)}{g_1(k)} - \varepsilon_1(k) \quad (6.13)$$

Note that $d_1(k)$ is bounded above given the fact that $\varepsilon_1(k)$, $d'_1(k)$, and $g_1(k)$ are all bounded.

Step 2: Design of the control input $u(k)$

Writing the error $e_2(k)$ from (6.9) as

$$\begin{aligned} e_2(k+1) &= x_2(k+1) - \hat{x}_{2d}(k+1) = f_2(k) + g_2(k)u(k) \\ &\quad - \hat{x}_{2d}(k+1) + d'_2(k) \end{aligned} \quad (6.14)$$

where $\hat{x}_{2d}(k+1)$ is the future value of $\hat{x}_{2d}(k)$. From (6.7), we could assume that $\hat{x}_{2d}(k+1)$ can be obtained as a nonlinear function of $z_2(k)$, where $z_2(k) = [x_1(k), x_2(k), e_1(k), e_2(k), \hat{w}_1(k)]^T$. In other words, $\hat{x}_{2d}(k+1) = f(z_2(k))$ where $f(\cdot) : R^{5 \times 1} \rightarrow R$ is a smooth and nonlinear mapping. This is considered as a change in variables. However, a lot of work has been done on one-step predictor schemes in the literature and they are too numerous to mention. A dynamical NN can be used to predict a state of the system one step ahead. Consequently, in this chapter, $\hat{x}_{2d}(k+1)$ can be approximated by using a second NN (dynamical NN as it will be shown based on the weight updates) since the input vector, $z_2(k)$ is restricted to a compact set. In fact, Lyapunov analysis demonstrates that once the inputs to the NN (or states of the nonlinear system) are on the compact set they will continue to stay within the compact set.

Alternatively, the value of $\hat{x}_{2d}(k+1)$ can be obtained by employing a filter (Campos et al. 2000, Lewis et al. 2000). Choosing the desired control input by using the second NN to approximate the unknown dynamics as

$$\begin{aligned} u_d(k) &= \frac{1}{g_2(k)}(-f_2(k) + \hat{x}_{2d}(k+1)) + l_2e_2(k) + l_1e_1(k) \\ &= w_2^T(k)\sigma(v_2^T z_2(k)) + \varepsilon_2(k) + l_2e_2(k) + l_1e_1(k) \\ &= w_2^T(k)\sigma(k) + \varepsilon_2(k) + l_2e_2(k) + l_1e_1(k) \end{aligned} \quad (6.15)$$

where $w_2(k) \in R^{n_2 \times 1}$ is the matrix of target weights of the hidden to the output layer, $v_2 \in R^{5 \times n_2}$ is the weight matrix of the input to the hidden layer, n_2 is the number of nodes in the hidden layer, $\sigma(k)$ is the vector of activation functions,

$\varepsilon_2(k)$ is the approximation error bounded above such that $|\varepsilon_2(k)| \leq \varepsilon_{2N}$, and $l_2 \in R$ is a design constant, with the NN input given by $z_2(k)$. The input to the hidden-layer weights will not be tuned after they are selected initially at random to form a basis (Igelnik and Pao 1995). The actual control input, $u(k)$ is now selected by using the second NN as

$$u(k) = \hat{w}_2^T(k)\sigma(k) + l_2e_2(k) + l_1e_1(k) \quad (6.16)$$

where $\hat{w}_2(k) \in R^{n_2 \times 1}$ is the actual weight matrix for the second NN. Carefully observing (6.16), it is clear that the actual control input consists of a proportional controller outer loop and an NN inner loop. Adding delayed values of $e_1(k)$ and $e_2(k)$ in (6.16) will render a PD controller outer loop. Though the analysis carried out hereafter uses a proportional controller in the outer loop and the same analysis can be done with a PD controller without changing the design procedure.

Substituting (6.15) and (6.16) into (6.14) yields

$$e_2(k+1) = g_2(k)(l_1e_1(k) + l_2e_2(k) + \xi_2(k) + d_2(k)) \quad (6.17)$$

where

$$\xi_2(k) = \tilde{w}_2^T(k)\sigma(k) \quad (6.18)$$

and

$$d_2(k) = \frac{d'_2(k)}{g_2(k)} - \varepsilon_2(k) \quad (6.19)$$

Equation 6.11 and Equation 6.17 represent the closed-loop error dynamics for the nonlinear system (6.1). The next step is to design the tracking error and adaptive critic-based NN controllers.

6.2.1.2 Weight Updates

In the tracking error-based NN controller, two feedforward NN are employed to approximate the nonlinear dynamics and their weights are tuned online using the tracking errors. It is required to show that the errors, $e_1(k)$ and $e_2(k)$ and the NN weights, $\hat{w}_1(k)$ and $\hat{w}_2(k)$, are bounded. To accomplish this, first, we present an assumption on the target weights and activation functions. Second, a discrete-time NN weight-tuning algorithm, given in Table 6.1, is introduced so that closed-loop stability is inferred.

TABLE 6.1
Discrete-Time Two-layer NN Controller Using
Tracking Error Notion

The virtual control input is

$$\hat{x}_{2d}(k) = \hat{w}_1^T(k)\phi(k) + l_1 e_1(k)$$

The control input is given by

$$u(k) = \hat{w}_2^T(k)\sigma(k) + l_2 e_2(k) + l_1 e_1(k)$$

The NN weight tuning for the first NN is given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi(k)(\hat{w}_1^T(k)\phi(k) + l_1 e_1(k))$$

with the second NN weight tuning to be provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k)(\hat{w}_2^T(k)\sigma(k) + l_2 e_2(k))$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $l_1 \in R$, and $l_2 \in R$ are design parameters.

Assumption 6.2.3: The target weights and the activation functions for all the NNs are bounded above by known positive values so that

$$\|w_1(k)\| \leq w_{1\max} \quad \|w_2(k)\| \leq w_{2\max} \quad \|\phi(\cdot)\| \leq \phi_{\max} \quad \text{and} \quad \|\sigma(\cdot)\| \leq \sigma_{\max} \quad (6.20)$$

Theorem 6.2.1 (*Discrete-Time NN Controller for Nonstrict Feedback System*): Consider the system given in (6.1) and let Assumptions 6.2.1 through 6.2.3 hold. Let the disturbances and NN approximation errors be bounded above by known constants defined by d_{1N} , d_{2N} , ε_{1N} , and ε_{2N} , respectively. Let the first NN weight tuning be given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi(k)(\hat{w}_1^T(k)\phi(k) + l_1 e_1(k)) \quad (6.21)$$

with the second NN weight being tuned by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k)(\hat{w}_2^T(k)\sigma(k) + l_2 e_2(k)) \quad (6.22)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $l_1 \in R$, and $l_2 \in R$ are design parameters. Let the virtual and actual control inputs be defined by (6.7) and (6.16), respectively. The tracking errors, $e_1(k)$ (6.3), and $e_2(k)$ (6.9), the NN weights estimates, $\hat{w}_1(k)$ and $\hat{w}_2(k)$ are UUB, with the bounds specifically given by (6.A.8) through (6.A.11)

provided the design parameters are selected as:

$$1. \quad 0 < \alpha_1 \|\phi(\cdot)\|^2 < 1 \quad (6.23)$$

$$2. \quad 0 < \alpha_2 \|\sigma(\cdot)\|^2 < l_2 \quad (6.24)$$

$$3. \quad 0 < |l_1| < \frac{1}{2g_{1M}\sqrt{5 + \frac{1}{l_2}}} \quad (6.25)$$

$$4. \quad 0 < l_2 < \frac{-g_{2M} + \sqrt{(5 + g_{2M}^2)}}{10g_{2M}} \quad (6.26)$$

Proof: See Appendix 6.A at the end of this chapter.

Remarks:

1. The proposed scheme renders a well-defined controller by overcoming the problem of $\hat{g}_i(k)$, $\forall i = 1, 2$ becoming zero since a single NN is used to approximate the nonlinear functions, $f(\cdot)$ and $g(\cdot)$, in (6.5) and (6.15) in comparison with Lewis et al. (1999).
2. It is important to note that in this theorem there is no persistency of excitation (PE) condition, Certainty equivalence (CE), and LIP assumptions for the NN controller, in contrast to standard work in discrete-time adaptive control (Astrom and Wittenmark 1995). In the latter, a parameter identifier is first designed and the parameter estimation errors are shown to converge to small values by using a Lyapunov function. Then in the tracking proof, it is assumed that the parameter estimates are exact by invoking a CE assumption and another Lyapunov function is selected that weights only the tracking error terms to demonstrate the closed-loop stability and tracking performance. By contrast in our proof, the Lyapunov function shown in Appendix 6.A is of the form

$$J(k) = \frac{e_1^2(k)}{8g_{1M}^2} + \frac{e_2^2(k)}{8l_2g_{2M}^2} + \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) + \frac{1}{\alpha_2} \tilde{w}_2^T(k) \tilde{w}_2(k)$$

which weighs the tracking errors, $e_1^2(k)$ and $e_2^2(k)$ and the NN estimation errors for the controller, $\tilde{w}_1(k)$ and $\tilde{w}_2(k)$. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

3. The NN weight updating rules (6.21) and (6.22) are much simpler than in Lewis et al. (1999) and Jagannathan (2001) since (6.21) and (6.22) do not include an extra term, referred to as discrete-time ε -modification (Jagannathan and Lewis 1996), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required. As a result, the complexity of the proof as well as the computational overhead is reduced significantly without the PE condition.
4. The NN weights are tuned online using tracking error information and no performance criterion is used.
5. Noncausal problem in the discrete-time backstepping design is overcome in this work via a one-step nonlinear predictor based on NN approximation.
6. The NN weights may be initialized at zero and stability will be maintained by the outer-loop conventional controller until the NNs learn. This means that there is no off-line learning phase needed.
7. Though two-layer NNs are used to approximate certain nonlinear dynamics, instead, any function approximator such as B -splines, radial basis functions (RBFs), fuzzy basis functions can be employed as well without changing the design.
8. Conditions (6.23) and (6.24) can be checked easily. For instance, suppose that the hidden layer of the first NN consists of n_1 nodes with the radial basis, the hyperbolic tangent sigmoid, or the log sigmoid transfer function as the activation function, then $\|\varphi(\cdot)\|^2 \leq n_1$. The α_1 can be selected as $0 < \alpha_1 < (1/n_1)$ to satisfy (6.23). Similar analysis can be performed for (6.24).
9. The controller gains l_1 and l_2 have to satisfy (6.25) and (6.26) in order for the closed-loop system to be stable. Given the bounds g_{1m} and g_{2m} , a suitable value of l_1 and l_2 can be selected easily so that the closed-loop poles will remain within the unit circle. On the other hand, such constraints do not exist for controller design parameters and NN learning rates or adaptation gains in continuous time. Consequently, the design parameters for the NN controllers in continuous time are selected arbitrarily.

6.2.2 ADAPTIVE CRITIC-BASED NN CONTROLLER DESIGN

In the previous section, an adaptive NN controller was designed by using tracking error information. In this section, an adaptive critic NN controller will be designed for the nonlinear discrete-time system (6.1). In the case of adaptive

critic-based NN controller, in addition to the two action-generating NNs that are used to approximate the nonlinear system dynamics, a critic NN is introduced to tune the weights of the action-generating NNs. Two-layer feedforward NNs are used for the two action-generating and critic NNs. To proceed further, first, we design the critic NN and then its online weight-tuning scheme is proposed. The design of the two action-generating NNs is quite similar to the design procedure in Section 6.2.1. Note that a single critic NN signal is used to tune two action-generating NNs in contrast to the standard adaptive critic NN architecture (Barto 1992; Werbos 1992) where each action-generating NN is tuned by a separate critic NN signal. This approach is similar to supervised actor-critic reinforcement learning methodology (Rosenstein and Barto 2004) where a supervised signal is provided to both action and critic NN.

6.2.2.1 Critic NN Design

A choice of the critic signal is given by

$$R(k) = \hat{w}_3^T(k)\varphi(v_3^T z_3(k)) = \hat{w}_3^T(k)\varphi(k) \quad (6.27)$$

where $\hat{w}_3(k) \in R^{n_3 \times 1}$ represent the matrix of actual weights of the hidden to the output layer, $v_3 \in R^{3 \times n_3}$ is the weight matrix of the input to the hidden layer, which is selected initially at random and held constant thereafter (Igelnik and Pao 1995), n_3 is the number of nodes in the hidden layer, and $\varphi(k)$ is the hidden-layer activation function. The critic NN inputs are given by $z_3(k) = [e_1^T(k)Pe_1(k), e_2^T(k)Qe_2(k), u(k)^TGu(k)]^T$, where the quadratic terms are viewed as the real-valued instantaneous utility function of the plant performance with P , Q , and G as design matrices. When the tracking errors, $e_1(k)$, $e_2(k)$, and the input, $u(k)$, are small, system performance is good. Though this utility function is quite simple, the adaptive critic architecture allows complex utility functions. A more standard utility function is typically quadratic and a scalar function of tracking errors, $e_1(k)$ and $e_2(k)$, given by

$$z_3(k) = e_1^T(k)Pe_1(k) + e_2^T(k)Qe_2(k) \quad (6.28)$$

A quadratic term in the control input can be added as well similar to the case in standard optimal control. The structure of the proposed adaptive critic NN controller is shown in Figure 6.1 and the controller derived is given in Table 6.2.

In the figure, the critic NN, $\hat{w}_3^T(k)\varphi(k)$ generates the critic signal, $R(k)$ which in turn is used to tune the weights of action NN1, $\hat{w}_1^T(k)\phi(k)$ and action NN2, $\hat{w}_2^T(k)\sigma(k)$. The action NN1 is employed to deliver the virtual control signal, $\hat{x}_{2d}(k)$ and the action NN2 is utilized to select the actual control input $u(k)$. The design parameters, $A \in R$, $B \in R$, $l_1 \in R$, $l_2 \in R$, P , Q , and G

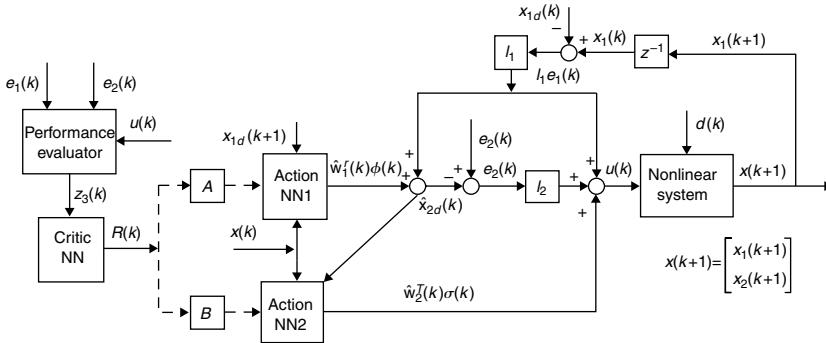


FIGURE 6.1 Adaptive critic NN-based controller structure.

in the figure are selected by the designer as explained in the Theorem 6.2.2. In order to prove the stability of the closed-loop system, the following analysis is needed.

6.2.2.2 Weight-Tuning Algorithms

It is required to show that the errors, $e_1(k)$ and $e_2(k)$, and the NN weight matrices, $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded. First, we present bounds on the ideal weights and activation functions of critic NN. Second, the action-generating and critic NN weight tuning are given so that closed-loop stability is inferred.

Assumption 6.2.4: The critic NN ideal weights and activation functions are bounded by known positive values so that

$$\|w_3\| \leq w_{3\max} \quad \|\varphi(\cdot)\| \leq \varphi_{\max} \quad (6.29)$$

Theorem 6.2.2 (Adaptive Critic NN Controller for Nonstrict Feedback Nonlinear Systems): Consider the hypothesis presented in Theorem 6.2.1 and let Assumptions 6.2.1 through 6.2.4 hold. Let the first action-generating NN weight tuning, NN1 be given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi(k) (\hat{w}_1^T(k) \phi(k) + l_1 e_1(k) + AR(k)) \quad (6.30)$$

and the second action-generating NN weight tuning, NN2, be provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k) (\hat{w}_2^T(k) \sigma(k) + l_2 e_2(k) + BR(k)) \quad (6.31)$$

TABLE 6.2
Discrete-Time Adaptive Critic NN Controller

The virtual control input is

$$\hat{x}_{2d}(k) = \hat{w}_1^T(k)\phi(k) + l_1 e_1(k)$$

The control input is given by

$$u(k) = \hat{w}_2^T(k)\sigma(k) + l_2 e_2(k) + l_1 e_1(k)$$

The performance criterion used as input to the critic NN is given by

$$z_3(k) = [e_1^T(k)Pe_1(k), e_2^T(k)Qe_2(k), u^T(k)Gu(k)]^T$$

or

$$z_3(k) = e_1^T(k)Pe_1(k) + e_2^T(k)Qe_2(k)$$

The critic NN output is

$$R(k) = \hat{w}_3^T(k)\varphi(v_3^T z_3(k)) = \hat{w}_3^T(k)\varphi(k)$$

The NN weight-tuning for the first action-generating NN is given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi(k)(\hat{w}_1^T(k)\phi(k) + l_1 e_1(k) + AR(k))$$

and let the second action-generating NN weight-tuning be provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k)(\hat{w}_2^T(k)\sigma(k) + l_2 e_2(k) + BR(k))$$

with the critic NN weights tuned by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \varphi(k)(\hat{w}_3^T(k)\varphi(k) + l_1 e_1(k))$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$ are learning rate parameters or adaptation gains, $l_1 \in R$, $l_2 \in R$, $A \in R$, $B \in R$, P , Q , and G are design parameters.

with the critic NN weights being tuned by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \varphi(k)(\hat{w}_3^T(k)\varphi(k) + l_1 e_1(k)) \quad (6.32)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$, $l_1 \in R$, $l_2 \in R$, $A \in R$, and $B \in R$ are design parameters. Let the virtual and actual control inputs be given by (6.7) and (6.16), respectively. The tracking error, $e_1(k)$ (6.3), and $e_2(k)$ (6.9), the NN weight estimates, $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are UUB, with the bounds specifically given by (6.A.25) through (6.A.29) provided the design parameters

are selected as:

$$1. \quad 0 < \alpha_1 \|\phi(\cdot)\|^2 < 1 \quad (6.33)$$

$$2. \quad 0 < \alpha_2 \|\sigma(\cdot)\|^2 < l_2 \quad (6.34)$$

$$3. \quad 0 < \alpha_3 \|\varphi(\cdot)\|^2 < 1 \quad (6.35)$$

$$4. \quad 0 < |l_1| < \frac{1}{2g_{1M}\sqrt{11 + \frac{1}{l_2}}} \quad (6.36)$$

$$5. \quad 0 < l_2 < \frac{-g_{2M} + \sqrt{(7 + g_{2M}^2)}}{14g_{2M}} \quad (6.37)$$

$$6. \quad 0 < 3A^2 + \frac{3B^2}{k_2} \leq \frac{1}{2} \quad (6.38)$$

Proof: See Appendix 6.A at the end of this chapter.

Remarks:

1. The weight-tuning Equation 6.30 through 6.32 show that the critic and the action-generating NNs have a semirecurrent structure where the output from each NN node in both the input and output layers is fed back to their corresponding inputs. The action and critic NNs are tuned by a supervisory conventional input based on tracking error.
2. The mutual dependence between the two NNs (action-generating and critic NNs) in the adaptive critic NN architecture results in coupled tuning law equations. Moreover, additional complexities arise due to further interactions from the second action-generating NN, which is tuned by the same critic NN. In fact, the proposed NN controller with two action-generating NNs and a critic NN can be viewed as an NN controller of third order (Lewis et al. 2002). In effect, these two-layer NNs function together as a single NN with three tunable layers.
3. The proposed scheme renders a well-defined controller (see Remark (1) from Theorem 6.2.1) by overcoming the problem of $\hat{g}_i(k)$, $\forall i = 1, 2$ becoming zero since a single action-generating NN is used to approximate the nonlinear functions, $f(\cdot)$ and $g(\cdot)$, in (6.5) and (6.15).
4. It is important to note that in this theorem the CE and LIP assumptions for the adaptive critic NN controller are relaxed, in contrast to standard work in discrete-time adaptive control (Astrom and Wittenmark 1995). The LIP assumption is relaxed by using the NN approximation property. The Lyapunov function shown in the Appendix 6.A is of

the form

$$J(k) = \frac{e_1^2(k)}{8g_{1M}^2} + \frac{e_2^2(k)}{8l_2g_{2M}^2} + \sum_{i=1}^3 \frac{1}{\alpha_i} \tilde{w}_i^T(k) \tilde{w}_i(k)$$

which weights the tracking errors, $e_1^2(k)$ and $e_2^2(k)$ the NN estimation errors for the controller, $\tilde{w}_1(k)$, $\tilde{w}_2(k)$, and $\tilde{w}_3(k)$. The Lyapunov function now includes not only the estimation errors from the two action-generating NNs, but also a term from the critic NN. The proof is exceedingly complex due to the presence of several different variables. However, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner.

5. The PE condition is not required for the adaptive critic NN controller. In fact, the NN weight updating rules (6.30) through (6.32) are simpler than in Lewis et al. (1999) and Jagannathan (2001) since they do not include an extra term, referred to as discrete-time ε -modification (Jagannathan and Lewis 1996), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms. The Lyapunov proof demonstrates that the additional term in the weight tuning is not required. As a result, the complexity of the proof as well as the computational overhead is reduced significantly without the PE condition even though a separate critic NN is added.
6. The action-generating NN weights are tuned online using the critic NN output. A range of performance indices can be selected for evaluating the performance of the adaptive critic NN controller.
7. The NN weights may be initialized at zero and stability will be maintained by the outer-loop conventional controller until the NNs learn. This means that there is no off-line learning phase needed for any of the NNs. Note that the critic NN is not trained off-line when compared to standard adaptive critic NN controllers (Werbos 1991, 1992).
8. Though two-layer NNs are used to approximate certain nonlinear dynamics, any function approximator such as B -splines, RBFs, and fuzzy basis functions can also be employed without changing the design.
9. Conditions (6.33) to (6.35) can be checked easily. For instance, suppose that the hidden layer of the action-generating NN consists of n_1 nodes with the radial basis, the hyperbolic tangent sigmoid or the log sigmoid transfer function as the activation function, then

$\|\varphi(\cdot)\|^2 \leq n_1$. The α_1 can be selected as $0 < \alpha_o < (1/n_1)$ to satisfy (6.33). Similar analysis can be performed for (6.34) and (6.35) except that the adaptation gains are tied together with the outer-loop conventional controller gains. Equations 6.33 through 6.35 indicate that for faster learning, the closed-loop poles of the system have to be near the origin.

10. The outer-loop controller gains, l_1 and l_2 , have to satisfy (6.36) and (6.37) in order for the closed-loop system to be stable. Given the bounds, g_{1m} and g_{2m} , a suitable value of l_1 and l_2 can be selected easily so that the closed-loop poles will remain within the unit circle. Such constraints do not exist for controller design parameters and NN learning rates or adaptation gains in continuous time. These parameters are selected arbitrarily in NN design of continuous-time systems.

Example 6.2.1 (Adaptive Critic NN Control for General Nonlinear Systems in Nonstrict Feedback Form): The purpose of this example is to verify the performance of the adaptive critic NN controller whose control objective is to make the state, $x_1(k)$ to follow a desired trajectory $x_{1d}(k)$. The adaptive critic NN controller developed in Section 6.2.2 is used on the following nonlinear system, given in nonstrict feedback form, as

$$x_1(k+1) = -\frac{3}{16} \left(\frac{x_1(k)}{1+x_2^2(k)} \right) + 0.5x_1(k) + (3 + 1.5 \sin(x_2(k)))x_2(k) \quad (6.39)$$

$$x_2(k+1) = -\frac{1}{16} \left(\frac{x_2(k)}{1+x_1^2(k)} \right) + x_2(k) + (0.2 + 0.15 \cos(x_1(k)))u(k) \quad (6.40)$$

where $x_i(k) \in R$, $i = 1, 2$ are the states and $u(k) \in R$ is the control input. Note that both $f_1(k)$ and $g_1(k)$ include state $x_2(k)$.

The reference signal was selected as $x_{1d}(k) = 2 \sin(\omega kT + \xi)$, where $\omega = 0.5$, $\xi = \pi/2$ with a sampling interval of $T = 50$ msec. The total simulation time is taken as 250 sec. The gains of the proportional controller are selected a priori as $l_1 = -0.2$ and $l_2 = 0.3$. By adding delayed values of the tracking error in (6.7) and (6.15), a PD controller results.

The NN1 $\hat{w}_1^T \phi(k)$, NN2 $\hat{w}_2^T \sigma(k)$, and critic NN3 $\hat{w}_3^T \varphi(k)$ are considered to have 15 nodes each in their hidden layers. The number of nodes in the hidden layer of an NN heavily influences the complexity of the NN system. In general, the larger the number of hidden-layer neurons, the more complex the NN system

is and higher the expected accuracy in approximation. Hence, there is a trade-off between complexity and accuracy in the choice of the number of nodes in the hidden layer.

For weight updating, the learning rates are selected as $\alpha_1 = 0.1$, $\alpha_2 = 0.03$, and $\alpha_3 = 0.1$. Parameters A and B are selected as $A = 0.1$ and $B = 0.01$. The inputs to NN1, NN2, and critic NN3 are taken as $z_1(k) \in R^{3 \times 1}$, $z_2(k) \in R^{5 \times 1}$, and $z_3(k) \in R^{3 \times 1}$, respectively. The design parameter is chosen as $P = Q = G = 1$. All the initial weights are selected at random over an interval of $[0, 1]$ and all the activation functions used in the hidden layer are hyperbolic tangent sigmoid functions. The input-to-hidden layer weights, once randomly selected, are not tuned (Igelnik and Pao 1995).

Figure 6.2 illustrates the performance of the adaptive critic NN controller. From the figure, it is clear that the system tracking performance is superior with the adaptive critic NN controller. The NN control input is presented in Figure 6.3 where the control input is sufficiently smooth and it can be implemented in today's embedded system hardware. Figure 6.4 and Figure 6.5 depict the performance of the proportional controller and its control input without the NN being included. The proportional gains were not altered in both the simulations. From Figure 6.4, it is clear that the tracking performance has deteriorated in comparison with Figure 6.2. The NN is not trained off-line and the weights are initialized at zero.

Example 6.2.2 (Adaptive Critic NN Controller for SI Engines: A Practical Example): The dynamics of a single-cylinder SI engine, which is represented

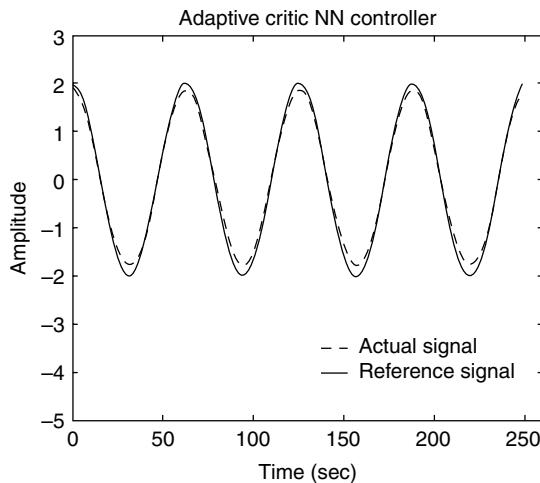


FIGURE 6.2 Adaptive critic NN controller performance.

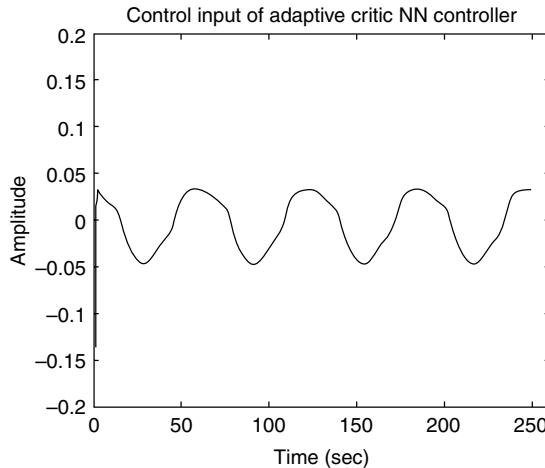


FIGURE 6.3 NN control input.

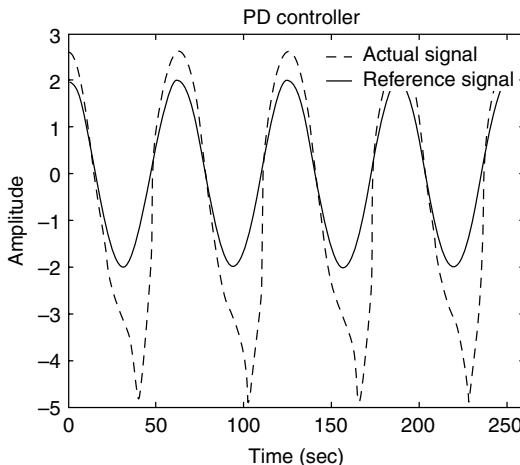


FIGURE 6.4 Performance of a conventional controller.

in state-space form has a striking resemblance with the experimental data, is given by Daw et al. (1997)

$$\begin{aligned} x_1(k+1) &= F(k) \times x_1(k) + (1 - F(k)) \times AF - (F(k)/R) \times CE(k) \\ &\quad \times x_2(k) + d'_1(k) \end{aligned} \quad (6.41)$$

$$x_2(k+1) = F(k) \times (1 - CE(k)) \times x_2(k) + MF + u(k) + d'_2(k) \quad (6.42)$$

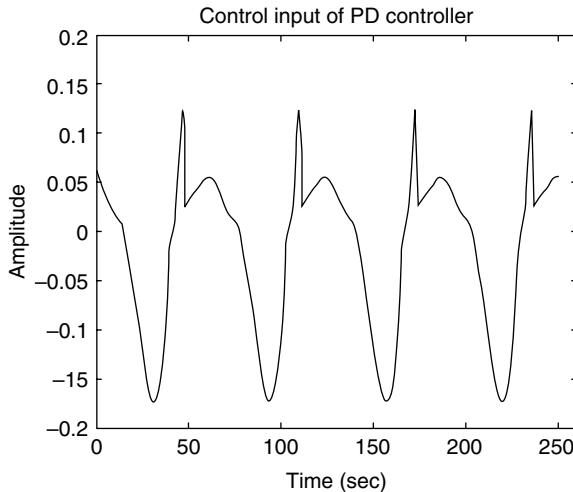


FIGURE 6.5 Control input.

and

$$CE(k) = \frac{CE_{\max}}{1 + 100 - ((1/R) \times (x_2(k)/x_1(k)) - \phi_m)/(\phi_u - \phi_l)} \quad \phi_m = \frac{\phi_u + \phi_l}{2} \quad (6.43)$$

where $x_1(k)$ is the mass of air before k th burn, $x_2(k)$ is the mass of fuel before k th burn, $u(k)$ is the small change in mass of fresh fuel per cycle, MF is the mass of fresh fuel per cycle, AF is the mass of fresh air fed per cycle and the following parameters whose values are indicated by engine or fuel characteristics and $F(k)$ is the fraction of cylinder gas remaining, R is the stoichiometric air-fuel ratio, approximately 14.6, $CE(k)$ is the combustion efficiency, ϕ_m , ϕ_l , ϕ_u are system parameters, and $d'_1(k)$ and $d'_2(k)$ are unknown but bounded disturbances. The dynamic Equation 6.41 through 6.43 are in nonstrict feedback form.

Literature shows that controlling engines at lean operation can reduce emissions as much as 30% (Inoue et al. 1993) and it improves fuel efficiency by about 5 to 10%. Similarly, EGR is a common method used in today's engines to reduce unwanted exhaust gases. Operating an engine with high EGR levels is shown to significantly reduce the NO_x , CO, and unburned hydrocarbon products. However, the engine exhibits strong cyclic dispersion in heat release causing instability in engine operation both at lean fuel and with high EGR levels (Davis et al. 1999). Therefore, automotive engines currently do not operate lean or with high EGR levels.

This example is used to evaluate the performance of the proposed adaptive critic NN controller for the practical application, where the objective is to

reduce cyclic dispersion in heat release. The cyclic dispersion in heat release is reduced when the variation in the equivalence ratio, $(x_2(k)/x_1(k))$ is reduced and this goal can be achieved by driving both the system states close enough to their desired targets. The simulation parameters are selected as: 1000 cycles are considered at an equivalence ratio of 0.71 with $R = 14.6$, $F(k) = 0.14$, mass of new air = 1.0, the standard deviation of mass of new fuel is 0.007, $\phi_u = 0.685$, $\phi_l = 0.665$, the desired mass of air is taken as $X_{1d} = 0.9058$, and the desired mass of fuel is calculated as $X_{2d} = R \times 0.71 \times X_{1d} = 9.3895$. A 5% unknown noise is added to the residual gas fraction as a way to include stochastic perturbation of system parameters. The gains of the outer-loop conventional controller are selected as $l_1 = l_2 = 0.1$, respectively.

The NN1 $\hat{w}_1^T \phi(k)$, NN2 $\hat{w}_2^T \sigma(k)$, and critic NN3 $\hat{w}_3^T \varphi(k)$ were selected with 15 nodes each in their hidden layers. For weight updating the learning rates are chosen as $\alpha_1 = 0.01$, $\alpha_2 = 0.001$, and $\alpha_3 = 0.1$. Parameters A and B are selected as $A = 0.1$, $B = 0.01$, and $P = Q = G = 1$. The inputs to NN1, NN2, and critic NN3 are taken as $z_1(k) \in R^{3 \times 1}$, $z_2(k) \in R^{5 \times 1}$, and $z_3(k) \in R^{3 \times 1}$, respectively. All the initial weights are selected randomly over an interval of $[0, 1]$ and all the activation functions in the hidden layer are hyperbolic tangent sigmoid functions.

Figure 6.6 (He and Jagannathan 2003b) shows that the error between the actual and desired equivalence ratio is bounded with the adaptive critic NN controller even though the combustion process dynamics comprising the residual gas fraction and combustion efficiency are considered unknown. In fact, it is very difficult to predict the residual fraction left over in each cylinder after

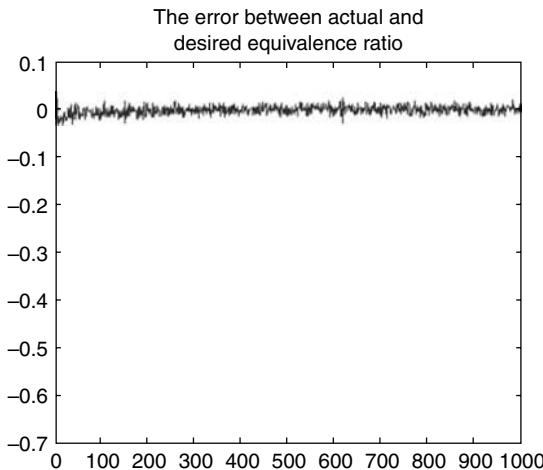


FIGURE 6.6 Equivalence ratio error.

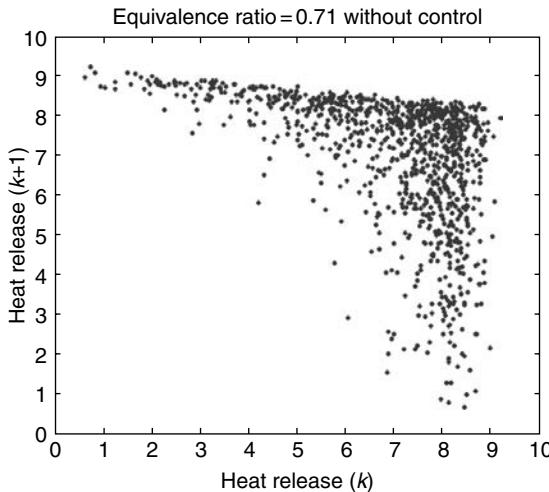


FIGURE 6.7 Cyclic dispersion without control.

combustion. The cyclic dispersion in heat release is depicted in Figure 6.7, which indicates that without any control, the engine performance is unstable. Moreover, an after-treatment device such as a catalytic converter is typically used to absorb the unwanted exhaust gases. In order for an after-treatment device to function properly, the engine must operate near stoichiometric conditions where the equivalence ratio is equal to one.

Figure 6.8 shows that the engine still works satisfactorily at lean conditions under small perturbation on both the residual gas fraction and fuel, even though minimal dispersion in heat release is noted. This amount of cyclic dispersion in heat release is considered acceptable. To evaluate the contributions of the NN, the NNs were disengaged from the inner loop. From Figure 6.9, it is clear that the unacceptable amount of dispersion in heat release still exists.

6.3 OUTPUT FEEDBACK NN CONTROLLER DESIGN

In the previous section, the adaptive NN control of nonlinear discrete-time system was presented using two different NN control architectures and by assuming that the states of the nonlinear system are available for measurement. However, if certain states are not available for measurement due to sensor costs as in complex nonlinear industrial systems, the control law developed in the previous section cannot be implemented in practice. To overcome this limitation, in the output feedback control design an observer is first designed to estimate

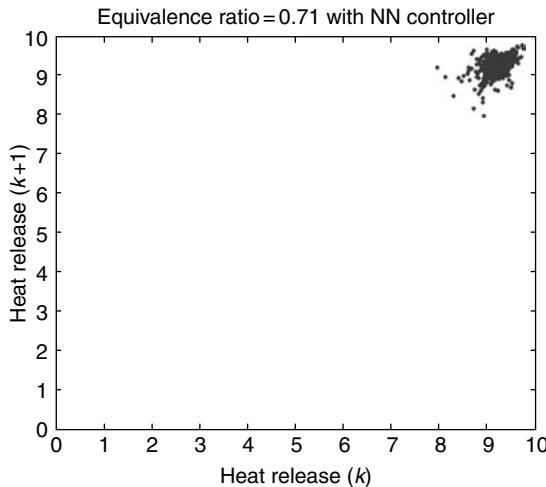


FIGURE 6.8 Heat release with NN controller.

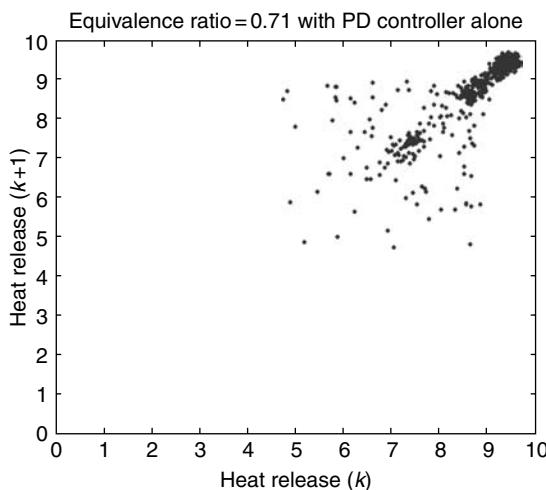


FIGURE 6.9 Heat release with a conventional controller.

certain system states and then the estimated values will be substituted for the unavailable ones.

All available output feedback controller designs were given for only nonlinear discrete-time systems (He and Jagannathan 2004) in strict feedback form. In this section, an adaptive NN output feedback controller from He and

Jagannathan (2005) is presented to deliver a desired tracking performance for a class of discrete-time nonlinear systems in nonstrict feedback form (6.1). The proposed adaptive NN output feedback controller design employs the backstepping approach discussed in Section 6.1.2. Besides using two NN for generating the virtual and real control inputs via backstepping, a third NN is used to estimate a certain state of the nonlinear system. Due to the presence of three NNs, the closed-loop analysis is quite involved, similar to the case of adaptive critic, but Lyapunov stability is still demonstrated. The NN controller scheme does not require an off-line learning phase and the NN weights can be initialized randomly. The UUB of the closed-loop tracking and the estimation errors are shown.

The present control objective is to drive the system state, $x_1(k)$ to track a desired trajectory, $x_{1d}(k)$ even when the state $x_2(k)$ is unavailable. We will design an NN observer to estimate this unavailable state. Subsequently, the estimated state is used to design the adaptive NN output feedback controller. Two-layer NNs are employed for the NN observer and the controller. The weight-updating rule for the NN observer and the closed-loop stability of the observer-controller system are analyzed by using the Lyapunov approach in the next section.

The weights of the first layer for the two-layer NN are selected initially at random to provide a basis (Igelnik and Pao 1995) and then they are held constant throughout the tuning process, as they are not dependent upon time. By fixing the weights of the first layer, computational overheads will be reduced, though it may require a larger number of hidden-layer neurons for suitable approximation.

Before we proceed with the observer design, it is necessary to review the notation that is used in this chapter. Throughout this section, all quantities with $\hat{\cdot}$ represent estimated variables, quantities with \sim represent the estimation error. Subscripts o and c refer to the observer and the controller quantities, respectively. The design of the NN observer is discussed next.

6.3.1 NN OBSERVER DESIGN

Recall the nonlinear system from (6.2). Writing the nonlinear system (6.2) into a vector form as

$$x(k+1) = f(k) + d'(k) \quad (6.44)$$

where

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \quad f(k) = \begin{bmatrix} f_1(k) + g_1(k)x_2(k) \\ f_2(k) + g_2(k)u(k) \end{bmatrix} \quad (6.45)$$

and

$$d'(k) = \begin{bmatrix} d'_1(k) \\ d'_2(k) \end{bmatrix} \quad (6.46)$$

The term $f(k - 1)$ can be viewed as an unknown smooth function vector and therefore it can be approximated by a two-layer NN. The outputs of the NN observer are $\hat{x}_1(k)$ and $\hat{x}_2(k)$, which denote the estimated values of $x_1(k)$ and $x_2(k)$, respectively. Here, we assume that the initial value of $u(0)$ is bounded. In the next section, it is shown via Lyapunov's analysis that all the values of $u(k)$ are bounded $\forall k \in R$. The proposed NN observer for the system (6.44) is defined as

$$\hat{x}(k) = \hat{w}_o^T(k - 1)\varphi(v_o^T\hat{z}_o(k - 1)) = \hat{w}_o^T(k - 1)\varphi(\hat{z}_o(k - 1)) \quad (6.47)$$

where $\hat{x}(k)$ is the estimated value of $x(k)$ and

$$\hat{x}(k) = \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \end{bmatrix}_{2 \times 1} \quad \hat{z}_o(k - 1) = \begin{bmatrix} \hat{x}_1(k - 1) \\ \hat{x}_2(k - 1) \\ u(k - 1) \end{bmatrix}_{3 \times 1} \quad (6.48)$$

Note that the vector, $\hat{z}_o(k - 1)$ is the input to the NN observer, the matrices, $\hat{w}_o \in R^{n_o \times 2}$ and $v_o \in R^{3 \times n_o}$, represent the hidden-to-output layer and input-to-hidden layer weights, respectively, whereas n_o denotes the number of the nodes in the hidden layer.

Assumption 6.3.1: The ideal weights and activation functions of the observer NN are bounded by known positive values so that

$$\|w_o(k)\| \leq w_{o \max} \quad \text{and} \quad \|\varphi(k)\| \leq \varphi_{\max} \quad (6.49)$$

Now define the state estimation errors as

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k) \quad i = 1, 2 \quad (6.50)$$

and the estimation errors can be expressed compactly in a vector form as

$$\tilde{x}(k) = \hat{x}(k) - x(k) \quad (6.51)$$

where $\tilde{x}(k) \in R^2$. The unknown nonlinear smooth function $f(k - 1)$ is approximated using a two-layer NN as

$$\begin{aligned} f(k - 1) &= w_o^T(k - 1)\varphi(v_o^T z_o(k - 1)) + \varepsilon_o(k - 1) \\ &= w_o^T(k - 1)\varphi(z_o(k - 1)) + \varepsilon_o(k - 1) \end{aligned} \quad (6.52)$$

where the matrix, $w_o \in R^{n_o \times 2}$, represents the target weights of the output layer of the NN observer, v_o is the target weight matrix of the input-to-hidden layer

and $\varepsilon_o(k - 1) \in R$ is the functional reconstruction error, which is considered to be bounded above by a known positive constant $\|\varepsilon_o(k)\| \leq \varepsilon_{oM}$.

Combining (6.44), (6.47), (6.51), and (6.52) to obtain the estimation error dynamics as

$$\begin{aligned}\tilde{x}(k) &= \hat{x}(k) - x(k) = \hat{w}_o^T(k - 1)\varphi(\hat{z}_o(k - 1)) - w_o^T(k - 1)\varphi(z_o(k - 1)) \\ &\quad - \varepsilon_o(k - 1) - d'(k - 1) \\ &= \xi_o(k - 1) + d_o(k - 1)\end{aligned}\tag{6.53}$$

where

$$\begin{aligned}\xi_o(k - 1) &= \tilde{w}_o^T(k - 1)\varphi(\hat{z}_o(k - 1)) = (\hat{w}_o(k - 1) - w_o(k - 1))^T \\ &\quad \times \varphi(\hat{z}_o(k - 1))\end{aligned}\tag{6.54}$$

$$w_o^T(k - 1)\varphi(\tilde{z}_o(k - 1)) = w_o^T(k - 1)\varphi(\hat{z}_o(k - 1)) - w_o^T(k - 1)\varphi(z_o(k - 1))\tag{6.55}$$

and

$$d_o(k - 1) = w_o^T(k - 1)\varphi(\tilde{z}_o(k - 1)) - d'(k - 1) - \varepsilon_o(k - 1)\tag{6.56}$$

It is clear from (6.56) that $d_o(k - 1)$ is bounded, since $w_o^T(k - 1)\varphi(\tilde{z}_o(k - 1))$, $d'(k - 1)$, and $\varepsilon_o(k - 1)$ terms are individually bounded above. It is now important to demonstrate the upper bound on the estimation errors and the closed-loop stability of the nonlinear system.

6.3.2 ADAPTIVE NN CONTROLLER DESIGN

The adaptive NN output feedback controller design using the backstepping approach is first introduced and the weight-updating rules are derived for the observer and controller via Lyapunov analysis. Theorem 6.3.1 ensures that all the signals in the system are bounded. To proceed, the following mild assumption is required.

Assumption 6.3.2: The target weights and activation functions of the two control NNs are bounded by known positive values so that

$$\|w_1(k)\| \leq w_{1\max} \quad \|w_2(k)\| \leq w_{2\max} \quad \|\phi(\cdot)\| \leq \phi_{\max} \quad \text{and} \quad \|\sigma(\cdot)\| \leq \sigma_{\max}\tag{6.57}$$

Next the adaptive NN output feedback control design is discussed in a step-by-step manner.

Step 1. Virtual controller design

Define the tracking error between the actual and desired trajectory as

$$e_1(k) = x_1(k) - x_{1d}(k) \quad (6.58)$$

where $x_{1d}(k)$ is the desired trajectory. Combining with (6.44), (6.58) can be rewritten as

$$\begin{aligned} e_1(k+1) &= x_1(k+1) - x_{1d}(k+1) \\ &= f_1(k) + g_1(k)x_2(k) - x_{1d}(k+1) + d'_1(k) \end{aligned} \quad (6.59)$$

By viewing $x_2(k)$ as a virtual control input, a desired feedback control signal can be selected as

$$x_{2d}(k) = \frac{1}{g_1(k)}(-f_1(k) + x_{1d}(k+1)) \quad (6.60)$$

Since $f_1(k)$ and $g_1(k)$ are unknown nonlinear and smooth functions, the desired feedback control $x_{2d}(k)$ cannot be implemented in practice. However, $x_{2d}(k)$ is a smooth function of $x_1(k)$ and $x_2(k)$. By utilizing a two-layer NN to approximate this unknown function, $x_{2d}(k)$ can be expressed as

$$x_{2d}(k) = w_1^T(k)\phi(v_1^T x(k)) + \varepsilon_1(k) = w_1^T(k)\phi(k) + \varepsilon_1(k) \quad (6.61)$$

where $w_1(k) \in R^{n_1 \times 1}$ denotes the constant target hidden-to-output layer weights, the matrix, v_1 , represents the input-to-hidden layer weights, n_1 is the number of nodes in the hidden layer, $\phi(k)$ is the hidden-layer activation function vector, and $\varepsilon_1(k)$ is the reconstruction error vector assumed to be bounded above $\|\varepsilon_1(k)\| \leq \varepsilon_{1M}$ with the input to the NN taken as $x(k)$. Only the hidden-layer NN weights are a function of time, whereas the input-layer weights are selected initially at random and held constant (Igelnik and Pao 1995).

Since $x_2(k)$ is unavailable, the input to the NN is chosen as $\hat{x}(k)$. Consequently, the virtual control input is chosen as

$$\hat{x}_{2d}(k) = \hat{w}_1^T(k)\phi(\hat{x}(k)) \quad (6.62)$$

where $\hat{w}_1(k)$ is the actual weight matrix for the first NN in the controller design. Define the error in weights during estimation by

$$\tilde{w}_1(k) = \hat{w}_1(k) - w_1(k) \quad (6.63)$$

Define the error between $x_2(k)$ and $\hat{x}_{2d}(k)$ as

$$e_2(k) = x_2(k) - \hat{x}_{2d}(k) \quad (6.64)$$

Equation 6.58 can be expressed using (6.59) for $x_2(k)$ as

$$e_1(k+1) = f_1(k) + g_1(k)(e_2(k) + \hat{x}_{2d}(k)) - x_{1d}(k+1) + d'_1(k) \quad (6.65)$$

or equivalently,

$$\begin{aligned} e_1(k+1) &= f_1(k) + g_1(k)(e_2(k) + x_{2d}(k) - x_{2d}(k) + \hat{x}_{2d}(k)) \\ &\quad - x_{1d}(k+1) + d'_1(k) \\ &= g_1(k)(e_2(k) - w_1^T(k)\phi(x(k)) - \varepsilon_1(k) + \hat{w}_1^T(k)\phi(\hat{x}(k))) + d'_1(k) \\ &= g_1(k)(e_2(k) + \tilde{w}_1^T(k)\phi(\hat{x}(k)) + w_1^T(k)\phi(\tilde{x}(k)) - \varepsilon_1(k)) + d'_1(k) \\ &= g_1(k)(e_2(k) + \xi_1(k) + d_1(k)) \end{aligned} \quad (6.66)$$

where

$$\xi_1(k) = \tilde{w}_1^T(k)\phi(\hat{x}(k)) \quad (6.67)$$

$$w_1^T(k)\phi(\tilde{x}(k)) = w_1^T(k)(\phi(\hat{x}(k)) - \phi(x(k))) \quad (6.68)$$

and

$$d_1(k) = \frac{d'_1(k)}{g_1(k)} - \varepsilon_1(k) + w_1^T(k)\phi(\tilde{x}(k)) \quad (6.69)$$

Note that $d_1(k)$ is bounded given the fact that $\varepsilon_1(k)$, $w_1^T(k)\phi(\tilde{x}(k))$, $d'_1(k)$, and $g_1(k)$ are all bounded.

Step 2: Design of the control input $u(k)$

Rewrite the error $e_2(k)$ from (6.64) as

$$\begin{aligned} e_2(k+1) &= x_2(k+1) - \hat{x}_{2d}(k+1) \\ &= f_2(k) + g_2(k)u(k) - \hat{x}_{2d}(k+1) + d'_2(k) \end{aligned} \quad (6.70)$$

where $\hat{x}_{2d}(k+1)$ is the future value of $\hat{x}_{2d}(k)$. Unfortunately, $\hat{x}_{2d}(k+1)$ is not available in the current time step. However, from (6.59) and (6.62), it is clear that $\hat{x}_{2d}(k+1)$ is a smooth nonlinear function of $x(k)$, $\hat{x}_{2d}(k)$, and the system error, $e_1(k)$ and $e_2(k)$. Consequently, term $\hat{x}_{2d}(k+1)$ can be approximated by using an NN if the system states and error vector can be restricted to lie within a compact set. In fact, the following theorem demonstrates that the states stay within the compact set. On the other hand, the value of $\hat{x}_{2d}(k+1)$ can also be obtained by employing a filter (Campos et al. 2000).

Select the desired control input by using a second NN in the controller design as

$$\begin{aligned} u_d(k) &= \frac{1}{g_2(k)}(-f_2(k) + \hat{x}_{2d}(k+1)) + l_1 e_1(k) \\ &= w_2^T(k)\sigma(v_2^T z(k)) + \varepsilon_2(k) = w_2^T(k)\sigma(k) + \varepsilon_2(k) \end{aligned} \quad (6.71)$$

where $w_2(k)$ is the matrix of target weights for the second NN, $\sigma(k)$ is the vector of activation functions, $\varepsilon_2(k)$ is the approximation error considered bounded above $\|\varepsilon_2(k)\| \leq \varepsilon_{2M}$, l_1 is the design constant, and $z(k)$ is the NN input given by (6.72). Considering the fact that state $x_2(k)$ cannot be measured, $z(k)$ cannot be obtained. Instead, $z(k)$ is substituted with $\hat{z}(k)$, where

$$z(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \hat{x}_{2d}(k) \\ e_1(k) \\ e_2(k) \end{bmatrix}_{5 \times 1} \quad \hat{z}(k) = \begin{bmatrix} \hat{x}_1(k) \\ \hat{x}_2(k) \\ \hat{x}_{2d}(k) \\ \hat{e}_1(k) \\ \hat{e}_2(k) \end{bmatrix}_{5 \times 1} \quad (6.72)$$

with

$$\hat{e}_1(k) = \hat{x}_1(k) - x_{1d}(k) \quad (6.73)$$

and

$$\hat{e}_2(k) = \hat{x}_2(k) - \hat{x}_{2d}(k) \quad (6.74)$$

The actual control input is now selected as

$$u(k) = \hat{w}_2^T(k)\sigma(\hat{z}(k)) \quad (6.75)$$

where $\hat{w}_2(k)$ is the actual weight matrix of the second NN in the controller design.

Substituting (6.71) and (6.75) into (6.70) yields

$$\begin{aligned}
 e_2(k+1) &= f_2(k) + g_2(k)(u_d(k) - u_d(k) + u(k)) - \hat{x}_{2d}(k+1) + d'_2(k) \\
 &= g_2(k)(-w_2^T(k)\sigma(z(k)) - \varepsilon_2(k) + \hat{w}_2^T(k)\sigma(\hat{z}(k)) \\
 &\quad + l_1 e_1(k)) + d'_2(k) \\
 &= g_2(k)(l_1 e_1(k) + \xi_2(k) + d_2(k))
 \end{aligned} \tag{6.76}$$

where

$$\tilde{w}_2(k) = \hat{w}_2(k) - w_2(k) \tag{6.77}$$

$$\xi_2(k) = \tilde{w}_2^T(k)\sigma(\hat{z}(k)) \tag{6.78}$$

$$w_2^T(k)\sigma(\tilde{z}(k)) = w_2^T(k)(\sigma(\hat{z}(k)) - \sigma(z(k))) \tag{6.79}$$

and

$$d_2(k) = \frac{d'_2}{g_2(k)} - \varepsilon_2(k) + w_2^T(k)\sigma(\tilde{z}(k)) \tag{6.80}$$

Note $d_2(k)$ is bounded since $\varepsilon_2(k)$, $w_2^T(k)\sigma(\tilde{z}(k))$, $d'_2(k)$, and $g_2(k)$ are bounded.

Equation 6.66 and Equation 6.76 represent the closed-loop error dynamics of the nonstrict feedback nonlinear system. It is required to show that the estimation errors (6.53), the system errors (6.66) and (6.76) are suitably small and the NN weight matrices $\hat{w}_o(k)$, $\hat{w}_1(k)$, and $\hat{w}_2(k)$ for the observer and controller NNs, respectively, remain bounded, for then the control $u(k)$ is bounded. The structure of the proposed adaptive NN output feedback controller is shown in Figure 6.10 where the controller structure is naturally derived from the mathematical analysis.

The control NN1 is used to generate the virtual control signal $\hat{x}_{2d}(k)$ whereas the control NN2 generates the actual control input, $u(k)$. In the following section, Theorem 6.3.1 ensures that all the signals in the system are bounded when the novel weight-updating laws, presented in Table 6.3, are employed.

6.3.3 WEIGHT UPDATES FOR THE OUTPUT FEEDBACK CONTROLLER

Theorem 6.3.1 (NN Output Feedback Controller): Consider the nonlinear discrete-time system to be controlled, given by (6.1) and let Assumptions 6.1.1 and 6.3.2 hold. Let the NN reconstruction error bounds, ε_{oM} , ε_{1M} , and ε_{2M} and disturbance bounds, d_{1M} and d_{2M} be known constants. Let the observer NN

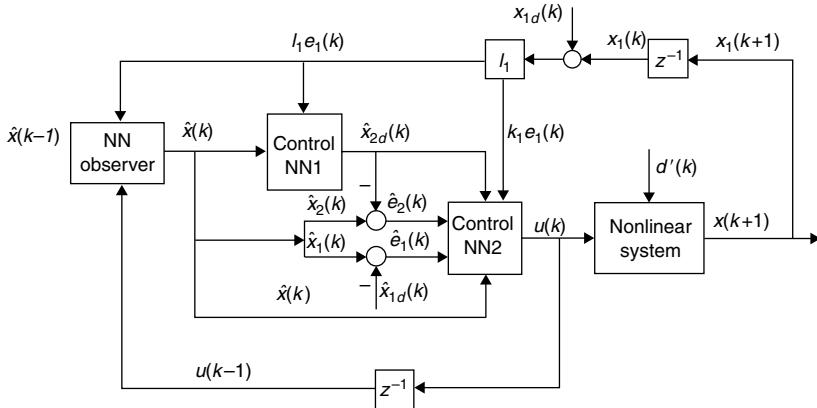
**FIGURE 6.10** Adaptive NN output feedback controller structure.

TABLE 6.3
Discrete-Time NN Output Feedback Controller

The NN observer is defined as

$$\hat{x}(k) = \hat{w}_o^T(k-1)\phi(v_o^T\hat{z}_o(k-1)) = \hat{w}_o^T(k-1)\phi(\hat{z}_o(k-1))$$

The virtual control input is

$$\hat{x}_{2d}(k) = \hat{w}_1^T(k)\phi(v_1^T\hat{x}(k))$$

The control input is given by

$$u(k) = \hat{w}_2^T(k)\sigma(v_2^T\hat{z}(k))$$

The NN weight tuning for the observer NN is given by

$$\hat{w}_o(k+1) = \hat{w}_o(k) - \alpha_o\varphi(\hat{z}_o(k))(\hat{w}_o^T(k)\varphi(\hat{z}_o(k)) + l_1 e_1(k+1)A)^T$$

and the virtual control NN weight tuning is provided by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1\phi(\hat{x}(k))(\hat{w}_1^T(k)\phi(\hat{x}(k)) + l_1 e_1(k))$$

with the actual control input NN weights tuned by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2\sigma(\hat{z}(k))(\hat{w}_1^T(k)\sigma(\hat{z}(k)) + l_1 e_1(k))$$

where $\alpha_o \in R$, $\alpha_1 \in R$, and $\alpha_2 \in R$, are learning rate parameters or adaptation gains and $l_1 \in R$ is a design parameter.

weight tuning be given by

$$\hat{w}_o(k+1) = \hat{w}_o(k) - \alpha_o \varphi(\hat{z}_o(k)) (\hat{w}_o^T(k) \varphi(\hat{z}_o(k)) + l_1 e_1(k+1) A)^T \quad (6.81)$$

where $A = [1, 1]^T$, with the virtual control NN weight tuning, NN1, provided by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi(\hat{x}(k)) (\hat{w}_1^T(k) \phi(\hat{x}(k)) + l_1 e_1(k)) \quad (6.82)$$

and the second control input weights, NN2, are tuned by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \sigma(\hat{z}(k)) (\hat{w}_2^T(k) \sigma(\hat{z}(k)) + l_1 e_1(k)) \quad (6.83)$$

where $\alpha_o \in R$, $\alpha_1 \in R$, $\alpha_2 \in R$, and $k_1 \in R$ are design parameters. Let the NN observer, virtual and actual control inputs be defined as (6.47), (6.62), and (6.75), respectively. The estimation errors (6.53), the tracking errors (6.66) and (6.76), and the NN weights $\hat{w}_o(k)$, $\hat{w}_1(k)$, and $\hat{w}_2(k)$ are UUB with the bounds specifically given by (6.B.12) through (6.B.17), provided the design parameters are selected as:

$$1. \quad 0 < \alpha_o \|\varphi(k)\|^2 < 1 \quad (6.84)$$

$$2. \quad 0 < \alpha_1 \|\phi(k)\|^2 < 1 \quad (6.85)$$

$$3. \quad 0 < \alpha_2 \|\sigma(k)\|^2 < 1 \quad (6.86)$$

$$4. \quad |l_1| < \frac{1}{3\sqrt{13}g_{1m}g_{2m}} \quad (6.87)$$

Proof: See Appendix 6.B at the end of this chapter.

Remarks:

1. A well-defined controller is developed since one NN approximates both the nonlinear functions, $f(\cdot)$ and $g(\cdot)$, in comparison with having two NNs to approximate each of the nonlinear functions. By selecting a single NN to compensate a certain nonlinear function, the extra terms that are normally included to show the estimate of $g(\cdot)$, denoted as $\hat{g}(\cdot)$, away from zero (Lewis et al. 1999) are not required.
2. It is important to note that in this theorem there is no PE condition, CE and LIP assumptions for the NN observer and NN controller, in contrast to standard work in discrete-time adaptive control (Astrom

and Wittenmark 1995). In our proof, the Lyapunov function shown in Appendix 6.B of the form

$$\begin{aligned} J(k) = & \frac{1}{4}\tilde{x}^T(k-1)\tilde{x}(k-1) + \frac{l_2}{6g_{1m}^2}e_1^2(k) + \frac{1}{6g_{2m}^2}e_2^2(k) + \frac{1}{\alpha_o}\text{tr} \\ & \times (\tilde{w}_o^T(k-1)\tilde{w}_o(k-1)) + \frac{1}{\alpha_1}\tilde{w}_1^T(k)\tilde{w}_1(k) + \frac{1}{\alpha_2}\tilde{w}_2^T(k)\tilde{w}_2(k) \end{aligned}$$

which weighs the delayed value of the estimation error, $\tilde{x}(k-1)$, tracking errors, $e_1^2(k)$ and $e_2^2(k)$, the NN estimation errors for the observer, $\tilde{w}_o(k-1)$, and the controller, $\tilde{w}_1(k)$ and $\tilde{w}_2(k)$. Though the proof is exceedingly complex, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner. Further, Lyapunov proof demonstrates that the additional term in the weight tuning referred to as discrete-time of ε -modification (Jagannathan and Lewis 1996), which is normally used to provide robustness due to the coupling in the proof between the tracking errors and NN weight estimation error terms is not required. As a result, the complexity of the proof as well as the computational overhead is reduced significantly without the PE condition.

3. A separation principle is normally employed in the design of an output feedback controller for linear systems. According to the separation principle, one may design a state observer and then in a separate design, a control system can be manufactured by assuming that the entire system is actually available as measurements. In other words, according to the separation principle, one may implement a controller by using the estimated observed states of a system and under certain conditions, the overall closed-loop system can be shown to be stable. Generally, the separation principle is not valid for nonlinear systems and hence, it is relaxed in the controller design.
4. There does not seem to be a well-established method for designing a smooth observer-controller for uncertain nonlinear systems in discrete time. The NN-based output feedback scheme is a good candidate for compensating structured and unstructured uncertainties.
5. Neural networks in the observer and the controller to approximate the nonlinear terms are spirited by a general methodology. However, different NN architectures with different weight-tuning laws render different controller structures as discussed in Section 6.2.
6. When the actual weights of the observer and the controller NNs are initialized to zero, the outer loop functions as either a proportional or a PD feedback controller (if an additional delayed term is included)

that keeps the system stable until the NN begins to learn. This means that there is no off-line learning phase needed.

7. Though two-layer NNs are used to approximate certain nonlinear dynamics, any function approximator such as B -splines, RBFs, fuzzy basis functions, can also be employed without changing the design.
8. Conditions (6.84) to (6.86) can be checked easily. For instance, suppose that the hidden layer of the observer NN consists of n_o nodes with the radial basis, the hyperbolic tangent sigmoid or the log sigmoid transfer function as the activation function, then $\|\varphi(\cdot)\|^2 \leq n_o$. The α_o can be selected as $0 < \alpha_o < 1/n_o$ to satisfy (6.84). Similar analysis can be performed for (6.85) and (6.86).
9. The controller gain l_1 has to satisfy (6.87) in order for the closed-loop system to be stable. Given the bounds, g_{1m} and g_{2m} , a suitable value of l_1 can be selected easily so that the closed-loop poles will remain within the unit circle.

Example 6.3.1 (Output Feedback Control of a Discrete-Time Nonlinear System): Consider the following nonlinear system, given in nonstrict feedback form, as

$$\begin{aligned} x_1(k+1) &= -\frac{1}{64} \frac{x_1(k)}{(5+x_2^2(k))} + x_1(k) + (2 + 0.1 \times (\cos(x_1(k)) \\ &\quad + \sin(x_2(k))))x_2(k) \\ x_2(k+1) &= -\frac{1}{16} \frac{x_2(k)}{(1+x_1^2(k))} - \left(\frac{7}{(1+x_1^2(k)+x_2^2(k))} \right) u(k) \\ y(k) &= x_1(k) \end{aligned} \tag{6.88}$$

where $x_i(k) \in R$, $i = 1, 2$ are the states, $u(k) \in R$ is the control input, $y(k) \in R$ is the system output, the state $x_1(k)$ is known via the output $y(k)$, whereas the state $x_2(k)$ is not measurable. Therefore, an NN observer is used to obtain an estimate of $x_2(k)$. Note that $f_1(k) = -(1/64)(x_1(k)/(5+x_2^2(k))) + x_1(k)$ and $g_1(k) = 2 + 0.1 \times (\cos(x_1(k)) + \sin(x_2(k)))$ are nonlinear functions of both states $x_1(k)$ and $x_2(k)$.

The objective is to drive the state, $x_1(k)$ to track a predefined reference signal using the proposed adaptive NN output feedback controller. The reference signal was selected as $x_{1d}(k) = 2 \sin(\omega kT + \xi)$, where $\omega = 0.5$, $\xi = \pi/2$ with a sampling interval of $T = 50$ msec. The total simulation time is 250 sec. The proportional controller gain is taken as $l_1 = -0.05$.

The number of hidden-layer neurons in the observer NN, $\hat{w}_o^T \varphi(k)$, controller NN1 and NN2, $\hat{w}_1^T \phi(k)$ and $\hat{w}_2^T \sigma(k)$ each was taken as 15. For weight updating,

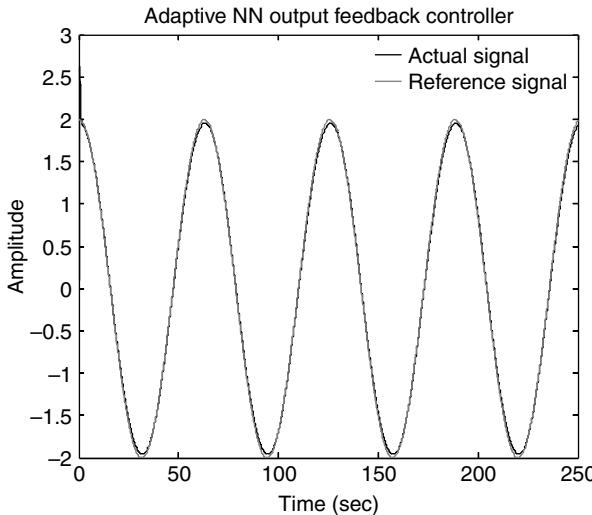


FIGURE 6.11 Adaptive NN output controller performance.

the learning rates are selected as $\alpha_o = 0.01$, $\alpha_1 = 0.1$, and $\alpha_2 = 0.1$. The inputs to observer NN, $\hat{w}_o^T \varphi(k)$, control NNs, $\hat{w}_1^T \phi(k)$ and $\hat{w}_2^T \sigma(k)$ are selected as $\hat{z}_o(k)$ (6.48), $\hat{x}(k)$ (6.48), and $\hat{z}(k)$ (6.72), respectively. The initial input-to-hidden layer weights for the three NNs are selected at random over an interval of $[0, 1]$ and all the activation functions used are hyperbolic tangent sigmoid functions. The initial weights for the hidden-to-output layers for all the three NNs are chosen to be zero.

Two cases are considered. First, the adaptive output feedback NN controller is applied to the system and then the NNs are removed so that the contributions of the NNs and the proportional controller can be evaluated. Figure 6.11 illustrates the tracking performance of the adaptive NN output feedback controller. From the figure, it is clear that the system tracking performance is superior with the adaptive NN output feedback controller even when one of the states is not measured. The NN control input is presented in Figure 6.12, where it clearly shows that the input is bounded. On the other hand, Figure 6.13 and Figure 6.14 display the performance of the proportional controller and the control input, respectively, without the NNs included. The proportional controller gain was not altered from the previous simulation. From Figure 6.13, it is clear that the tracking performance has deteriorated in comparison with Figure 6.11 though the tracking error and the control inputs are bounded. Since the outer-loop proportional controller ensures stability while the NNs learn, no off-line training phase is required for the NNs and their weights can be initialized to zero.

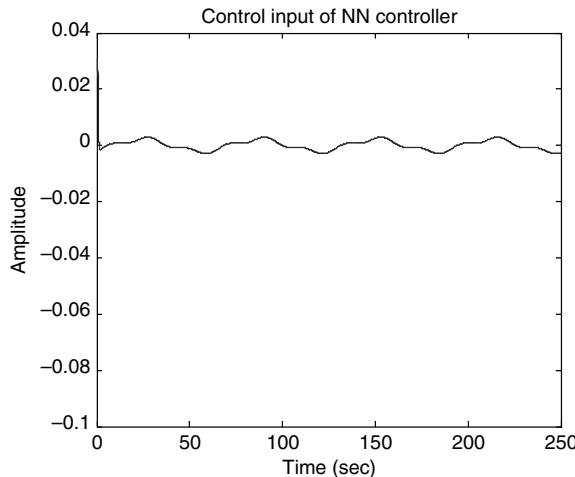


FIGURE 6.12 NN controller input.

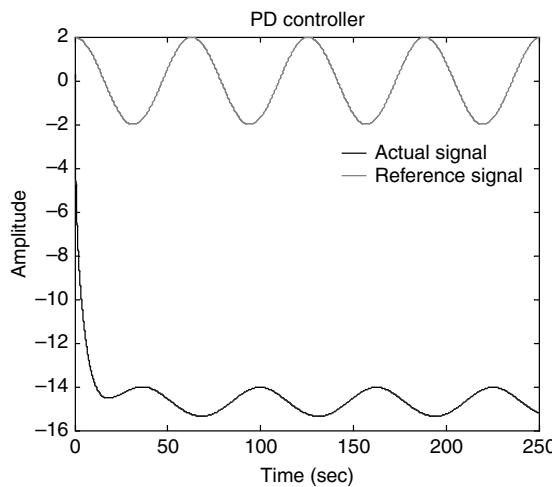


FIGURE 6.13 Proportional controller performance.

6.4 CONCLUSIONS

This chapter presents a suite of adaptive NN controllers to deliver a desired tracking performance for the control of a class of unknown nonlinear systems in discrete time when the system is represented in nonstrict feedback form. A well-defined controller methodology is developed where a single NN is used

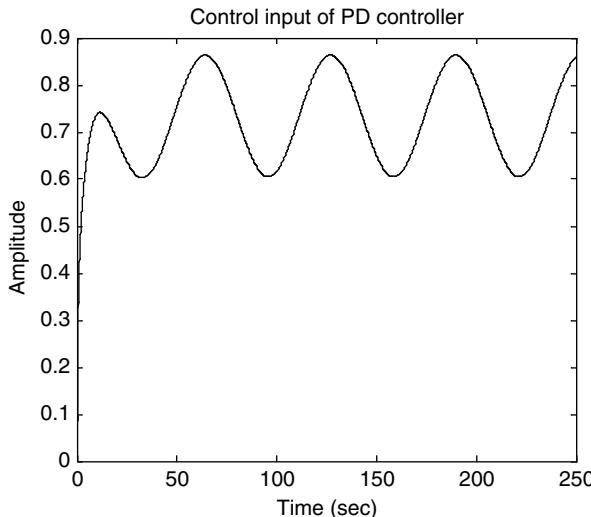


FIGURE 6.14 Proportional control input.

to compensate two nonlinear functions. Initially, tracking error and adaptive critic-based methodologies are used to design a controller for the nonlinear discrete-time system by assuming that the states are available for measurement. A comparison between the two NN control architectures is performed. Since it is very expensive to measure all the states of a nonlinear system, an adaptive NN output feedback controller is introduced using the backstepping approach. The stability analysis of the closed-loop control system is shown and the UUB of the closed-loop tracking and weight estimation errors is demonstrated. The backstepping NN approach does not require an off-line learning phase and the NN weights can be initialized at zero or at random.

REFERENCES

- Astrom, K.J. and Wittenmark, B., *Adaptive Control*, Addison Wesley, Reading, MA, 1995.
- Barron, A.R., Universal approximation bounds for superposition of a sigmoidal function, *IEEE Trans. Inform. Theory* 39, 930–945, 1993.
- Barto, A.G., Reinforcement learning and adaptive critic methods, in *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 469–492, 1992.
- Campos, J., Lewis, F.L., and Selmic, R., Backlash compensation with filtered prediction in discrete time nonlinear systems by dynamic inversion using neural networks, *Proc. IEEE Conf. Decis. Contr.*, 4, 3534–3540, 2000.

- Chen, F.C. and Khalil, H.K., Adaptive control of nonlinear discrete-time systems using neural networks, *IEEE Trans. Automat. Contr.*, 40, 791–801, 1995.
- Davis, Jr., Daw, C.S., Feldkamp, L.A., Hoard, J.W., Yuan, F., and Connolly, T., Method of controlling cyclic variation engine combustion, U.S. Patent, 5,921,221, 1999.
- Daw, C.S., Finney, C.E.A., Kennel, M.B., and Connolly, F.T., Observing and modeling nonlinear dynamics in an internal combustion engine, *Phys. Rev. E*, 57, 2811–2819, 1997.
- Ge, S.S., Li, G.Y., and Lee, T.H., Adaptive NN control for a class of strict-feedback discrete-time nonlinear systems via backstepping, *Proceedings of the IEEE Conference on Decision and Control*, Orlando, FL, pp. 3146–3151, 2001.
- He, P. and Jagannathan, S., Discrete-time neural network control of a class of nonlinear systems in non strict feedback form, *Proc. IEEE Conf. Decis. Contr.*, 6, 5703–5708, 2003a.
- He, P. and Jagannathan, S., Neuro emission controller for spark ignition engines, *Proc. Int. Joint Conf. Neural Netw.*, 2, 1535–1540, 2003b.
- He, P. and Jagannathan, S., Adaptive critic neural network-based controller for nonlinear systems with input constraints, *Proc. IEEE Conf. Decis. Contr.*, 6, 5709–5714, 2003c.
- He, P. and Jagannathan, S., Discrete-time neural network output feedback control of a class of nonlinear systems in nonstrict feedback form, *Proc. Am. Contr. Conf.*, Boston, 2439–2444, 2004.
- He, P., Chen, Z., and Jagannathan, S., Reinforcement-learning neural network-based control of nonlinear discrete-time systems in non-strict form, *Proc. IEEE Conf. Decis. Contr.*, 3, 2580–2585, 2005.
- Igelnik, B. and Pao, Y.H., Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, 6, 1320–1329, 1995.
- Inoue, T., Matsushita, S., Nakanishi, K., and Okano, H., Toyota lean combustion system — the third generation system, *Society of Automotive Engineers*, New York, SAE Technical Paper series, Pub. 930873, 1993.
- Jagannathan, S., Control of a class of nonlinear discrete-time systems using multi layer neural networks, *IEEE Trans. Neural Netw.*, 12, 1113–1120, 2001.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Automat. Contr.*, 41, 1693–1699, 1996.
- Kanellakopoulos, I., Kokotovic, P.V., and Morse, A.S., Systematic design of adaptive controllers for feedback linearizable systems, *IEEE Trans. Automat. Contr.*, 36, 1241–1253, 1991.
- Kokotovic, P.V., The joy of feedback: nonlinear and adaptive, *IEEE Contr. Syst. Mag.*, 12, 7–17, 1992.
- Kuljaca, O., Swamy, N., Lewis, F.L., and Kwan, C.M., Design and implementation of industrial neural network controller using backstepping, *IEEE Trans. Indust. Electron.*, 50, 193–201, 2003.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.

- Lewis, F.L., Jagannathan, S., and Yesilderek, A., *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, UK, 1999.
- Lewis, F.L., Campos, J., and Selmic, R., *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society for Industrial and Applied Mathematics, Philadelphia, 2002.
- Werbos, P.J., A menu of designs for reinforcement learning over time, in *Neural Networks for Control*, Miller, W.T., Sutton, R.S., and Werbos, P.J., Eds., MIT Press, Cambridge, pp. 67–95, 1991.
- Werbos, P.J., Neurocontrol and supervised learning: an overview and evaluation, in *Handbook of Intelligent Control*, White, D.A. and Sofge, D.A., Eds., Van Nostrand Reinhold, New York, pp. 65–90, 1992.
- Yeh, P.C. and Kokotovic, P.V., Adaptive output feedback design for a class of nonlinear discrete-time systems, *IEEE Trans. Automat. Contr.*, 40, 1663–1668, 1995.

PROBLEMS

SECTION 6.2

6.2-1: *Adaptive critic NN control.* The objective is to verify the performance of the adaptive critic NN controller whose control objective is to make the state, $x_1(k)$ follow a desired trajectory $x_{1d}(k)$. Design the adaptive critic NN controller for the following nonlinear system, given in nonstrict feedback form, as

$$x_1(k+1) = -\frac{1}{16} \left(\frac{x_1(k)}{1+x_2^2(k)} \right) + 0.2x_1(k) + (3 + 1.5 \cos(x_2(k)))x_2(k) \quad (6.89)$$

$$x_2(k+1) = -\frac{5}{16} \left(\frac{x_2(k)}{1+x_1^2(k)} \right) + x_2(k) + (0.2 + 0.15 \sin(x_1(k))) \times \cos(x_1(k))u(k) \quad (6.90)$$

where $x_i(k) \in R$, $i = 1, 2$ are the states, and $u(k) \in R$ is the control input. Note that both $f_1(k)$ and $g_1(k)$ include state $x_2(k)$.

The reference signal was selected as $x_{1d}(k) = 2 \sin(\omega kT + \xi)$, where $\omega = 0.5$, $\xi = \pi/2$ with a sampling interval of $T = 10$ msec. Compare the performance of the adaptive controller with that of a conventional proportional, integral, and derivative (PID) controller.

6.2-2: *Adaptive critic NN controller for SI engines.* Recall Example 6.2.2 where a one-layer NN is used to design a controller for minimizing the cyclic dispersion

in heat release. Design a two-layer NN controller for this example by appropriately deriving the weight-tuning updates similar to the two-layer NNs in Chapter 3.

6.2-3: *Adaptive critic NN controller for SI engines operating with EGR.* Use the dynamics of the SI engines operating with EGR levels from Chapter 10, design a one-layer NN controller for minimizing the cyclic dispersion in heat release. (*Hint:* Even though the EGR dynamics are given as an additional state equation, one can assume that it is held fixed. This in turn results in different nonlinear functions, $f_i(\cdot)$ and $g_i(\cdot)$ in the first two equations. However, it still turns out to be a nonstrict feedback nonlinear system.)

SECTION 6.3

6.3-1: *(Adaptive NN output feedback controller).* Consider the following nonlinear system, given in nonstrict feedback form, as

$$\begin{aligned} x_1(k+1) &= -\frac{1}{6} \frac{x_1(k)}{(5 + x_2^2(k))} + x_1(k) + (1 + 0.1(\cos(x_1(k)) \\ &\quad + \sin(x_2(k))))x_2(k) \\ x_2(k+1) &= -\frac{5}{16} \frac{x_2(k)}{(1 + x_1^2(k))} - \left(\frac{7}{(1 + x_1^2(k) + x_2^2(k))} \right) u(k) \\ y(k) &= x_1(k) \end{aligned} \tag{6.91}$$

where $x_i(k) \in R$, $i = 1, 2$ are the states, $u(k) \in R$ is the control input, $y(k) \in R$ is the system output, the state $x_1(k)$ is known via the output $y(k)$, whereas the state $x_2(k)$ is not measurable. Therefore, an NN observer is required to obtain an estimate of $x_2(k)$. Note that $f_1(k) = -\frac{1}{6}(x_1(k)/(5 + x_2^2(k))) + x_1(k)$ and $g_1(k) = 1 + 0.1(\cos(x_1(k)) + \sin(x_2(k)))$ are nonlinear functions of both $x_1(k)$ and $x_2(k)$. The objective is to drive the state, $x_1(k)$ to track a predefined reference signal as $x_{1d}(k) = 2 \sin(\omega kT + \xi)$, where $\omega = 0.5$, $\xi = \pi/2$ with a sampling interval of $T = 10$ msec. Simulate the controller using MATLAB®.

6.3-2: *(SI engine dynamics).* For Problem 6.2-2, design an output feedback controller to track a desired air value of 0.1.

APPENDIX 6.A

Proof of Theorem 6.2.1: Define the Lyapunov function candidate

$$J(k) = \frac{e_1^2(k)}{8g_{1M}^2} + \frac{e_2^2(k)}{8l_2g_{2M}^2} + \frac{1}{\alpha_1}\tilde{w}_1^T(k)\tilde{w}_1(k) + \frac{1}{\alpha_2}\tilde{w}_2^T(k)\tilde{w}_2(k) \quad (6.A.1)$$

where g_{1M} and g_{2M} are the upper bounds of function $g_1(k)$ and $g_2(k)$, respectively, given a compact set (see Assumption 6.1.2), l_2 , α_1 , and α_2 are design parameters (see Theorem 6.2.1).

The first difference of Lyapunov function is given by

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) \quad (6.A.2)$$

The first difference $\Delta J_1(k)$ is obtained using (6.11) as

$$\begin{aligned} \Delta J_1(k) &= \frac{1}{8g_{1M}^2}(e_1^2(k+1) - e_1^2(k)) \\ &= \frac{1}{8g_{1M}^2}((g_1(k)(l_1e_1(k) + e_2(k) + \zeta_1(k) + d_1(k)))^2 - e_1^2(k)) \\ &\leq \frac{1}{2} \left(\left(l_1^2 - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + e_2^2(k) + \zeta_1^2(k) + d_1^2(k) \right) \end{aligned} \quad (6.A.3)$$

Now taking the second term in the first difference (6.A.1) and substituting (6.17) into (6.A.1) to get

$$\begin{aligned} \Delta J_2(k) &= \frac{1}{8l_2g_{2M}^2}(e_2^2(k+1) - e_2^2(k)) \\ &= \frac{1}{8k_2g_{2M}^2}((g_2(k)(l_1e_1(k) + l_2e_2(k) + \zeta_2(k) + d_2(k)))^2 - e_2^2(k)) \\ &\leq \frac{1}{2} \left(\left(\frac{1}{l_2} \right) \left(l_2^2 - \frac{1}{3g_{2M}^2} \right) e_2^2(k) + \frac{\zeta_2^2(k)}{l_2} + \frac{d_2^2(k)}{l_2} + \frac{l_1^2e_1^2(k)}{l_2} \right) \end{aligned} \quad (6.A.4)$$

Taking the third term in (6.A.1) and substituting the weight updates from (6.21) and simplifying to get

$$\begin{aligned}
\Delta J_3(k) &= \frac{1}{\alpha_1} \tilde{w}_1^T(k+1) \tilde{w}_1(k+1) - \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) \\
&= \frac{1}{\alpha_1} [(I - \alpha_1 \phi(k) \phi^T(k)) \tilde{w}_1(k) - \alpha_1 \phi(k) (w_1^T(k) \phi(k) + l_1 e_1(k))]^T \\
&\quad \times [(I - \alpha_1 \phi(k) \phi^T(k)) \tilde{w}_1(k) - \alpha_1 \phi(k) (w_1^T(k) \phi(k) + l_1 e_1(k))] \\
&\quad - \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) \\
&= -(2 - \alpha_1 \phi^T(k) \phi(k)) \zeta_1^2(k) - 2(1 - \alpha_1 \phi^T(k) \phi(k)) (w_1^T \phi(k) \\
&\quad + l_1 e_1(k)) \zeta_1(k) + \alpha_1 \phi^T(k) \phi(k) (w_1^T \phi(k) + l_1 e_1(k))^2 \quad (6.A.5)
\end{aligned}$$

Taking the fourth term in (6.A.1) and substituting the weight updates from (6.22) and simplifying to get

$$\begin{aligned}
\Delta J_4(k) &= \frac{1}{\alpha_2} \tilde{w}_2^T(k+1) \tilde{w}_2(k+1) - \frac{1}{\alpha_2} \tilde{w}_2^T(k) \tilde{w}_2(k) \\
&= \frac{1}{\alpha_2} \left[\left(I - \frac{\alpha_2}{k_2} \sigma(k) \sigma^T(k) \right) \tilde{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k) (w_2^T \sigma(k) + l_2 e_2(k)) \right]^T \\
&\quad \times \left[\left(I - \frac{\alpha_2}{l_2} \sigma(k) \sigma^T(k) \right) \tilde{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k) (w_2^T \sigma(k) + l_2 e_2(k)) \right] \\
&\quad - \frac{1}{\alpha_2} \tilde{w}_2^T(k) \tilde{w}_2(k) \\
&= - \left(\frac{1}{l_2} \right) \left(2 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) \zeta_2^2(k) \\
&\quad - 2 \left(\frac{1}{l_2} \right) \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (w_2^T \sigma(k) + l_2 e_2(k)) \zeta_2(k) \\
&\quad + \frac{\alpha_2}{l_2^2} \sigma^T(k) \sigma(k) (w_2^T \sigma(k) + l_2 e_2(k))^2 \quad (6.A.6)
\end{aligned}$$

Combining (6.A.3) to (6.A.6) to get the first difference and simplifying to get

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k)$$

$$\begin{aligned} & \leq \frac{1}{2} \left(l_1^2 + \frac{l_1^2}{l_2} - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + \frac{1}{2l_2} \left(l_2^2 + l_2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) \\ & - \frac{1}{2} \xi_1^2(k) - \frac{1}{2k_2} \xi_2^2(k) - (1 - \alpha_1 \phi^T(k) \phi(k)) (\xi_1^2(k) + 2(w_1^T(k) \phi(k) \\ & + l_1 e_1(k)) \xi_1(k)) - \frac{1}{l_2} \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (\xi_2^2(k) + 2(w_2^T \sigma(k) \\ & + l_2 e_2(k)) \xi_2(k)) + \alpha_1 \phi^T(k) \phi(k) (w_1^T \phi(k) + l_1 e_1(k))^2 + \frac{\alpha_2}{l_2^2} (w_2^T \sigma(k) \\ & + l_2 e_2(k))^2 \sigma^T(k) \sigma(k) + d_1^2(k) + \frac{d_2^2(k)}{l_2} \\ \Delta J(k) & \leq \frac{1}{2} \left(5k_1^2 + \frac{l_1^2}{l_2} - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + \frac{1}{2l_2} \left(3l_2^2 + l_2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) \\ & - \frac{1}{2} \xi_1^2(k) - \frac{1}{2k_2} \xi_2^2(k) - (1 - \alpha_1 \phi^T(k) \phi(k)) (\xi_1(k) + w_1^T \phi(k) \\ & + l_1 e_1(k))^2 - \left(\frac{1}{l_2} \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (\xi_2(k) + w_2^T \sigma(k) + l_2 e_2(k))^2 \right. \\ & \left. + d_1^2(k) + d_2^2(k)/l_2 + 2w_{1\max}^2 \phi_{\max}^2 + w_{2\max}^2 \sigma_{\max}^2/l_2 \right) \end{aligned} \quad (6.A.7)$$

This implies that $\Delta J \leq 0$ as long as (6.23) through (6.26) hold and

$$|e_1(k)| > \frac{2\sqrt{2}g_{1M}D_M}{\sqrt{1 - 20l_1^2g_{1M}^2 - (4g_{1M}^2l_1^2/l_2)}} \quad (6.A.8)$$

or

$$|e_2(k)| > \frac{2\sqrt{2}l_2g_{2M}D_M}{\sqrt{(1 - 4l_2g_{2M}^2 - 20l_2^2g_{2M}^2)}} \quad (6.A.9)$$

or

$$|\xi_1(k)| > \sqrt{2}D_M \quad (6.A.10)$$

or

$$|\zeta_2(k)| > \sqrt{2l_2}D_M \quad (6.A.11)$$

where

$$D_M^2 = d_1^2(k) + \frac{d_2^2(k)}{l_2} + 2w_{1\max}^2\phi_{\max}^2 + \frac{w_{2\max}^2\sigma_{\max}^2}{l_2} \quad (6.A.12)$$

According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the system tracking errors and the weight estimation errors are UUB. The boundedness of $|\zeta_1(k)|$ and $|\zeta_2(k)|$ implies that $\|\tilde{w}_1(k)\|$ and $\|\tilde{w}_2(k)\|$ are bounded and this further implies that the weight estimates $\hat{w}_1(k)$ and $\hat{w}_2(k)$ are bounded. Therefore, all the signals in the closed-loop system are bounded.

Proof of Theorem 6.2.2: Define the Lyapunov function candidate

$$J(k) = \frac{e_1^2(k)}{8g_{1M}^2} + \frac{e_2^2(k)}{8l_2g_{2M}^2} + \sum_{i=1}^3 \frac{1}{\alpha_i} \tilde{w}_i^T(k) \tilde{w}_i(k) \quad (6.A.13)$$

where g_{1M} and g_{2M} are the upper bounds of function $g_1(k)$ and $g_2(k)$, respectively, given a compact set (see Assumption 6.2.1), l_2 , α_1 , α_2 , and α_3 are design parameters (see Theorem 6.2.2).

The first difference of Lyapunov function is given by

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) + \Delta J_5(k) \quad (6.A.14)$$

The first difference $\Delta J_1(k)$ is obtained using Equation 6.12 as

$$\begin{aligned} \Delta J_1(k) &= \frac{1}{8g_{1M}^2}(e_1^2(k+1) - e_1^2(k)) \\ &= \frac{1}{8g_{1M}^2}((g_1(k)(l_1 e_1(k) + e_2(k) + \zeta_1(k) + d_1(k)))^2 - e_1^2(k)) \\ &\leq \frac{1}{2} \left(\left(l_1^2 - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + e_2^2(k) + \zeta_1^2(k) + d_1^2(k) \right) \end{aligned} \quad (6.A.15)$$

Now, taking the second term in the first difference (6.A.13) and substituting (6.18) into (6.A.13) to get

$$\begin{aligned}
 \Delta J_2(k) &= \frac{1}{8k_2 g_{2M}^2} (e_2^2(k+1) - e_2^2(k)) \\
 &= \frac{1}{8k_2 g_{2M}^2} ((g_2(k)(l_1 e_1(k) + l_2 e_2(k) + \zeta_2(k) + d_2(k)))^2 - e_2^2(k)) \\
 &\leq \frac{1}{2} \left(\frac{1}{l_2} \left(l_2^2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) + \frac{\zeta_2^2(k)}{l_2} + \frac{d_2^2(k)}{l_2} + \frac{l_1^2 e_1^2(k)}{l_2} \right)
 \end{aligned} \tag{6.A.16}$$

Taking the third term in (6.A.13) and substituting the weights updates from (6.30) and simplifying to get

$$\begin{aligned}
 \Delta J_3(k) &= \frac{1}{\alpha_1} \tilde{w}_1^T(k+1) \tilde{w}_1(k+1) - \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) \\
 &= \frac{1}{\alpha_1} [(I - \alpha_1 \phi(k) \phi^T(k)) \tilde{w}_1(k) - \alpha_1 \phi(k) (w_1^T(k) \phi(k) \\
 &\quad + l_1 e_1(k) + AR(k))]^T [(I - \alpha_1 \phi(k) \phi^T(k)) \tilde{w}_1(k) \\
 &\quad - \alpha_1 \phi(k) (w_1^T(k) \phi(k) + l_1 e_1(k) + AR(k))] - \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) \\
 &= -(2 - \alpha_1 \phi^T(k) \phi(k)) \zeta_1^2(k) - 2(1 - \alpha_1 \phi^T(k) \phi(k)) (w_1^T \phi(k) \\
 &\quad + l_1 e_1(k) + AR(k)) \zeta_1(k) + \alpha_1 \phi^T(k) \phi(k) (w_1^T \phi(k) \\
 &\quad + l_1 e_1(k) + AR(k))^2
 \end{aligned} \tag{6.A.17}$$

Taking the fourth term in (6.A.13) and substituting the weights updates from (6.31) and simplifying to get

$$\begin{aligned}
\Delta J_4(k) &= \frac{1}{\alpha_2} \tilde{w}_2^T(k+1) \tilde{w}_2(k+1) - \frac{1}{\alpha_2} \tilde{w}_2^T(k) \tilde{w}_2(k) \\
&= \frac{1}{\alpha_2} \left[\left(I - \frac{\alpha_2}{l_2} \sigma(k) \sigma^T(k) \right) \tilde{w}_2(k) - \frac{\alpha_2}{l_2} \sigma(k) (w_2^T \sigma(k) \right. \\
&\quad \left. + l_2 e_2(k) + BR(k)) \right]^T \left[\left(I - \frac{\alpha_2}{l_2} \sigma(k) \sigma^T(k) \right) \tilde{w}_2(k) \right. \\
&\quad \left. - \frac{\alpha_2}{l_2} \sigma(k) (w_2^T \sigma(k) + l_2 e_2(k) + BR(k)) \right] - \frac{1}{\alpha_2} \tilde{w}_2^T(k) \tilde{w}_2(k) \\
&= - \left(\frac{1}{l_2} \right) \left(2 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) \zeta_2^2(k) \\
&\quad - 2 \left(\frac{1}{l_2} \right) \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (w_2^T \sigma(k) + l_2 e_2(k) + BR(k)) \zeta_2(k) \\
&\quad + \frac{\alpha_2}{l_2^2} \sigma^T(k) \sigma(k) (w_2^T \sigma(k) + l_2 e_2(k) + BR(k))^2 \tag{6.A.18}
\end{aligned}$$

Taking the fourth term in (6.A.13) and substituting the weights updates from (6.32) and simplifying to get

$$\begin{aligned}
\Delta J_5(k) &= \frac{1}{\alpha_3} \tilde{w}_3^T(k+1) \tilde{w}_3(k+1) - \frac{1}{\alpha_3} \tilde{w}_3^T(k) \tilde{w}_3(k) \\
&= \frac{1}{\alpha_3} [(I - \alpha_3 \varphi(k) \varphi^T(k)) \tilde{w}_3(k) - \alpha_3 \varphi(k) (w_3^T(k) \varphi(k) + l_1 e_1(k))]^T \\
&\quad \times [(I - \alpha_3 \varphi(k) \varphi^T(k)) \tilde{w}_3(k) - \alpha_3 \varphi(k) (w_3^T(k) \varphi(k) + l_1 e_1(k))] \\
&\quad - \frac{1}{\alpha_3} \tilde{w}_3^T(k) \tilde{w}_3(k) \\
&= -(2 - \alpha_3 \varphi(k)^T \varphi(k)) \zeta_3^2(k) - 2(1 - \alpha_3 \varphi(k)^T \varphi(k)) (w_3^T \varphi(k) \\
&\quad + l_1 e_1(k)) \zeta_3(k) + \alpha_3 \varphi(k)^T \varphi(k) (w_3^T \varphi(k) + l_1 e_1(k))^2 \tag{6.A.19}
\end{aligned}$$

where $\zeta_3(k) = \tilde{w}_3^T(k) \varphi(k)$.

Combining (6.A.15) through (6.A.19) to get the first difference and simplifying to get

$$\begin{aligned}
\Delta J(k) &= \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) + \Delta J_5(k) \\
&\leq \frac{1}{2} \left(l_1^2 + \frac{l_1^2}{l_2} - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + \frac{1}{2l_2} \left(l_2^2 + l_2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) \\
&\quad - \frac{1}{2} \xi_1^2(k) - \frac{1}{2k_2} \xi_2^2(k) - (1 - \alpha_1 \phi^T(k) \phi(k)) (\xi_1^2(k) + 2(w_1^T(k) \phi(k) \\
&\quad + l_1 e_1(k) + AR(k)) \xi_1(k)) - \frac{1}{l_2} \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (\xi_2^2(k) \\
&\quad + 2(w_2^T \sigma(k) + l_2 e_2(k) + BR(k)) \xi_2(k)) - (2 - \alpha_3 \varphi(k)^T \varphi(k)) \xi_3^2(k) \\
&\quad - 2(1 - \alpha_3 \varphi(k)^T \varphi(k)) (w_3^T \varphi(k) + l_1 e_1(k)) \xi_3(k) + \alpha_1 \phi^T(k) \phi(k) \\
&\quad \times (w_1^T \phi(k) + l_1 e_1(k) + AR(k))^2 + d_1^2(k) + \frac{d_2^2(k)}{l_2} + \frac{\alpha_2}{l_2^2} \sigma^T(k) \sigma(k) \\
&\quad \times (w_2^T \sigma(k) + l_2 e_2(k) + BR(k))^2 + \alpha_3 \varphi(k)^T \varphi(k) \\
&\quad \times (w_3^T \varphi(k) + l_1 e_1(k))^2
\end{aligned}$$

$$\begin{aligned}
\Delta J(k) &\leq \frac{1}{2} \left(l_1^2 + \frac{l_1^2}{l_2} - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + \frac{1}{2l_2} \left(l_2^2 + l_2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) - \frac{1}{2} \xi_1^2(k) \\
&\quad - \frac{1}{2k_2} \xi_2^2(k) - (1 - \alpha_1 \phi^T(k) \phi(k)) (\xi_1(k) + w_1^T \phi(k) + l_1 e_1(k) \\
&\quad + AR(k))^2 - \left(\frac{1}{l_2} \right) \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) (\xi_2(k) + w_2^T \sigma(k) \\
&\quad + l_2 e_2(k) + BR(k))^2 - (2 - \alpha_3 \varphi(k)^T \varphi(k)) \xi_3^2(k) - 2(1 - \alpha_3 \varphi(k)^T \\
&\quad \times \varphi(k)) (w_3^T \varphi(k) + l_1 e_1(k)) \xi_3(k) + \alpha_3 \varphi(k)^T \varphi(k) (w_3^T \varphi(k) + l_1 e_1(k))^2 \\
&\quad + (w_1^T \phi(k) + l_1 e_1(k) + AR(k))^2 + \frac{1}{l_2} (w_2^T \sigma(k) + l_2 e_2(k) + BR(k))^2 \\
&\quad + d_1^2(k) + \frac{d_2^2(k)}{l_2}
\end{aligned} \tag{6.A.20}$$

Since

$$\begin{aligned} (w_1^T \phi(k) + l_1 e_1(k) + AR(k))^2 &\leq 3(w_1^T \phi(k) + Aw_3^T \varphi(k))^2 + 3l_1^2 e_1^2(k) \\ &\quad + 3A^2 \zeta_3^2(k) \end{aligned} \quad (6.A.21)$$

and

$$\begin{aligned} \frac{1}{l_2} (w_2^T \sigma(k) + l_2 e_2(k) + BR(k))^2 &\leq \frac{3}{k_2} [(w_2^T \sigma(k) + Bw_3^T \varphi(k))^2 + l_2^2 e_2^2(k) \\ &\quad + B^2 \zeta_3^2(k)] \end{aligned} \quad (6.A.22)$$

and choose

$$0 < 3A^2 + \frac{3B^2}{l_2} \leq \frac{1}{2} \quad (6.A.23)$$

The first difference of the Lyapunov function is expressed as

$$\begin{aligned} \Delta J(k) &\leq \frac{1}{2} \left(11l_1^2 + \frac{l_1^2}{l_2} - \frac{1}{4g_{1M}^2} \right) e_1^2(k) + \frac{1}{2l_2} \left(7l_2^2 + l_2 - \frac{1}{4g_{2M}^2} \right) e_2^2(k) + \\ &\quad - \frac{1}{2} \xi_1^2(k) - \frac{1}{2k_2} \xi_2^2(k) - \frac{1}{2} \xi_3^2(k) - (1 - \alpha_1 \phi^T(k) \phi(k)) (\xi_1(k) \\ &\quad + w_1^T \phi(k) + l_1 e_1(k) + AR(k))^2 \left(\frac{1}{l_2} \right) \left(1 - \frac{\alpha_2}{l_2} \sigma^T(k) \sigma(k) \right) \\ &\quad \times (\xi_2(k) + w_2^T \sigma(k) + l_2 e_2(k) + BR(k))^2 + -(1 - \alpha_3 \varphi(k)^T \varphi(k)) \\ &\quad \times (\xi_3(k) + w_3^T \varphi(k) + l_1 e_1(k))^2 + d_{1M}^2(k) + d_{2M}^2(k)/l_2 \\ &\quad + 6w_1^2 \max \phi^2 + 6w_2^2 \max \sigma^2/l_2 + 4w_3^2 \max \varphi^2 \end{aligned} \quad (6.A.24)$$

This implies that $\Delta J \leq 0$ as long as (6.33) through (6.38) hold and

$$|e_1(k)| > \frac{2\sqrt{2}D_M g_{1M}}{\sqrt{1 - 44l_1^2 g_{1M}^2 - (4g_{1M}^2 l_1^2/l_2)}} \quad (6.A.25)$$

or

$$|e_2(k)| > \frac{2\sqrt{2}l_2 g_{2M} D_M}{\sqrt{(1 - 4l_2 g_{2M}^2 - 28l_2^2 g_{2M}^2)}} \quad (6.A.26)$$

or

$$|\zeta_1(k)| > \sqrt{2}D_M \quad (6.A.27)$$

or

$$|\zeta_2(k)| > \sqrt{2l_2}D_M \quad (6.A.28)$$

or

$$|\zeta_3(k)| > \sqrt{2}D_M \quad (6.A.29)$$

where

$$D_M^2 = d_{1M}^2(k) + d_{2M}^2(k)/l_2 + 6w_{1\max}^2\phi_{\max}^2 + 6w_{2\max}^2\sigma_{\max}^2/l_2 + 4w_{3\max}^2\varphi_{\max}^2 \quad (6.A.30)$$

According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the system tracking errors and the weight estimation errors are UUB. The boundedness of $|\zeta_1(k)|$, $|\zeta_2(k)|$, and $|\zeta_3(k)|$ implies that $\|\tilde{w}_1(k)\|$, $\|\tilde{w}_2(k)\|$, and $\|\tilde{w}_3(k)\|$ are bounded, and this further implies that the weight estimates $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded. Therefore, all the signals in the closed-loop system are bounded.

APPENDIX 6.B

Proof of Theorem 6.3.1: Define the Lyapunov function candidate

$$\begin{aligned} J(k) = & \frac{1}{4}\tilde{x}^T(k-1)\tilde{x}(k-1) + \frac{l_2}{6g_{1m}^2}e_1^2(k) + \frac{1}{6g_{2m}^2}e_2^2(k) + \frac{1}{\alpha_o}\text{tr}(\tilde{w}_o^T(k-1) \\ & \times \tilde{w}_o(k-1)) + \frac{1}{\alpha_1}\tilde{w}_1^T(k)\tilde{w}_1(k) + \frac{1}{\alpha_2}\tilde{w}_2^T(k)\tilde{w}_2(k) \end{aligned} \quad (6.B.1)$$

where $l_2 \in R$ is a design parameter with $l_2 > 0$, g_{1m} and g_{2m} are given in Assumption 6.2.3 and $\alpha_o \in R$, $\alpha_1 \in R$, $\alpha_2 \in R$ are design parameters (see Theorem 6.3.1). The first difference of Lyapunov function is given by

$$\Delta J(k) = \Delta J_1(k) + \Delta J_2(k) + \Delta J_3(k) + \Delta J_4(k) + \Delta J_5(k) + \Delta J_6(k) \quad (6.B.2)$$

The first term, $\Delta J_1(k)$, is obtained using (6.53) as

$$\begin{aligned} \Delta J_1(k) = & \frac{1}{4}\tilde{x}^T(k)\tilde{x}(k) - \frac{1}{4}\tilde{x}^T(k-1)\tilde{x}(k-1) \\ & \leq \frac{1}{2}\|\xi_o(k-1)\|^2 + \frac{1}{2}\|d_o(k-1)\|^2 + \frac{1}{4}\|\tilde{x}(k-1)\|^2 \end{aligned} \quad (6.B.3)$$

Now taking the second term in the first difference (6.B.1) and substituting (6.66) into (6.B.1) to get

$$\begin{aligned}\Delta J_2(k) &= \frac{l_2}{6g_{1m}^2} [e_1^2(k+1) - e_1^2(k)] \\ &\leq \frac{l_2}{2} e_2^2(k) + \frac{l_2}{2} \xi_1^2(k) + \frac{l_2}{2} d_1^2(k) - \frac{l_2 e_1^2(k)}{6g_{1m}^2} \\ &\leq \frac{l_2}{2} e_2^2(k) + \frac{1}{2} \xi_1^2(k) + \frac{1}{2} d_1^2(k) - \frac{l_2 e_1^2(k)}{6g_{1m}^2}\end{aligned}\quad (6.B.4)$$

Taking the third term in (6.B.1) and substituting (6.76) into it and simplifying to get

$$\begin{aligned}\Delta J_3(k) &= \frac{1}{6g_{1m}^2} [e_2^2(k+1) - e_2^2(k)] \\ &\leq \frac{l_1^2}{2} e_1^2(k) + \frac{1}{2} \xi_2^2(k) + \frac{1}{2} d_2^2(k) - \frac{1}{6g_{2m}^2} e_2^2(k)\end{aligned}\quad (6.B.5)$$

Taking the fourth term in (6.B.1) and substituting (6.81) and simplifying to get

$$\begin{aligned}\Delta J_4(k) &= \frac{1}{\alpha_o} \text{tr}(\tilde{w}_o^T(k) \tilde{w}_o(k)) - \frac{1}{\alpha_o} \text{tr}(\tilde{w}_o^T(k-1) \tilde{w}_o(k-1)) \\ &= -(2 - \alpha_o \|\varphi(k-1)\|^2) \|\xi_o(k-1)\|^2 + \alpha_o \|\varphi(k-1)\|^2 \|w_o^T(k-1) \\ &\quad \times \varphi(k-1) + l_1 e_1(k) A\|^2 - (2 - \alpha_o \|\varphi(k-1)\|^2) \xi_o^T(k-1) \\ &\quad \times (w_o^T(k-1) \varphi(k-1) + l_1 e_1(k) A)\end{aligned}\quad (6.B.6)$$

The fifth term $\Delta J_5(k)$ is obtained using weight-updating rule (6.82) as

$$\begin{aligned}\Delta J_5(k) &= \frac{1}{\alpha_1} \tilde{w}_1^T(k+1) \tilde{w}_1(k+1) - \frac{1}{\alpha_1} \tilde{w}_1^T(k) \tilde{w}_1(k) \\ &= -(2 - \alpha_1 \|\phi(k)\|^2) \xi_1^2(k) + \alpha_1 \|\phi(k)\|^2 (w_1^T(k) \phi(k) + l_1 e_1(k))^2 \\ &\quad - 2(1 - \alpha_1 \|\phi(k)\|^2) \xi_1(k) (w_1^T(k) \phi(k) + l_1 e_1(k))\end{aligned}\quad (6.B.7)$$

Using (6.83), the last term $\Delta J_6(k)$ is expressed as

$$\begin{aligned}\Delta J_6(k) &= \frac{1}{\alpha_2}(\tilde{w}_2^T(k+1)\tilde{w}_2(k+1)) - \frac{1}{\alpha_2}\tilde{w}_2^T(k)\tilde{w}_2(k) \\ &= -(2 - \alpha_2\|\sigma(k)\|^2)\xi_2^2(k) + \alpha_2\|\sigma(k)\|^2(w_2^T(k)\sigma(k) + l_1e_1(k))^2 \\ &\quad - 2(1 - \alpha_2\|\sigma(k)\|^2)\xi_2(k)(w_2^T(k)\sigma(k) + l_1e_1(k))\end{aligned}\quad (6.B.8)$$

Combining (6.B.3) through (6.B.8) to get the first difference and simplifying to get

$$\begin{aligned}\Delta J(k) &= \sum_{i=1}^6 \Delta J_i(k) \\ &= \frac{1}{4}\|\tilde{x}(k-1)\|^2 - \left(\frac{l_2}{6g_{1m}^2} - \frac{13}{2}l_1^2\right)e_1^2(k) - \left(\frac{l_2}{6g_{2m}^2} - \frac{l_2}{2}\right)e_2^2(k) \\ &\quad - \frac{1}{2}\|\xi_o(k-1)\|^2 - \frac{1}{2}\xi_1^2(k) - \frac{1}{2}\xi_2^2(k) + D_M^2 \\ &\quad - (1 - \alpha_0\|\varphi(k-1)\|^2)\|\xi_o(k) - (w_o^T(k-1)\varphi(k-1) + l_1e_1(k)A)\|^2 \\ &\quad - (1 - \alpha_1\|\phi(k)\|^2)(\xi_1(k) - (w_1^T(k)\phi(k) + l_1e_1(k)))^2 \\ &\quad - (1 - \alpha_2\|\sigma(k)\|^2)(\xi_2(k) - (w_2^T(k)\sigma(k) + l_1e_1(k)))^2\end{aligned}\quad (6.B.9)$$

where

$$\begin{aligned}D_M^2 &= \frac{1}{2}\|d_o(k-1)\|^2 - \frac{1}{2}d_1^2(k) - \frac{1}{2}d_2^2(k) + 2w_o^T(k-1)\varphi(k-1) \\ &\quad + 2w_1^T(k)\phi(k) + 2w_2^T(k)\sigma(k)\end{aligned}\quad (6.B.10)$$

This implies that $\Delta J \leq 0$ as long as (6.84) through (6.87) along with the following condition

$$0 < l_2 < \frac{1}{3g_{2m}^2}\quad (6.B.11)$$

and

$$\|\tilde{x}(k-1)\| > 2D_M\quad (6.B.12)$$

or

$$|e_1(k)| > \frac{D_M}{\sqrt{(l_2/6g_{1m}^2) - ((13/2)l_1^2)}} \quad (6.B.13)$$

or

$$|e_2(k)| > \frac{D_M}{\sqrt{(1/6g_{2m}^2) - (l_2/2)}} \quad (6.B.14)$$

or

$$\|\xi_o(k-1)\| > \sqrt{2}D_M \quad (6.B.15)$$

or

$$|\xi_1(k)| > \sqrt{2}D_M \quad (6.B.16)$$

or

$$|\xi_2(k)| > \sqrt{2}D_M \quad (6.B.17)$$

According to the standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that $x(k-1)$, $e_1(k)$, $e_2(k)$, and the weight estimation errors are UUB. The boundedness of $\xi_o(k-1)$, $\xi_1(k)$, and $\xi_2(k)$ implies that $\|\tilde{w}_o(k)\|$, $\|\tilde{w}_1(k)\|$, and $\|\tilde{w}_2(k)\|$ are bounded and this further implies that the weight estimates $\hat{w}_o(k)$, $\hat{w}_1(k)$, and $\hat{w}_2(k)$ are bounded. Therefore, all the closed-loop signals in the observer–controller system are bounded.

7 System Identification Using Discrete-Time Neural Networks

System identification is the process of determining a dynamic model for an unknown system that can subsequently be used for feedback control purposes. On the other hand, state estimation involves determining the unknown internal states of a dynamic system. System identification provides one technique for estimating the states. The area of system identification has received significant attention over the past decades and now it is a fairly mature field with many powerful methods available at the disposal of control engineers. Online system identification methods to date are based on recursive methods such as least squares, for most systems that are expressed as linear in the parameters (LIP). To overcome this LIP assumption, neural networks (NNs) are now employed for system identification since these networks learn complex mappings from a set of examples. As seen in the previous chapters, due to NN approximation properties (Cybenko 1989) as well as the inherent adaptation features of these networks, NN present a potentially appealing alternative to modeling of nonlinear systems. Moreover, from a practical perspective, the massive parallelism and fast adaptability of NN implementations provide additional incentives for further investigation.

Several approaches have been presented for system identification without using NN (Landau 1979; Ljung and Soderstrom 1983; Goodwin and Sin 1984; Narendra and Annaswamy 1989) and using NN (Narendra and Parthasarathy 1990; Jagannathan and Lewis 1996). Most of the development is done in continuous time due to the simplicity of deriving adaptation schemes. To the contrary, very few results are available for the system identification in discrete-time using NNs. However, most of the schemes for system identification using NN have been demonstrated through empirical studies, or convergence of the output error is shown under ideal conditions (Narendra and Parthasarathy 1990). Others (Sadegh 1993) have shown the stability of the overall

system or convergence of the output error using linearity in the parameters assumption.

Both recurrent and dynamic NN, in which the NN has its own dynamics (Narendra and Parthasarathy 1990), have been utilized for system identification. Most identification schemes using either multilayer feedforward or recurrent NN include identifier structures which do not guarantee the boundedness of the identification error of the system under nonideal conditions even in the open-loop configuration. In addition, convergence results if at all any are only given under some stringent conditions such as the initialization of the NN with stabilizing weights in the neighborhood of the global minimum, which is a very unrealistic assumption since it is very hard to find the stabilizing weights. With improper weight initialization, many authors report undesirable behavior. Furthermore, the backpropagation algorithm, often used for system identification, requires the evaluation of sensitivity derivatives along the network signal paths, which is usually impossible in closed-loop uncertain systems since the required Jacobians are unknown.

The main objective of this chapter is to provide techniques for estimating the internal states of unknown dynamical systems using dynamical NN (Jagannathan and Lewis 1996). This is achieved by first identifying the unknown system dynamics. It is very important to note that solving the state estimation problem involves only a small subset of the topic of system identification. In order to relax the linear in the unknown parameter assumption and show the boundedness of the state estimation errors using multilayer NN, novel learning schemes are necessary for weight tuning to identify four classes of discrete-time nonlinear systems that are commonly used in the literature. Here, the NN weights are tuned online with no preliminary off-line learning phase needed. The weight-tuning mechanisms guarantee convergence of the NN weights when initialized at zero, even though there do not exist target weights such that an NN can perfectly reconstruct a function that approximates the desired nonlinear system. The identifier structure ensures good performance (bounded identification error and weight estimates) as shown through the Lyapunov's approach, so that convergence to a stable solution is guaranteed with mild assumptions. Extension of this approach to closed-loop scenarios is rather straightforward, but is not necessary in the case of identification of systems alone.

The identifier is composed of an NN incorporated into a dynamical system, where the structure comes from error notions standard in the system identification and control literature. It is shown that the delta rule in each layer yields a passive NN (Jagannathan 1994; Jagannathan and Lewis 1996); this guarantees the boundedness of all the signals in the system. The convergence analysis using a three-layer NN is extended to a general n -layer case. For more details see Jagannathan (1994).

Once an NN has been tuned to identify a dynamical system, it is of great interest to determine the structural information contained in the NN by using the learned NN weights. Structural information can be very useful in controller design. This can be accomplished in many ways, including the Volterra series approach in the work of Billings and coworkers (Billings et al. 1992; Fung et al. 1997), which determines a generalized frequency response function (GFRF) of a given NN.

7.1 IDENTIFICATION OF NONLINEAR DYNAMICAL SYSTEMS

The ability of NN to approximate large classes of nonlinear functions makes them prime candidates for the identification of nonlinear systems. Four models representing multi-input/multi-output (MIMO) nonlinear systems are in common use (Landau 1979, 1993; Narendra and Parthsarathy 1990). These four models are in nonlinear autoregressive moving average (NARMA) form, and cover a very large range of systems. They are therefore considered here. These four nonlinear canonical forms are

$$x(k+1) = \sum_{i=0}^{n-1} \alpha_i x(k-i) + g(u(k), u(k-1), \dots, u(k-m+1)) + d(k) \quad (7.1)$$

$$x(k+1) = f(x(k), x(k-1), \dots, x(k-n+1)) + \sum_{i=0}^{n-1} \beta_i u(k-i) + d(k) \quad (7.2)$$

$$\begin{aligned} x(k+1) &= f(x(k), x(k-1), \dots, x(k-n+1)) + g(u(k), u(k-1), \dots, \\ &\quad u(k-m+1)) + d(k) \end{aligned} \quad (7.3)$$

$$\begin{aligned} x(k+1) &= f(x(k), x(k-1), \dots, x(k-n+1); u(k), u(k-1), \dots, \\ &\quad u(k-m+1)) + d(k) \end{aligned} \quad (7.4)$$

with unknown nonlinear functions $f(\cdot) \in \Re^n$, $g(\cdot) \in \Re^n$, state $x(k) \in \Re^n$, coefficients $\alpha_i \in \Re^{n \times n}$, $\beta_i \in \Re^{n \times n}$, control $u(k) \in \Re^n$, and disturbance $d(k) \in \Re^n$ an unknown vector acting on the system at the instant with $\|d(k)\| \leq d_M$ a known constant. When Model I is selected then $\alpha_i \in \Re^{n \times n}$ are chosen such that the roots of the polynomial $z^n - \alpha_0 z^{n-1} - \dots - \alpha_{n-1} = 0$ lie in the

interior of the unit disc. The four models are shown graphically in Figure 7.1. The next step will be to select suitable identifier models for the MIMO systems.

7.2 IDENTIFIER DYNAMICS FOR MIMO SYSTEMS

Consider MIMO discrete-time nonlinear systems given in multivariable form as one of the models (7.1) through (7.4). The problem of identification consists of setting up a suitably parameterized identification model and adjusting the parameters of the model so that when subjected to the same input $u(k)$ as

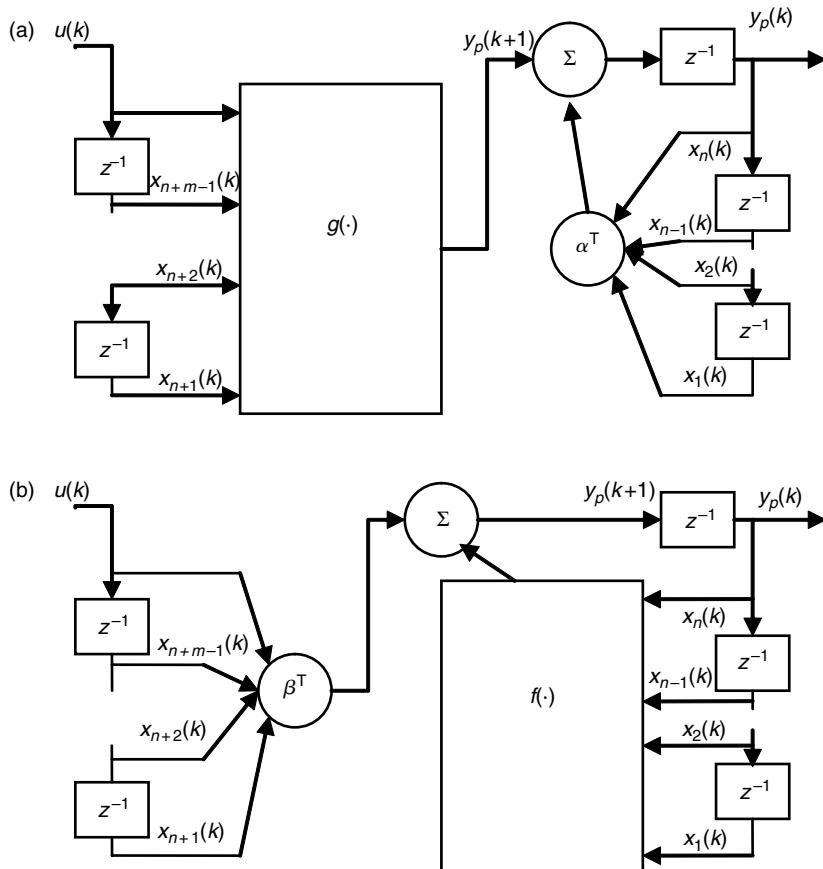
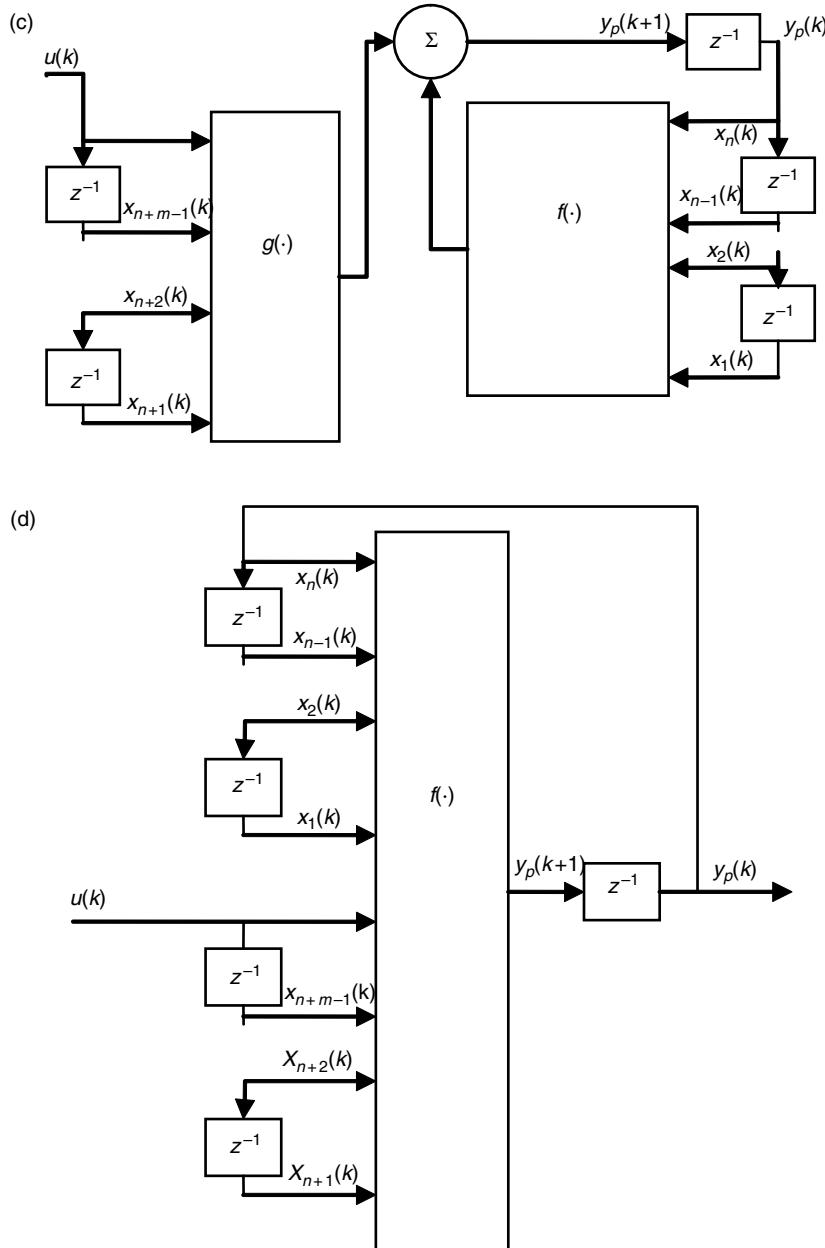


FIGURE 7.1 Multilayer NN identifier models.

**FIGURE 7.1** Continued.

the system, it produces an output $\hat{x}(k)$ that is close to the actual $x(k)$. Taking the structure of the identifier same as that of the system, the systems given in (7.1) through (7.4) are identified, respectively, by the following estimators

$$x(k+1) = \sum_{i=0}^{n-1} \alpha_i x(k-i) + \hat{g}(u(k), u(k-1), \dots, u(k-m+1)) + d(k) \quad (7.5)$$

$$x(k+1) = \hat{f}(x(k), x(k-1), \dots, x(k-n+1)) + \sum_{i=0}^{n-1} \beta_i u(k-i) + d(k) \quad (7.6)$$

$$x(k+1) = \hat{f}(x(k), x(k-1), \dots, x(k-n+1)) + \hat{g}(u(k), u(k-1), \dots, u(k-m+1)) + d(k) \quad (7.7)$$

$$x(k+1) = \hat{f}(x(k), x(k-1), \dots, x(k-n+1); u(k), u(k-1), \dots, u(k-m+1)) + d(k) \quad (7.8)$$

where $\hat{f}(\cdot)$ is an estimate of $f(\cdot)$ and $\hat{g}(\cdot)$ an estimate of $g(\cdot)$.

In this work, NN will be employed to provide the estimate of $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$. Due to the universal approximation properties, there exist static NNs that approximate $f(\cdot)$ and $g(\cdot)$. When embedded in the dynamics (7.5) through (7.8), the result is a dynamic or recurrent NN estimator that, for the same initial conditions, produces the same output as the plant for any specified input. The identification procedure consists of adjusting the weights of the NN in the model using the weight updates presented in Section 7.3 to guarantee internal stability and closeness of $\hat{x}(\cdot)$ with $x(k)$.

Define the identification error as

$$e(k) = x(k) - \hat{x}(k) \quad (7.9)$$

Then the error dynamics of (7.1) through (7.4) and (7.9) can be expressed, respectively, as

$$e(k+1) = \tilde{g}(\cdot) + d(k) \quad (7.10)$$

$$e(k+1) = \tilde{f}(\cdot) + d(k) \quad (7.11)$$

$$e(k+1) = \tilde{f}(\cdot) + \tilde{g}(\cdot) + d(k) \quad (7.12)$$

$$e(k+1) = \tilde{f}(\cdot) + d(k) \quad (7.13)$$

where the functional estimation errors are given by

$$\tilde{f}(\cdot) = f(\cdot) - \hat{f}(\cdot) \quad (7.14)$$

and

$$\tilde{g}(\cdot) = g(\cdot) - \hat{g}(\cdot) \quad (7.15)$$

These are error systems wherein the identification error is driven by the functional estimation error.

In the remainder of this chapter, Equation 7.10 through Equation 7.13 are utilized to focus on selecting suitable NN-tuning schemes that guarantee the stability of the identification error $e(k)$. It is important to note that (7.11) and (7.13) are similar except the nonlinear function in (7.13) is a more general function of the state vector, the input vector, and their delayed values. Denote by $\tilde{x}(k)$ the appropriate argument of $\tilde{f}(\cdot)$, which consists of $x(k)$ and previous values in (7.11) and also includes $u(k)$ and previous values in (7.13). Then both equations can be represented as

$$e(k + 1) = \tilde{f}(\tilde{x}(k)) + d(k) \quad (7.16)$$

This is the error system resulting from either identifier (7.6) or (7.8).

Equation 7.10 and Equation 7.12 are also similar except that $\tilde{f}(\cdot)$ is missing in the former. For analysis purposes, they are both taken as the more general system

$$e(k + 1) = \tilde{f}(\tilde{x}(k)) + g(\tilde{u}(k)) + d(k) \quad (7.17)$$

where $\tilde{u}(k)$ denotes $u(k)$ and its previous values. This is the error system resulting from either (7.5) or (7.7). Subsequent analysis considers these two forms of error system.

7.3 NN IDENTIFIER DESIGN

In this section, multilayer NN are used to provide the approximations $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ in the identifier systems. Stability analysis is performed by Lyapunov's direct method for multilayer NN weight-tuning schemes consisting of a delta rule in each layer. Note that one NN is used to approximate $\hat{f}(\cdot)$ in the error system (7.16) whereas two NN are required, one for $\hat{f}(\cdot)$ and one for $\hat{g}(\cdot)$, for (7.17).

Assume that there exist some constant weights W_{1f}, W_{2f}, W_{3f} and W_{1g}, W_{2g}, W_{3g} for three-layer NN so that the nonlinear functions $f(\cdot)$ in (7.16)

and (7.17) and $g(\cdot)$ in (7.17) can be written on a compact set S as

$$f(\tilde{x}(k)) = W_{3f}^T \phi_{3f}(W_{2f}^T \phi_{2f}(W_{1f}^T \phi_{1f}(\tilde{x}(k)))) + \varepsilon_f(k) \quad (7.18)$$

$$g(\tilde{u}(k)) = W_{3g}^T \phi_{3g}(W_{2g}^T \phi_{2g}(W_{1g}^T \phi_{1g}(\tilde{u}(k)))) + \varepsilon_g(k) \quad (7.19)$$

where the functional estimation errors satisfy $\|\varepsilon_f(k)\| \leq \varepsilon_{Nf}$ and $\|\varepsilon_g(k)\| \leq \varepsilon_{Ng}$, with the bounding constants ε_{Nf} and ε_{Ng} known. Unless the network is minimal, the ideal weights may not be unique (Sussmann 1992). The best weights may then be defined as those which minimize the supremum norm of S of $\varepsilon(k)$. This issue is not a major concern here, as it is needed to know only existence of target weights; their actual values are not needed. This assumption is similar to Erzberger's assumption (Erzberger 1968) in LIP adaptive control and multilayer NN are employed.

Assumption 7.3.1 (Bounded NN Weights): The ideal weights are bounded by known positive values so that $\|W_{1f}\| \leq W_{1f \max}$, $\|W_{2f}\| \leq W_{2f \max}$, and $\|W_{3f}\| \leq W_{3f \max}$. Similarly, $\|W_{1g}\| \leq W_{1g \max}$, $\|W_{2g}\| \leq W_{2g \max}$, and $\|W_{3g}\| \leq W_{3g \max}$.

7.3.1 STRUCTURE OF THE NN IDENTIFIER AND ERROR SYSTEM DYNAMICS

Define the NN functional estimate for $f(\cdot)$ and $g(\cdot)$ by

$$\hat{f}(\tilde{x}(k)) = \hat{W}_{3f}^T(k) \phi_{3f}(\hat{W}_{2f}^T(k) \phi_{2f}(\hat{W}_{1f}^T(k) \phi_{1f}(\tilde{x}(k)))) + \varepsilon_f(k) \quad (7.20)$$

$$\hat{g}(\tilde{u}(k)) = \hat{W}_{3g}^T(k) \phi_{3g}(\hat{W}_{2g}^T(k) \phi_{2g}(\hat{W}_{1g}^T(k) \phi_{1g}(\tilde{u}(k)))) + \varepsilon_g(k) \quad (7.21)$$

where \hat{W}_{1f} , \hat{W}_{2f} , \hat{W}_{3f} and \hat{W}_{1g} , \hat{W}_{2g} , \hat{W}_{3g} are the current values of the weights as given by the tuning algorithms to be derived. The estimates of the input-layer activation function outputs are denoted by $\hat{\phi}_{1f}(k) = \phi_{1f}(\tilde{x}(k))$ and $\hat{\phi}_{1g}(k) = \phi_{1g}(\tilde{u}(k))$. Then the estimates of the activation function outputs of the hidden and output layers are denoted by

$$\hat{\phi}_{(i+1)f}(k) = \phi(\hat{W}_{if}^T \hat{\phi}_{if}(k)) \quad i = 1, \dots, n \quad (7.22)$$

$$\hat{\phi}_{(i+1)g}(k) = \phi(\hat{W}_{ig}^T \hat{\phi}_{ig}(k)) \quad i = 1, \dots, n \quad (7.23)$$

where $n = 3$.

Since the standard NN activation functions, including sigmoids, tanh, RBF, etc., are bounded by known positive values for a given trajectory, one has

$$\begin{aligned}\|\hat{\phi}_{1f}(k)\| &\leq \phi_{1f} \max, & \|\hat{\phi}_{2f}(k)\| &\leq \phi_{2f} \max, & \|\hat{\phi}_{3f}(k)\| &\leq \phi_{3f} \max \\ \|\hat{\phi}_{1g}(k)\| &\leq \phi_{1g} \max, & \|\hat{\phi}_{2g}(k)\| &\leq \phi_{2g} \max, & \text{and} & \|\hat{\phi}_{3g}(k)\| \leq \phi_{3g} \max\end{aligned}$$

The NN weight estimation errors are given by

$$\tilde{W}_{1f}(k) = W_{1f} - \hat{W}_{1f}(k), \quad \tilde{W}_{2f}(k) = W_{2f} - \hat{W}_{2f}(k), \quad \tilde{W}_{3f}(k) = W_{3f} - \hat{W}_{3f}(k) \quad (7.24)$$

and

$$\tilde{W}_{1f}(k) = W_{1f} - \hat{W}_{1f}(k), \quad \tilde{W}_{2f}(k) = W_{2f} - \hat{W}_{2f}(k), \quad \tilde{W}_{3f}(k) = W_{3f} - \hat{W}_{3f}(k) \quad (7.25)$$

The network layer output errors are defined as

$$\tilde{\phi}_{2f}(k) = \phi_{2f} - \hat{\phi}_{2f}(k), \quad \tilde{\phi}_{3f}(k) = \phi_{3f} - \hat{\phi}_{3f}(k) \quad (7.26)$$

and

$$\tilde{\phi}_{2g}(k) = \phi_{2g} - \hat{\phi}_{2g}(k), \quad \tilde{\phi}_{3g}(k) = \phi_{3g} - \hat{\phi}_{3g}(k) \quad (7.27)$$

Using the functional estimate of $f(\cdot)$ and $g(\cdot)$ in (7.20) and (7.21), the error Equation 7.16 and Equation 7.17 can be expressed as

$$e(k+1) = e_f(k) + \delta(k) \quad (7.28)$$

and

$$e(k+1) = e_f(k) + e_g(k) + \delta(k) \quad (7.29)$$

where one defines

$$e_f(k) \equiv \tilde{W}_{3f}^T(k) \hat{\phi}_{3f}(k) \quad (7.30)$$

$$e_g(k) \equiv \tilde{W}_{3g}^T(k) \hat{\phi}_{3g}(k) \quad (7.31)$$

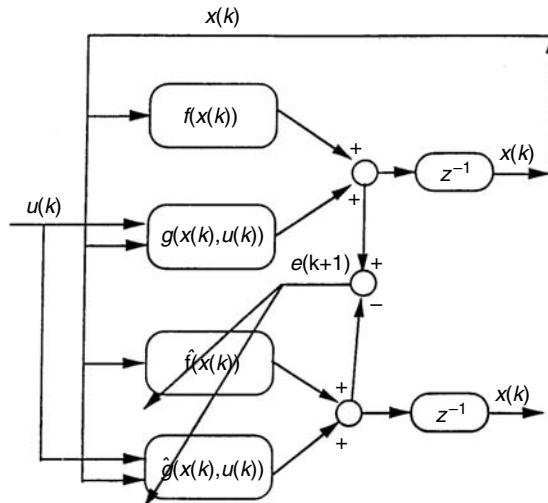


FIGURE 7.2 Multilayer NN identifier structure.

and

$$\delta(k) \equiv W_{3f}^T \tilde{\phi}_{3f}(k) + \varepsilon_f(k) + d(k) \quad (7.32)$$

$$\delta(k) \equiv W_{3f}^T \tilde{\phi}_{3f}(k) + W_{3g}^T \tilde{\phi}_{3g}(k) + \varepsilon_f(k) + \varepsilon_g(k) + d(k) \quad (7.33)$$

The proposed identifier structure is shown in Figure 7.2. The next step is to determine the weight updates so that the tracking performance of the identification error dynamics is guaranteed.

7.3.2 MULTILAYER NN WEIGHT UPDATES

Novel weight-tuning schemes that guarantee the stability of the error systems (7.28) and (7.29) are presented in this section. It is required to demonstrate that the identification error $e(k)$ is suitably small and that the NN weight estimates in (7.20) and (7.21) remain bounded, given a bounded input $u(k)$. The next result provides NN weight-tuning schemes that guarantee stable identification. Persistence of identification (PE) for a multilayer discrete-time NN (Jagannathan and Lewis 1996) is defined in the proof.

Theorem 7.3.1 (Three-Layer NN Identifier): Given an unknown system in one of the four forms (7.1) through (7.4), select the estimator from the respective form (7.5) through (7.8) and let $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ if required, be given by NN as in

(7.20) and (7.21). Let the NN functional reconstruction error and the disturbance bounds, $\varepsilon_{Nf}, \varepsilon_{Ng}, d_M$, respectively, be known constants. Let NN weight tuning be provided for the input and hidden layers as

$$\hat{W}_{1f}(k+1) = \hat{W}_{1f}(k) - \alpha_{1f}\hat{\phi}_{1f}(k)[\hat{y}_{1f}(k) + B_{1f}e(k)]^T \quad (7.34)$$

$$\hat{W}_{2f}(k+1) = \hat{W}_{2f}(k) - \alpha_{2f}\hat{\phi}_{2f}(k)[\hat{y}_{2f}(k) + B_{2f}e(k)]^T \quad (7.35)$$

$$\hat{W}_{1g}(k+1) = \hat{W}_{1g}(k) - \alpha_{1g}\hat{\phi}_{1g}(k)[\hat{y}_{1g}(k) + B_{1g}e(k)]^T \quad (7.36)$$

$$\hat{W}_{2g}(k+1) = \hat{W}_{2g}(k) - \alpha_{2g}\hat{\phi}_{2g}(k)[\hat{y}_{2g}(k) + B_{2g}e(k)]^T \quad (7.37)$$

where $\hat{y}_{if}(k) = \hat{W}_{if}^T(k)\hat{\phi}_{if}(k)$; $\hat{y}_{ig}(k) = \hat{W}_{ig}^T(k)\hat{\phi}_{ig}(k)$; $i = 1, 2$, and

$$\|B_{if}\| \leq \kappa_{if} \quad i = 1, 2 \quad (7.38)$$

$$\|B_{ig}\| \leq \kappa_{ig} \quad i = 1, 2 \quad (7.39)$$

Let the weight tuning for the output layer be given by

$$\hat{W}_{3f}(k+1) = \hat{W}_{3f}(k) + \alpha_{3f}\hat{\phi}_{3f}(k)e^T(k+1) \quad (7.40)$$

$$\hat{W}_{3g}(k+1) = \hat{W}_{3g}(k) + \alpha_{3g}\hat{\phi}_{3g}(k)e^T(k+1) \quad (7.41)$$

with $\alpha_{if} > 0, \alpha_{ig} > 0, i = 1, 2, 3$ denoting constant learning rate parameters or adaptation gains.

Let output vectors of the input, hidden, and output layers, $\hat{\phi}_{1f}(k), \hat{\phi}_{2f}(k), \hat{\phi}_{3f}(k), \hat{\phi}_{1g}(k), \hat{\phi}_{2g}(k)$, and $\hat{\phi}_{3g}(k)$ be persistently exciting. Then the identification error, $e(k)$, and the errors in the weight estimates, $\tilde{W}_{1f}, \tilde{W}_{2f}, \tilde{W}_{3f}$ and $\tilde{W}_{1g}, \tilde{W}_{2g}, \tilde{W}_{3g}$ or weight estimates, $\hat{W}_{1f}, \hat{W}_{2f}, \hat{W}_{3f}$ and $\hat{W}_{1g}, \hat{W}_{2g}, \hat{W}_{3g}$, are uniformly ultimately bounded (UUB), with the bounds on $e(k)$ specifically given by (7.54), provided the following conditions hold:

Condition (a):

$$\alpha_{if}\|\hat{\phi}_{if}(k)\| < \begin{cases} 2, & i = 1, 2 \\ 1, & i = 3 \end{cases} \quad (7.42)$$

for the error system (7.28) and (7.42) plus

$$\begin{aligned} \alpha_{ig}\|\hat{\phi}_{ig}(k)\|^2 &< 2 \quad i = 1, 2 \\ (\alpha_{if}\|\hat{\phi}_{if}(k)\|^2 + \alpha_{ig}\|\hat{\phi}_{ig}(k)\|^2) &< 1 \end{aligned} \quad (7.43)$$

for the error system (7.29).

Condition (b):

$$c_0 < 1 \quad (7.44)$$

where c_0 is given for the error system (7.28) as

$$c_0 = \sum_{i=1}^2 \frac{\kappa_{if}^2}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} \quad (7.45)$$

and for the error system (7.29) as

$$c_0 = \sum_{i=1}^2 \frac{\kappa_{if}^2}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} + \sum_{i=1}^2 \frac{\kappa_{ig}^2}{(2 - \alpha_{ig} \|\hat{\phi}_{ig}(k)\|^2)} \quad (7.46)$$

Note: The parameters α_{if} , α_{ig} ; $\forall i = 1, \dots, 3$ and c_0 depend upon the trajectory. Given a trajectory, the constants α_{if} , α_{ig} ; $\forall i = 1, \dots, 3$ and c_0 , c_1 , and c_2 can be determined.

Proof: Let Ω and Ω_U be subsets of \Re^n and \Re^m , respectively, such that $e(0) \in \Omega$ and $\tilde{W}_i(0) \in \Omega_U$; $i = 1, 2, 3$ and the NN approximation holds. Using the Lyapunov function candidate

$$J = e^T(k)e(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} \right] \quad (7.47)$$

define

$$l^2 = \sup_{(e, \tilde{W}_i) \in \Omega \times \Omega_U} \left\{ e^T(k)e(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} \right] \right\} \quad (7.48)$$

Consider ΔJ on the set $\chi = [(e, \tilde{W}_i) : J(e, \tilde{W}_i) \leq l^2]$

Error system (7.28): Define the Lyapunov function candidate as in (7.47) and whose first difference, $\Delta J(e, \tilde{W}_i) \in \chi$, is given by

$$\begin{aligned} \Delta J &= e^T(k+1)e(k+1) - e^T(k)e(k) \\ &\quad + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k+1)\tilde{W}_{if}(k+1) - \tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} \right] \end{aligned} \quad (7.49)$$

Substituting the Equation 7.34, Equation 7.35, and Equation 7.40 yields,

$$\begin{aligned} \Delta J &\leq -(1 - c_0) \left[\|e(k)\|^2 - 2 \frac{c_1}{(1 - c_0)} \|e(k)\| - \frac{c_1}{(1 - c_0)} \right] \\ &\quad - [1 - \alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k)] \left\| e_f(k) - \frac{\alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k) \delta(k)}{[1 - \alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k)]} \right\|^2 \\ &\quad - \sum_{i=1}^n (2 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k)) \left\| \tilde{W}_{if}^T(k) \hat{\phi}_{if}(k) - \frac{(1 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k))}{[2 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k)]} \right. \\ &\quad \times \left. (W_{if}^T(k) \hat{\phi}_{if}(k) + B_{if} e(k)) \right\|^2 \end{aligned} \quad (7.50)$$

where

$$c_1 = \sum_{i=1}^2 \frac{\kappa_{if} \|\hat{\phi}_{if}(k)\| W_{if} \max}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} \quad (7.51)$$

and

$$c_2 = \frac{\delta_{\max}^2}{(1 - (\alpha_{3f} \|\hat{\phi}_{3f}(k)\|^2 + \alpha_{3g} \|\hat{\phi}_{3g}(k)\|^2))} + \sum_{i=1}^2 \frac{\|\hat{\phi}_{if}(k)\|^2 W_{if}^2 \max}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} \quad (7.52)$$

with

$$\delta_{\max} = W_{3f} \max \tilde{\phi}_{3f} \max + \varepsilon_{Nf} \max + d_M \quad (7.53)$$

Since c_0 , c_1 , c_2 are positive constants, $\Delta J \leq 0$ as long as

$$\|e(k)\| > \frac{1}{(1 - c_0)} \left[c_1 + \sqrt{c_1^2 + c_2(1 - c_0)} \right] \quad (7.54)$$

$|\sum_{k=k_0}^{\infty} \Delta J| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (7.42), (7.44), and (7.54) hold. The definition of J and inequality (7.54) imply that every initial condition is in the set χ . In other words, whenever the identification

error $e(k)$ is outside the region defined by (7.54), $J(e, \tilde{W}_i)$ will decrease. This further implies that $\|e(k)\|$ will not increase and will remain χ . This demonstrates that the identification error $e(k)$ is bounded for all $k \geq 0$ and it remains to show that the weight estimates $\hat{W}_{1f}(k)$, $\hat{W}_{2f}(k)$, $\hat{W}_{3f}(k)$, $\hat{W}_{1g}(k)$, $\hat{W}_{2g}(k)$, and $\hat{W}_{3g}(k)$ or equivalently $\tilde{W}_{1f}(k)$, $\tilde{W}_{2f}(k)$, $\tilde{W}_{3f}(k)$, $\tilde{W}_{1g}(k)$, $\tilde{W}_{2g}(k)$, and $\tilde{W}_{3g}(k)$.

Error system (7.29): Let Ω and Ω_U be subsets of \Re^n and \Re^m , respectively, such that $e(0) \in \Omega$ and $\tilde{W}_i(0) \in \Omega_U$; $i = 1, 2, 3$ for both $f(\cdot)$ and $g(\cdot)$ (here $m = 6$) and the NN approximation holds. Using the Lyapunov function candidate

$$J = e^T(k)e(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} + \frac{1}{\alpha_{ig}} \text{tr}\{\tilde{W}_{ig}^T(k)\tilde{W}_{ig}(k)\} \right] \quad (7.55)$$

define

$$\begin{aligned} l^2 = \sup_{(e, \tilde{W}_i) \in \Omega \times \Omega_U} & \left\{ e^T(k)e(k) + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} \right. \right. \\ & \left. \left. + \frac{1}{\alpha_{ig}} \text{tr}\{\tilde{W}_{ig}^T(k)\tilde{W}_{ig}(k)\} \right] \right\} \end{aligned} \quad (7.56)$$

Consider ΔJ on the set $\chi = [(e, \tilde{W}_i) : J(e, \tilde{W}_i) \leq l^2]$.

Define the Lyapunov function candidate as in (7.55) and whose first difference, $\Delta J(e, \tilde{W}_i) \in \chi$, is given by

$$\begin{aligned} \Delta J = & e^T(k+1)e(k+1) - e^T(k)e(k) \\ & + \sum_{i=1}^3 \left[\frac{1}{\alpha_{if}} \text{tr}\{\tilde{W}_{if}^T(k+1)\tilde{W}_{if}(k+1) - \tilde{W}_{if}^T(k)\tilde{W}_{if}(k)\} \right] \\ & + \sum_{i=1}^3 \left[\frac{1}{\alpha_{ig}} \text{tr}\{\tilde{W}_{ig}^T(k+1)\tilde{W}_{ig}(k+1) - \tilde{W}_{ig}^T(k)\tilde{W}_{ig}(k)\} \right] \end{aligned} \quad (7.57)$$

Substituting (7.34) through (7.41) in (7.57), one may obtain

$$\begin{aligned}
 \Delta J \leq & -(1 - c_0) \left[\|e(k)\|^2 - 2 \frac{c_1}{(1 - c_0)} \|e(k)\| - \frac{c_1}{(1 - c_0)} \right] \\
 & - [1 - (\alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k) + \alpha_{3g} \hat{\phi}_{3g}^T(k) \hat{\phi}_{3g}(k))] \\
 & \times \left\| (e_f(k) + e_g(k)) - \frac{(\alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k) + \alpha_{3g} \hat{\phi}_{3g}^T(k) \hat{\phi}_{3g}(k)) \delta(k)}{[1 - (\alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k) + \alpha_{3g} \hat{\phi}_{3g}^T(k) \hat{\phi}_{3g}(k))]} \right\|^2 \\
 & - \sum_{i=1}^n (2 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k)) \left\| \tilde{W}_{if}^T(k) \hat{\phi}_{if}(k) - \frac{(1 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k))}{[2 - \alpha_{if} \hat{\phi}_{if}^T(k) \hat{\phi}_{if}(k)]} \right. \\
 & \times (W_{if}^T(k) \hat{\phi}_{if}(k) + B_{if} e(k)) \left. \right\|^2 - \sum_{i=1}^2 (2 - \alpha_{ig} \hat{\phi}_{ig}^T(k) \hat{\phi}_{ig}(k)) \\
 & \times \left\| \tilde{W}_{ig}^T(k) \hat{\phi}_{ig}(k) - \frac{(1 - \alpha_{ig} \hat{\phi}_{ig}^T(k) \hat{\phi}_{ig}(k))}{[2 - \alpha_{ig} \hat{\phi}_{ig}^T(k) \hat{\phi}_{ig}(k)]} (W_{ig}^T(k) \hat{\phi}_{ig}(k) + B_{ig} e(k)) \right\|^2
 \end{aligned} \tag{7.58}$$

where

$$\begin{aligned}
 c_1 = & \frac{\delta_{\max}}{[1 - (\alpha_{if} \|\hat{\phi}_{if}(k)\|^2 + \alpha_{ig} \|\hat{\phi}_{ig}(k)\|^2)]} + \sum_{i=1}^2 \frac{\kappa_{if} \|\hat{\phi}_{if}(k)\| W_{if \max}}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} \\
 & + \sum_{i=1}^2 \frac{\kappa_{ig} \|\hat{\phi}_{ig}(k)\| W_{ig \max}}{(2 - \alpha_{ig} \|\hat{\phi}_{ig}(k)\|^2)}
 \end{aligned} \tag{7.59}$$

and

$$\begin{aligned}
 c_1 = & \frac{\delta_{\max}^2}{(1 - (\alpha_{3f} \|\hat{\phi}_{3f}(k)\|^2 + \alpha_{3g} \|\hat{\phi}_{3g}(k)\|^2))} + \sum_{i=1}^2 \frac{\|\hat{\phi}_{if}(k)\|^2 W_{if \max}^2}{(2 - \alpha_{if} \|\hat{\phi}_{if}(k)\|^2)} \\
 & + \sum_{i=1}^2 \frac{\|\hat{\phi}_{ig}(k)\|^2 W_{ig \max}^2}{(2 - \alpha_{ig} \|\hat{\phi}_{ig}(k)\|^2)}
 \end{aligned} \tag{7.60}$$

with

$$\delta_{\max} = W_{3f \max} \tilde{\phi}_{3f \max} + W_{3g \max} \tilde{\phi}_{3g \max} + \varepsilon_{Nf} + \varepsilon_{Ng} + d_M \tag{7.61}$$

Since c_0, c_1, c_2 are positive constants, $\Delta J \leq 0$ as long as (7.42) through (7.44) and (7.54) hold. In addition, $|\sum_{k=k_0}^{\infty} \Delta J| = |J(\infty) - J(0)| < \infty$ since $\Delta J \leq 0$ as long as (7.42) through (7.44) and (7.54) hold. The definition of J and inequality (7.54) imply that every initial condition in the set χ . In other words, whenever the identification error $e(k)$ is outside the region defined by (7.54), $J(e, \tilde{W}_i)$ will decrease. This further implies that $\|e(k)\|$ will not increase and will remain χ . This demonstrates that the identification error $e(k)$ is bounded for all $k \geq 0$ and it remains to show that the weight estimates $\hat{W}_{1f}(k), \hat{W}_{2f}(k), \hat{W}_{3f}(k), \hat{W}_{1g}(k), \hat{W}_{2g}(k)$, and $\hat{W}_{3g}(k)$ or equivalently $\tilde{W}_{1f}(k), \tilde{W}_{2f}(k), \tilde{W}_{3f}(k), \tilde{W}_{1g}(k), \tilde{W}_{2g}(k)$, and $\tilde{W}_{3g}(k)$.

The dynamics relative to error in weight estimates using (7.34) and (7.41) are given by

$$\begin{aligned}\tilde{W}_{if}(k+1) &= [I - \alpha_{if}\hat{\phi}_{if}(k)\phi_{if}^T(k)]\tilde{W}_{if}(k) + \alpha_{if}\hat{\phi}_{if}(k)[\tilde{W}_{if}^T(k)\hat{\phi}_{if}(k) \\ &\quad + B_{if}e(k)]^T \quad i = 1, 2\end{aligned}\quad (7.62)$$

$$\begin{aligned}\tilde{W}_{ig}(k+1) &= [I - \alpha_{ig}\hat{\phi}_{ig}(k)\phi_{ig}^T(k)]\tilde{W}_{ig}(k) + \alpha_{ig}\hat{\phi}_{ig}(k)[\tilde{W}_{ig}^T(k)\hat{\phi}_{ig}(k) \\ &\quad + B_{ig}e(k)]^T \quad i = 1, 2\end{aligned}\quad (7.63)$$

$$\tilde{W}_{3f}(k+1) = [I - \alpha_{3f}\hat{\phi}_{3f}(k)\phi_{3f}^T(k)]\tilde{W}_{3f}(k) - \alpha_{3f}\hat{\phi}_{3f}(k)[e_g(k) + \delta(k)]^T \quad (7.64)$$

$$\tilde{W}_{3g}(k+1) = [I - \alpha_{3g}\hat{\phi}_{3g}(k)\phi_{3g}^T(k)]\tilde{W}_{3g}(k) - \alpha_{3g}\hat{\phi}_{3g}(k)[e_f(k) + \delta(k)]^T \quad (7.65)$$

where the identification error is considered to be bounded. Applying the PE condition (3.21), the identification error bound (7.54) and Lemma 3.1.2 for the cases of $\phi_i(k) = \hat{\phi}_i(k); i = 1, 2$ the boundedness of, $\tilde{W}_{1f}(k), \tilde{W}_{2f}(k), \tilde{W}_{1g}(k), \tilde{W}_{2g}(k)$, in (7.62) and (7.63) and hence of $\hat{W}_{1f}(k), \hat{W}_{2f}(k), \hat{W}_{1g}(k)$, and $\hat{W}_{2g}(k)$ are assured. For the error system (7.28), the weight updates at the third layer of the NN are presented in (7.64) with $e_g(k) = 0$. Then applying the PE condition similar to the input and hidden layers, it is straightforward to guarantee the boundedness of $\tilde{W}_{3f}(k)$ and hence $\hat{W}_{3f}(k)$.

By contrast, for the case of error system (7.29) in order to show the boundedness of the error in the weight estimates at the third layer for both NN, the passivity property of the weight updates is necessary in addition to the PE condition. Otherwise, one has to assume that the initial parameter error estimates for both $f(\cdot)$ and $g(\cdot)$ are bounded. Assuming that the initial estimation errors are bounded for both NN $f(\cdot)$ and $g(\cdot)$ and applying the PE condition (3.4), the identification error bound (7.54) and using (7.64) and (7.65), one can conclude the boundedness of $\tilde{W}_{3f}(k)$ and $\tilde{W}_{3g}(k)$ or equivalently $\hat{W}_{3f}(k)$ and $\hat{W}_{3g}(k)$.

The most elegant way of showing the boundedness of the identification error and weight estimates is to employ passivity theory. Assuming that the closed-loop systems (7.28) and (7.29) with the weight updates (7.62) through (7.65) are passive, and employing the passivity theorem (Landau 1979), one can conclude the boundedness of the identification error and error in weight updates under the PE condition. However, in the next section, this assumption can be relaxed by showing that in fact the error in weight updates is passive.

Using the boundedness of both $\|e(k)\|$ and the error in weight estimates, one can observe that (e_i, \tilde{W}_i) will not increase when both the $f(\cdot)$ and $g(\cdot)$ NNs are included for the error system (7.29) and only the network $f(\cdot)$ in the case of error system (7.28) but (e_i, \tilde{W}_i) will remain χ . Since $\chi \supset \Omega \times \Omega_U$, this concludes the proof.

Discussion: Since $\|e(k)\|$ cannot increase far beyond the right-hand side of (7.54), in applications this may be taken as a practical bound on the norm of the error $e(k)$. Note from (7.54), that the identification error increases with the NN reconstruction error bounds and the disturbance bound d_M , yet small identification errors, but not arbitrarily small, may be achieved by selecting c_0 .

As is typical of the algorithms given in this book, there is no preliminary off-line learning phase for the NN. Tuning is performed online in real-time. The required terms for tuning are easily evaluated and measured in the feedback loop. The proof is easy to extend to the case of general n -layer NN in the approximations (7.20) and (7.21) (Jagannathan 1994). The NN tuning scheme for NN identification of nonlinear systems is given in Table 7.1.

7.4 PASSIVITY PROPERTIES OF THE NN

In this section, an interesting property of the NN is shown next — the NN identifier with tuning algorithms given in Table 7.1 makes the closed-loop system passive. The practical importance (Jagannathan and Lewis 1996) of this is that additional unknown bounded disturbances do not destroy the stability and identification properties of the system. Passivity was discussed in Chapter 2. Note the NNs used in the identifiers in this chapter are feedforward NNs with no dynamics. However, embedding them into the identifier dynamics turns them into dynamical or recurrent NNs. Additional dynamics are introduced by tuning the NNs online. Therefore passivity properties can be defined.

The complete closed-loop structure using the NN identifier is given in Figure 7.3. Note that all blocks appear in standard feedback configuration. Using the fact that dynamical NNs are passive and invoking the passivity theorem (Goodwin and Sin 1984) one can easily understand why the errors in the weight estimates of all the layers are bounded. The next result details the passivity properties engendered by the tuning rules in Table 7.1.

TABLE 7.1
Multilayer NN Identifier

The weight tuning is given by

Input and hidden layers:

$$\hat{W}_{if}(k+1) = \hat{W}_{if}(k) - \alpha_{if} \hat{\phi}_{if}(k) [\hat{y}_{if}(k) + B_{if}e(k)]^T, \quad i = 1, \dots, n-1$$

and

$$\hat{W}_{ig}(k+1) = \hat{W}_{ig}(k) - \alpha_{ig} \hat{\phi}_{ig}(k) [\hat{y}_{ig}(k) + B_{ig}e(k)]^T, \quad i = 1, \dots, n-1$$

where $\hat{y}_{if}(k) = \hat{W}_{if}^T(k) \hat{\phi}_{if}(k)$; $\hat{y}_{ig}(k) = \hat{W}_{ig}^T(k) \hat{\phi}_{ig}(k)$, and $\|B_{if}\| \leq \kappa_{if}$, $\|B_{ig}\| \leq \kappa_{ig}$, $i = 1, 2, \dots, n-1$.

Output layer:

$$\hat{W}_{nf}(k+1) = \hat{W}_{nf}(k) + \alpha_{nf} \hat{\phi}_{nf}(k) e^T(k+1)$$

$$\hat{W}_{ng}(k+1) = \hat{W}_{ng}(k) + \alpha_{ng} \hat{\phi}_{ng}(k) e^T(k+1)$$

with $\alpha_{if} > 0$, $\alpha_{ig} > 0$, $i = 1, 2, \dots, n$ denoting constant learning rate parameters or adaptation gains.

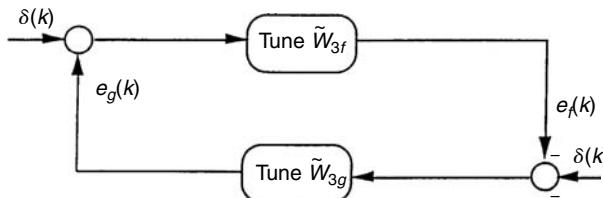


FIGURE 7.3 NN closed-loop identifier system.

Theorem 7.4.1 (Three-Layer NN Passivity Using Tuning Algorithms): Given an unknown system in one of the four forms (7.1) through (7.4), select the estimator from the respective form (7.5) through (7.8) and if required let $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ be given by NN as in (7.20) and (7.21). Then:

- The weight-tuning algorithms (7.34) through (7.37) make the maps from $W_i^T \hat{\phi}_i(k) + B_i e(k)$ to $\tilde{W}_i^T(k) \hat{\phi}_i(k)$; $i = 1, 2$ both passive maps for NN.
- The weight-tuning schemes (7.40) and (7.41) make the map from, $e_g(k) + \delta(k)$ for the case of (7.28), and $e_f(k) + \delta(k)$ for the case of (7.29), to $-\tilde{W}_{3f}^T(k) \hat{\phi}_{3f}(k)$ and $-\tilde{W}_{3g}^T(k) \hat{\phi}_{3g}(k)$ a passive map.

Proof: (a) Define the Lyapunov function candidate

$$J = \frac{1}{\alpha_{1f}} \text{tr}[\tilde{W}_{1f}^T(k) \tilde{W}_{1f}(k)] \quad (7.66)$$

whose first difference is given by

$$J = \frac{1}{\alpha_{1f}} \text{tr}[\tilde{W}_{1f}^T(k+1) \tilde{W}_{1f}(k+1) - \tilde{W}_{1f}^T(k) \tilde{W}_{1f}(k)] \quad (7.67)$$

Substituting the weight update law (7.34) in (7.67) yields

$$\begin{aligned} \Delta J = & -(2 - \alpha_{1f} \hat{\phi}_{1f}^T(k) \phi_{1f}(k))(-\tilde{W}_{1f}^T(k) \hat{\phi}_{1f}(k))^T (-\tilde{W}_{1f}^T(k) \hat{\phi}_{1f}(k)) \\ & + 2(1 - \alpha_{1f} \hat{\phi}_{1f}^T(k) \hat{\phi}_{1f}(k))(-\tilde{W}_{1f}^T(k) \hat{\phi}_{1f}(k))^T (W_{1f}^T(k) \hat{\phi}_{1f}(k) + B_{1f} e(k)) \\ & + \alpha_{1f} \hat{\phi}_{1f}^T(k) \hat{\phi}_{1f}(k) (W_{1f}^T(k) \hat{\phi}_{1f}(k) + B_{1f} e(k))^T (W_{1f}^T(k) \hat{\phi}_{1f}(k) \\ & + B_{1f} e(k)) \end{aligned} \quad (7.68)$$

Note (7.68) is in power form (2.33) as long as the condition (7.42) holds. This in turn guarantees the passivity of the weight-tuning mechanism (7.34).

Similarly, it can be demonstrated that the error in weight updates using (7.35) through (7.37) are in fact passive.

(b) Define the Lyapunov function candidate

$$J = \frac{1}{\alpha_{3f}} \text{tr}[\tilde{W}_{3f}^T(k) \tilde{W}_{3f}(k)] \quad (7.69)$$

whose first difference is given by

$$\Delta J = \frac{1}{\alpha_{3f}} \text{tr}[\tilde{W}_{3f}^T(k+1) \tilde{W}_{3f}(k+1) - \tilde{W}_{3f}^T(k) \tilde{W}_{3f}(k)] \quad (7.70)$$

Substituting the weight update law (7.40) in (7.67) yields

$$\begin{aligned} \Delta J = & -(2 - \alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k))(-\tilde{W}_{3f}^T(k) \hat{\phi}_{3f}(k))^T (-\tilde{W}_{3f}^T(k) \hat{\phi}_{3f}(k)) \\ & + 2(1 - \alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k))(-\tilde{W}_{3f}^T(k) \hat{\phi}_{3f}(k))^T (e_g(k) + \delta(k)) \\ & + \alpha_{3f} \hat{\phi}_{3f}^T(k) \hat{\phi}_{3f}(k) (e_g(k) + \delta(k))^T (e_g(k) + \delta(k)) \end{aligned} \quad (7.71)$$

which is in power form (2.33) for discrete-time systems as long as the condition (7.42) holds.

Similarly, it can be demonstrated that the error in weight updates using (7.41) are in fact passive.

Example 7.4.1 (NN Identification of Discrete-Time Nonlinear Systems): Consider the first order MIMO discrete-time nonlinear system described by

$$x(k+1) = f(x(k)) + u(k) \quad (7.72)$$

where

$$x(k) = [x_1(k) \quad x_2(k)]^T,$$

$$f(x(k)) = \begin{bmatrix} \frac{x_2(k)}{1+x_1^2(k)} \\ \frac{x_1(k)}{1+x_2^2(k)} \end{bmatrix},$$

and

$$u(k) = [u_1(k) \quad u_2(k)]^T.$$

To achieve the objective of identifying the nonlinear system, select an estimator of the form given by (7.6), with $\beta_i = 0, i > 0$, and $\beta_0 = I$, the identity matrix. The input is a periodic step input of magnitude two units with a period of 30 sec.

A sampling interval of 10 msec was considered. A three-layer NN was selected with two inputs, hidden, and two output nodes. Sigmoidal activation functions were employed in all the nodes in the hidden layer. The initial conditions for the plant and the model were chosen to be $[2 \quad -2]^T$ and $[0.1 \quad 0.6]^T$. The weights were initialized to zero with an initial threshold value of 3.0. No learning is performed initially to train the networks. The elements in the design matrix, B_i , are chosen to be 0.1. Consider the case where when the constant learning rate parameter is replaced with the projection algorithm where the adaptation gains are selected to be $\xi_1 = 1.0$, $\xi_2 = 1.0$, and $\xi_3 = 0.7$ with $\zeta_1 = \zeta_2 = \zeta_3 = 0.001$. Let us consider the case when a bounded disturbance given by

$$w(k) = \begin{cases} 0.0, & 0 \leq kT_m < 12 \\ 0.5, & kT_m \geq 12 \end{cases} \quad (7.73)$$

is acting on the plant at the time instant k . Figure 7.4 presents the tracking response of the NN identifier with projection algorithm. The magnitude of the disturbance can be increased, however the value should be bounded. The value

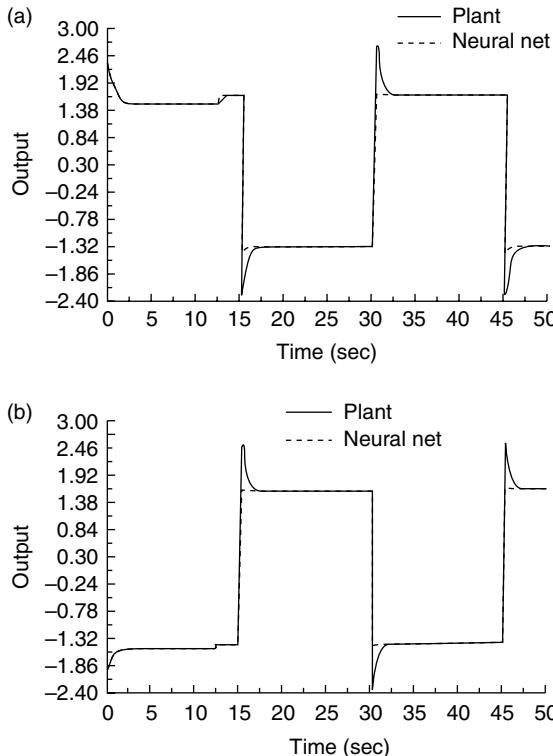


FIGURE 7.4 Response of the NN identifier with projection algorithm in the presence of bounded disturbances. (a) Desired and actual state 1. (b) Desired and actual state 2.

shown in (7.73) is employed for simulation purposes only. From the figure, it is clear that the response of the NN identifier is extremely impressive.

7.5 CONCLUSIONS

In this chapter, a general identifier was derived that estimates the system states in any of the four standard NARMA forms. NNs are used to estimate the nonlinear functions appearing in the dynamics so that the state estimate converges to the actual state in the unknown system.

A nonlinear-in-the-parameter three-layer NN was used so that the function approximation property of NN guarantees the existence of the identifier. Passivity properties of the NN identifier were discussed.

REFERENCES

- Billings, S.A., Jamalludin, H.B., and Chen, S., Properties of neural networks with applications to modeling nonlinear dynamical systems. *Int. J. Contr.*, 55, 193–224, 1992.
- Cybenko, G., Approximations by superpositions of sigmoidal activation function, *Math. Contr. Signals, Syst.*, 2, 303–314, 1989.
- Erzberger, H., Analysis and design of model following systems by state space techniques, *Proceedings of the Joint Automatic Control Conference*, Ann Arbor, pp. 572–581, 1968.
- Fung, C.F., Billings, S.A., and Zhang, H., Generalized transfer functions of neural networks, *Mech. Syst. Signal Process.*, 11, 843–868, 1997.
- Goodwin, G.C. and Sin, K.S., *Adaptive Filtering, Prediction and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- Jagannathan, S., *Intelligent control of nonlinear dynamical systems using multilayer neural networks*, Ph.D. Thesis, Department of Electrical Engineering, The University of Texas at Arlington, Arlington, TX, 1994.
- Jagannathan, S. and Lewis, F.L., Identification of nonlinear dynamical systems using multilayer neural networks, *Automatica*, 32, 1707–1712, 1996.
- Landau, I.D., *Adaptive Control: The Model Reference Approach*, Marcel Dekker, New York, 1979.
- Landau, I.D., Evolution of adaptive control, *ASME J. Dynam. Syst. Meas. Contr.*, 115, 381–391, 1993.
- Ljung, L. and Soderstrom, T., *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA, 1983.
- Narendra, K.S. and Parthasarathy, K.S., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, 4–27, 1990.
- Narendra, K.S. and Annaswamy, A.M., *Stable Adaptive Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Sadegh, N., A perceptron network for functional identification and control of nonlinear systems, *IEEE Trans. Neural Netw.*, 4, 982–988, 1993.
- Sussman, H.J., Uniqueness of the weights for minimal feedforward nets with given input-output map, *Neural Netw.*, 5, 589–593, 1992.

PROBLEMS

SECTION 7.3

7.3-1: Multilayer NN. For the system described by

$$x(k+1) = f(x(k), x(k-1)) + u(k)$$

where

$$f(x(k), x(k-1)) = \frac{x(k)x(k-1)[x(k) + 2.0]}{1 + x^2(k) + x^2(k-1)}$$

Design a multilayer NN identifier with or without learning phase and by using the developed delta rule-based weight tuning algorithm and appropriately choosing the adaptation gains for a sinusoidal input of a chosen magnitude and frequency.

7.3-2: *Multilayer NN.* For the system described by

$$x(k+1) = f(x(k), x(k-1)) + u(k)$$

where

$$f(x(k), x(k-1)) = \frac{x(k)}{1+x^2(k)} + u(k)$$

Design a multilayer NN identifier with or without learning phase and by using the developed delta rule-based weight tuning algorithm and appropriately choosing the adaptation gains for a sinusoidal input of a chosen magnitude and frequency.

7.3-3: *Stability and convergence for an n-layer NN.* Assume the hypothesis presented for three-layer NN and use the weight updates given in (7.34) through (7.41) and show the convergence and boundedness of identification error and error in weight updates for a *n*-layer NN.

SECTION 7.4

7.4-1: *Passivity properties for an n-layer NN.* Show the passivity properties of the input and hidden layers for an *n*-layer NN using delta rule-based weight tuning.

8 Discrete-Time Model Reference Adaptive Control

Recent advances in nonlinear control theory have inspired the development of adaptive control schemes for nonlinear plants. It is well known that the global stability properties of model reference adaptive systems (Narendra and Annaswamy 1989) are guaranteed under the assumption that there are no modeling errors and external disturbances acting on the plant. This restrictive assumption is often violated in applications, and therefore it is important to determine the stability and robustness of such adaptive techniques with respect to modeling errors and bounded disturbances.

Neural networks (NN) have been increasingly employed for the adaptive control of nonlinear systems, as these networks do not require a priori knowledge of the dynamics of the system to be controlled. To the contrary, in the conventional adaptive control, a regression matrix for each dynamic system needs to be computed that is quite complex and requires a lot of computational time. The NN-based adaptive control of nonlinear systems is being investigated by many researchers both in continuous- and discrete-time. Previous chapters (see Chapter 2 through Chapter 6) presented the direct adaptive control of nonlinear discrete-time systems that guarantee tracking performance through a Lyapunov-based approach (Jagannathan and Lewis 1996a). On the other hand, an indirect model reference adaptive controller design has been treated in Narendra and Parthasarathy (1990).

Persistent problems that remain to be adequately addressed in using discrete-time NN for direct model reference adaptive control (MRAC) include ad hoc controller structures and the inability to guarantee satisfactory performance of the system. Uncertainty in initializing the NN weights leads to the necessity for preliminary off-line training (Narendra and Parthasarathy 1990) or a stiff assumption that stabilizing weights are known. In addition, the backpropagation-tuning algorithm requires the evaluation of sensitivity derivatives along the network signal paths, which is highly time consuming,

and often impossible in closed-loop uncertain systems as the plant Jacobian matrix is unknown.

To confront all these issues head on, in the previous chapters, a novel scheme was investigated for a single and multilayer discrete-time NN whose weights were tuned online with no initial explicit learning phase needed. In other words, the NN exhibited a learning-while-functioning-feature instead of learning-then-control. These weights were updated by using the passivity approach. Weight initialization was easy and a local uniform ultimate boundedness (UUB) was demonstrated. Specifically, the weight-tuning mechanisms guaranteed the boundedness of the tracking error and the NN weights, when the weights were initialized at zero, even though there did not exist target weights such that the NN perfectly reconstructed a certain required function. In this paper (Jagannathan and Lewis 1996b), an approach similar to ϵ -modification is developed in discrete-time for the adaptive control of nonlinear systems that can be expressed as linear-in-the-unknown parameters (LIP). This approach presented in Jagannathan and Lewis (1996b) avoids the necessity of persistency of excitation (PE) condition on input signals.

On the other hand, in Jagannathan et al. (1996), MRAC of a class of nonlinear dynamical systems is presented and these results are covered in this chapter. PE is not needed, LIP is not required, and certainty equivalence (CE) is not used overcoming several limitations of standard adaptive control. The MRAC ensures good tracking performance, as shown through the Lyapunov's stability approach and the NN weights are bounded without using the passivity property of weight updates. It is found that the maximum permissible tuning rate for the NN weight tuning decreases as the NN size increases; this is a major drawback. Therefore, it is demonstrated that a projection algorithm can easily correct the problem, allowing larger NN to be tuned more quickly. The plant is assumed to be controllable and the state vector of the plant is assumed to be available for measurement. Simulation results are given to demonstrate the theoretical conclusions.

8.1 DYNAMICS OF AN M NTH-ORDER MULTI-INPUT AND MULTI-OUTPUT SYSTEM

Consider a m nth-order multi-input and multi-output (MIMO) discrete-time nonlinear system, to be controlled, given in the form,

$$\begin{aligned} x_1(k+1) &= x_2(k) \\ &\vdots \\ x_{n-1}(k+1) &= x_n(k) \\ x_n(k+1) &= f(x(k)) + u(k) + d(k) \quad y(k) = x_1(k) \end{aligned} \tag{8.1}$$

with state $x(k) = [x_1^T(k), \dots, x_n^T(k)]^T$ with $x_i(k) \in \Re^m$, control input $u(k) \in \Re^m$, output $y(k) \in \Re^m$, and $d(k) \in \Re^m$ a disturbance vector acting on the system at the instant k with $\|d(k)\| \leq d_M$ a known constant.

A reference model is chosen as

$$\bar{x}(k+1) = \bar{A}\bar{x}(k) + \bar{B}\bar{r}(k) \quad \bar{y}(k) = \bar{x}_1(k) \quad (8.2)$$

with state $\bar{x}(k) = [\bar{x}_1^T(k), \dots, \bar{x}_n^T(k)]^T$, and $\bar{r}(k) \in \Re^m$ a bounded reference input. The matrices \bar{A} and \bar{B} are selected so that an asymptotically stable (AS) reference model with desirable properties is obtained. It is desired to determine $u(k)$ so that the output of the plant $y(k) \in \Re^m$ follows the output of the reference model $\bar{y}(k) \in \Re^m$. It is assumed that the plant is controllable and the state vector of the plant is accessible. In other words, the aim is to determine the control input $u(k)$ for all $k \geq k_0$ so that

$$\lim_{k \rightarrow \infty} \|y(k) - u(k)\| \leq \delta \quad (8.3)$$

for some specified constant $\delta \geq 0$.

Define the output tracking error as

$$e(k) = y(k) - \bar{y}(k) = x_1(k) - \bar{x}_1(k) \quad (8.4)$$

and define the filtered tracking error, $r(k) \in \Re^m$, as

$$r(k) = e(k+n-1) + \lambda_1 e(k+n-2) + \dots + \lambda_{n-1} e(k) \quad (8.5)$$

where $e(k+n-1), e(k+n-2), \dots, e(k+1)$ are the future values of the error $e(k)$, and $\lambda_1, \lambda_2, \dots, \lambda_{n-1}$ are constant m by m matrices selected so that $|z^{n-1} + \lambda_1 z^{n-2} + \dots + \lambda_{n-1}|$ is stable. Note from (8.5) that the filtered tracking error depends upon the future values of the output of the reference model $\bar{y}(k)$ which may be available if the reference input $\bar{r}(k)$ is known ahead. On the other hand, suppose the reference model (8.2) is in the same form as the plant, which is,

$$\begin{aligned} \bar{x}_1(k+1) &= \bar{x}_2(k) \\ &\vdots \\ \bar{x}_{n-1}(k+1) &= \bar{x}_n(k) \\ \bar{x}_n(k+1) &= k_m \bar{x}(k) + \bar{r}(k) \quad y(k) = x_1(k) \end{aligned} \quad (8.6)$$

The state $k_m \in \Re^{m \times nm}$ is a design parameter matrix chosen by pole placement. Then $\bar{A} \in \Re^{nm \times nm}$, $\bar{B} \in \Re^{nm \times m}$ in (8.2) has a special form with $\bar{C} = [I \bar{0} \cdots \bar{0}]$ and $I \in \Re^{m \times m}$. In this case, future values of the reference are not needed.

In fact, it then follows that

$$e(k+i) = x_{i+1}(k) - \bar{x}_{i+1}(k) \equiv e_{i+1}(k) \quad i = 0, \dots, n-1$$

which is known at time k .

Using (8.6), Equation 8.5 can be rewritten as

$$r(k) = e_n(k) + \lambda_1 e_{n-1}(k) + \dots + \lambda_{n-1} e_1(k) \quad (8.7)$$

where $e_{n-1}(k), \dots, e_1(k)$ are the delayed values of the error $e_n(k)$. Equation 8.7 can be further expressed as

$$r(k+1) = e_n(k+1) + \lambda_1 e_{n-1}(k+1) + \dots + \lambda_{n-1} e_1(k+1) \quad (8.8)$$

Substituting (8.1) in (8.8), the dynamics of the m th-order MIMO system can be written in terms of the filtered tracking error as

$$\begin{aligned} r(k+1) &= f(x(k)) - k_m \bar{x}(k) - \bar{r}(k) + \lambda_1 e_n(k) + \dots + \lambda_{n-1} e_2(k) \\ &\quad + u(k) + d(k) \end{aligned} \quad (8.9)$$

Define the control input $u(k)$ in (8.9) as

$$u(k) = k_m \bar{x}(k) + \bar{r}(k) - \hat{f}(x(k)) + k_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) \quad (8.10)$$

with the diagonal gain matrix $k_v > 0$, and $\hat{f}(x(k))$ an estimate of $f(x(k))$. Then the closed-loop error system becomes

$$r(k+1) = k_v r(k) + \tilde{f}(x(k)) + d(k) \quad (8.11)$$

where the functional estimation error is given by

$$\tilde{f}(x(k)) = f(x(k)) - \hat{f}(x(k)) \quad (8.12)$$

This is an error system wherein the filtered tracking error is driven by the functional estimation error.

In this chapter, (8.11) is used to focus on selecting NN tuning algorithms that guarantee the stability of the filtered tracking error $r(k)$. Then since (8.5) and (8.7) with the input considered as $r(k)$ and the output $e(k)$, describe a stable system, standard techniques (Jagannathan and Lewis 1996b) guarantee that $e(k)$ exhibits a stable behavior.

8.2 NN CONTROLLER DESIGN

Approaches such as σ -modification or ε -modification (Narendra and Annaswamy 1987) are available for the robust adaptive control of continuous systems wherein a PE condition is not needed. Based on the author's knowledge at the time of writing this, weight-tuning updates in discrete-time similar to σ - or ε -modification to avoid the necessity for the PE condition, have been presented for the first time in Jagannathan and Lewis (1996b). In this chapter, an approach similar to σ ε -modification derived by Jagannathan et al. (1996) for model reference adaptive control of discrete-time systems using NN is discussed. Unfortunately, for guaranteed stability, it is found that the weight tuning using the delta rule at each layer must slow down as the NN becomes larger. This is a problem often noted in the NN control literature (Rumelhart et al. 1990). In the next section, by employing a projection algorithm, it is shown that the tuning rate can be made independent of the NN size.

Assume that there exist some constant ideal weights $W_n, W_{n-1}, \dots, W_2, W_1$ for an n -layer NN so that the nonlinear function in (8.1) can be written as $f(x) = W_n^T \phi_n(W_{n-1}^T \phi_{n-1} \dots)$ where $\|\varepsilon(x(k))\| \leq \varepsilon_N$, with the bounding constant ε_N known. Unless the network is minimal, suitable ideal weights may not be unique (Sussmann 1992). The target weights may be defined as those that minimize the supremum norm over S of $\varepsilon(x(k))$. The issue is not a major concern here as it is needed to know only existence of target weights; their actual values are not needed. This assumption is similar to Erzberger's assumption (Erzberger 1968) in LIP adaptive control. For notational convenience define the matrix of all the target weights as

$$Z = \begin{bmatrix} W_1 \\ \vdots \\ W_n \end{bmatrix}$$

with padding by zeros as required for dimensional consistency. Then the next mild bounding assumption can be stated.

Assumption 8.2.1: The target weights are bounded by known positive values so that

$$\|W_1\| \leq W_{1\max}, \|W_2\| \leq W_{2\max}, \dots, \|W_n\| \leq W_{n\max} \quad \text{or} \quad \|Z\| \leq Z_{\max}$$

8.2.1 NN CONTROLLER STRUCTURE AND ERROR SYSTEM DYNAMICS

Define the NN functional estimate by

$$f(x(k)) = \hat{W}_n^T \phi[\hat{W}_{n-1}^T \phi(\dots \hat{W}_1^T \phi(x(k)))] \quad (8.13)$$

with $\hat{W}_n, \hat{W}_{n-1}, \dots, \hat{W}_2, \hat{W}_1$ the current weight values. The vector of input-layer activation functions is given by $\hat{\phi}_1(k) = \phi_1(k) = \phi(x(k))$. Then the vector of activation functions of the hidden and output layer with the actual weights at the instant k is denoted by

$$\hat{\phi}_{m+1}(k) = \phi(\hat{W}_m^T \hat{\phi}_m(k)) \quad \forall m = 1, \dots, n-1 \quad (8.14)$$

For activation functions such as sigmoid, tanh, radial basis function (RBF), and so on, the following fact can be stated.

Fact 8.2.1: The activation functions are bounded by known positive values so that

$$\|\hat{\phi}_1(k)\| \leq \phi_{1\max}, \|\hat{\phi}_2(k)\| \leq \phi_{2\max}, \dots, \text{ and } \|\hat{\phi}_n(k)\| \leq \phi_{n\max} \quad (8.15)$$

The error in weights or weight estimation errors are defined as

$$\tilde{W}_n(k) = W_n - \hat{W}_n(k), \dots, \tilde{W}_2(k) = W_2 - \hat{W}_2(k), \tilde{W}_1(k) = W_1 - \hat{W}_1(k) \quad (8.16)$$

and the hidden-layer output errors are defined as

$$\tilde{\phi}_n(k) = \phi_n - \hat{\phi}_n(k), \dots, \tilde{\phi}_2(k) = \phi_2 - \hat{\phi}_2(k), \tilde{\phi}_1(k) = \phi_1 - \hat{\phi}_1(k) \quad (8.17)$$

Select the reference model given in (8.6), and the control input $u(k)$ in (8.10) is taken as

$$u(k) = -\hat{W}_n^T \hat{\phi}_n(k) + k_m \bar{x}(k) + \bar{r}(k) + k_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k) \quad (8.18)$$

where the functional estimate (8.13) is provided by an n -layer NN and denoted in (8.18) by $\hat{W}_n^T \hat{\phi}_n(k)$.

Then the closed-loop tracking error dynamics in (8.11) become

$$r(k+1) = k_v r(k) + \bar{e}_i(k) + W_n^T \tilde{\phi}(k) + \varepsilon(k) + d(k) \quad (8.19)$$

where the identification error is given by

$$\bar{e}_i(k) = \tilde{W}_n^T(k) \hat{\phi}_n(k) \quad (8.20)$$

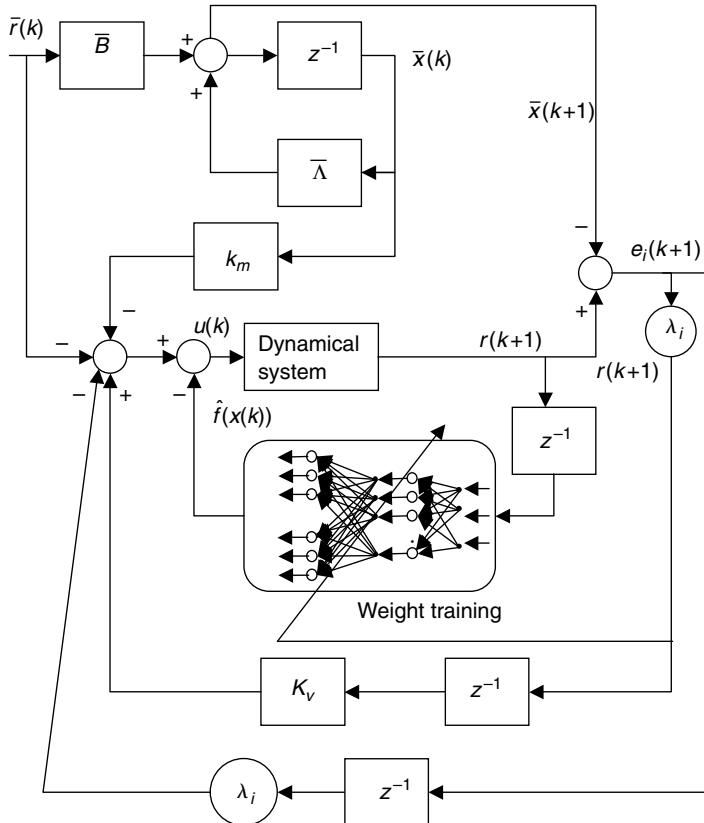


FIGURE 8.1 NN controller structure.

The proposed NN controller structure is illustrated in Figure 8.1. The output of the plant is processed through a series of delays in order to obtain the past values of the output and fed as inputs to the NN so that the nonlinear function in (8.1) can be suitably approximated. Thus, the NN controller derived in a straightforward manner using error notion naturally provides a dynamical NN control structure. Note that neither the input $u(k)$ nor its past values are needed by the NN. The next step is to determine the weight updates so that the tracking performance of the closed-loop error dynamics is guaranteed.

A novel improved NN weight-tuning paradigm that guarantees the stability of the closed-loop system (8.19) is presented in the next section. It is required to demonstrate that the tracking error $r(k)$ is bounded and that the NN weights $\hat{W}_i(k); \forall i = 1, \dots, n$ remain bounded, for then the control $u(k)$ is bounded.

TABLE 8.1
Model Reference Adaptive Controller Using an n -Layer NN

The control input is

$$u(k) = -\hat{W}_n^T \hat{\phi}_n(k) + k_m \bar{x}(k) + \bar{r}(k) + k_v r(k) - \lambda_1 e_n(k) - \dots - \lambda_{n-1} e_2(k)$$

Consider the weight-tuning algorithms provided for the input and hidden layers as:

$$\hat{W}_i(k+1) = \hat{W}_i(k) - \alpha_i \hat{\phi}_i(k) [\hat{y}_i(k) + B_i k_v r(k)]^T - \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\| \hat{W}_i(k),$$

$$i = 1, \dots, n-1$$

and for the output layer as:

$$\hat{W}_n(k+1) = \hat{W}_n(k) + \alpha_n \hat{\phi}_n(k) r^T(k+1) - \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \hat{W}_n(k)$$

with $\Gamma > 0$ a design parameter, and B_i , $i = 1, \dots, n$, are design parameter matrices selected such that $\|B_i\| \leq \kappa_i$.

8.2.2 WEIGHT UPDATES FOR GUARANTEED TRACKING PERFORMANCE

The tuning algorithm in Table 8.1 is derived in the next theorem, which guarantees performance of a discrete-time MRAC without the need for a PE condition in the case of a multilayer NN. This theorem relies on the extension to Lyapunov theory for dynamical systems as Theorem 1.5-6 in Lewis et al. (1993). The nonlinearity $f(x)$, the bounded disturbance $d(k)$, and the NN reconstruction error, ε , make it impossible to show that the first difference for a Lyapunov function is nonpositive for all values of $r(k)$ and weight values. In fact, it is only possible to show that the first difference is negative outside a compact set in the state space if either $\|r(k)\|$ or $\|\tilde{Z}(k)\|$ are above some specific bounds.

Therefore, if either norm increases too much, the Lyapunov function decreases so that both norms also decrease. If both norms are small, nothing may be said about the first difference of the Lyapunov function except it is probably positive, so that the Lyapunov function increases. This has the effect of making the boundary of a compact set an attractive region for the closed-loop system. This, however, allows one to conclude the boundedness of the output tracking error and the NN weights.

Theorem 8.2.1 (MRAC of Nonlinear Systems): Let the reference input $\bar{r}(k)$ be bounded and the NN functional estimation error and the disturbance bounds, ε_N, d_M respectively be known constants. Consider the weight-tuning algorithms

provided for the input and hidden layers as

$$\begin{aligned}\hat{W}_i(k+1) &= \hat{W}_i(k) - \alpha_i \hat{\phi}_i(k) [\hat{y}_i(k) + B_i k_v r(k)]^T \\ &\quad - \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\| \hat{W}_i(k); \quad i = 1, \dots, n-1\end{aligned}\quad (8.21)$$

and for the output layer as

$$\begin{aligned}\hat{W}_n(k+1) &= \hat{W}_n(k) + \alpha_n \hat{\phi}_n(k) r^T(k+1) \\ &\quad - \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \hat{W}_n(k); \quad i = n\end{aligned}\quad (8.22)$$

with $\Gamma > 0$ a design parameter, and B_i , $i = 1, \dots, n$, are design parameter matrices selected such that $\|B_i\| \leq \kappa_i$. Then, the tracking error $r(k)$ and the NN weight estimates $\hat{W}_i(k)$; $\forall i = 1, \dots, n$ are UUB, with the bounds given specifically by (8.38) and (8.39) provided the following conditions hold:

$$\begin{aligned}1. \quad \alpha_i \phi_{i \max}^2 &< 2, \quad \forall i = 1, \dots, n-1 \\ &< 1, \quad i = n\end{aligned}\quad (8.23)$$

$$2. \quad 0 < \Gamma < 1 \quad (8.24)$$

$$3. \quad k_v \max < \frac{1}{\sqrt{\sigma}} \quad (8.25)$$

where σ is given by

$$\sigma = \beta_n + \sum_{i=1}^{n-1} \kappa_i^2 \beta_i \quad (8.26)$$

with

$$\beta_i = \alpha_i \phi_{i \max}^2 + \frac{[(1 - \alpha_i \phi_{i \max}^2) + \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|]^2}{(2 - \alpha_i \phi_{i \max}^2)} \quad (8.27)$$

and

$$\beta_n = 1 + \alpha_n \phi_{n \max}^2 + \frac{[\alpha_n \phi_{n \ max}^2 + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\|]^2}{(2 - \alpha_n \phi_{n \ max}^2)} \quad (8.28)$$

Proof: Select the Lyapunov function candidate

$$J = r^T(k) r(k) + \sum_{i=1}^n \frac{1}{\alpha_i} \text{tr}(\tilde{W}_i^T(k) \tilde{W}_i(k)) \quad (8.29)$$

whose first difference is given by

$$\begin{aligned}\Delta J &= \Delta J_1 + \Delta J_2 \\ &= r^T(k+1)r(k+1) - r^T(k)r(k) \\ &\quad + \sum_{i=1}^n \frac{1}{\alpha_i} \text{tr}(\tilde{W}_i^T(k+1)\tilde{W}_i^T(k+1) - \tilde{W}_i^T(k)\tilde{W}_i^T(k))\end{aligned}\quad (8.30)$$

The first difference in (8.30) is computed in two steps and is put together in the third step. The following are the necessary steps for the computation of the first difference.

Step 1: Using the tracking error dynamics (8.19), the first term in (8.30), ΔJ_1 is obtained as

$$\begin{aligned}\Delta J_1 &= -r^T(k)[I - k_v^T k_v] + 2(k_v r(k))^T (\bar{e}_i(k) + W_n^T \tilde{\phi}_n(k) + \varepsilon(k) + d(k)) \\ &\quad \times \bar{e}_i^T(k) \bar{e}_i(k) + 2W_n^T \tilde{\phi}_n(k) + 2(\varepsilon(k) + d(k)) + (W_n^T \tilde{\phi}_n(k))^T W_n^T \tilde{\phi}_n(k) \\ &\quad \times 2(\varepsilon(k) + d(k))(\varepsilon(k) + d(k))^T (\varepsilon(k) + d(k))\end{aligned}\quad (8.31)$$

Step 2: Considering the input and hidden (8.21), output (8.22) weight updates and using these in the second term, one may obtain

$$\begin{aligned}\Delta J_2 &\leq -\sum_{i=1}^{n-1} [2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)] \left\| \tilde{W}_i^T(k) \hat{\phi}_i(k) \right. \\ &\quad \left. - \frac{[(1 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)) - \Gamma \|I - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)\|]}{(2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k))} \right. \\ &\quad \left. \times (W_i^T \hat{\phi}_i(k) + B_l k_v r(k)) \right\|^2 + \left(\sum_{i=1}^{n-1} \beta_i \kappa_i^2 \right) k_v^2 \max \|r(k)\|^2 \\ &\quad + \sum_{i=1}^{n-1} (\beta_i + 2\Gamma \|I - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)\|) W_{i \max}^2 \phi_{i \max}^2\end{aligned}$$

$$\begin{aligned}
& \times \sum_{i=1}^{n-1} (\beta_i + \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|) \kappa_i W_{i \max} \phi_{i \max} k_{v \max} \|r(k)\| \\
& - [1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k)] \left\| \bar{e}_i(k) - [[k_v r(k) \right. \right. \\
& \left. \left. + (\alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k) + \Gamma \|I - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k)\|) \right. \right. \\
& \left. \left. \times (W_3^T \tilde{\phi}(k) + \varepsilon(k) + d(k))] \times [(1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k))]^{-1} \right\|^2 \\
& + \sum_{i=1}^n \frac{1}{\alpha_i} \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|^2 \text{tr}\{\Gamma^2 \hat{W}_i^T(k) \hat{W}_i(k) + 2\Gamma \hat{W}_i^T(k) \tilde{W}_i(k)\} \\
& + \beta_n k_{v \max}^2 \|r(k)\|^2 + [\beta_n (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \phi_{n \max} W_{n \max}] (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + 2k_{v \max} \|r(k)\| [\beta_n (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \phi_{n \max} W_{n \max}] \quad (8.32)
\end{aligned}$$

where β_n and β_i are presented in (8.28) and (8.27) respectively.

Step 3: Combining (8.31) and (8.32) to get

$$\begin{aligned}
\Delta J \leq & -[1 - \sigma k_{v \max}^2] \|r(k)\|^2 + 2k_{v \max} \gamma \|r(k)\| + \rho - \sum_{i=1}^{n-1} [2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)] \\
& \times \left\| \tilde{W}_i^T(k) \hat{\phi}_i(k) - \frac{[(1 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)) - \Gamma \|I - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)\|]}{(2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k))} \right. \\
& \left. \times (W_i^T \hat{\phi}_i(k) + B_i k_v r(k)) \right\|^2 - (1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k)) \\
& \times \|\bar{e}_i(k) - ([k_v r(k) + (\alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k) + \Gamma \|I - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k)\|) \right. \\
& \left. \times (W_3^T \tilde{\phi}(k) + \varepsilon(k) + d(k))] (1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3^T(k))^{-1} \|^2 \\
& + \sum_{i=1}^n \frac{1}{\alpha_i} \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|^2 \text{tr}\{\Gamma^2 \hat{W}_i^T(k) \hat{W}_i(k) + 2\Gamma \hat{W}_i^T(k) \tilde{W}_i(k)\} \\
& + \beta_n k_{v \max}^2 \|r(k)\|^2 + [\beta_n (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \phi_{n \max} W_{n \max}] (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + 2k_{v \max} \|r(k)\| [\beta_n (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \\
& + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \phi_{n \max} W_{n \max}] \quad (8.33)
\end{aligned}$$

where

$$\begin{aligned} \gamma &= [\beta_n(W_{n \max} \hat{\phi}_{n \max} + \varepsilon_N + d_M) + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \phi_{n \max} W_{n \max}] \\ &\quad \times (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \sum_{i=1}^{n-1} (\beta_i + \Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|) \\ &\quad \times \kappa_i W_{i \max} \phi_{i \max} \end{aligned} \quad (8.34)$$

$$\begin{aligned} \rho &= [\beta_n(W_{n \max} \hat{\phi}_{n \max} + \varepsilon_N + d_M) + \Gamma \|I - \alpha_n \hat{\phi}_n(k) \hat{\phi}_n^T(k)\| \kappa_i \phi_{n \max} W_{n \max}] \\ &\quad \times (W_{n \max} \tilde{\phi}_{n \max} + \varepsilon_N + d_M) \sum_{i=1}^{n-1} (\beta_i + 2\Gamma \|I - \alpha_i \hat{\phi}_i(k) \hat{\phi}_i^T(k)\|) \\ &\quad \times W_{i \max}^2 \phi_{i \max}^2 \end{aligned} \quad (8.35)$$

Rewriting the last term in (8.33) in terms of $\|\tilde{Z}(k)\|$ and completing the squares for $\|\tilde{Z}(k)\|$ in (8.33), one may obtain

$$\begin{aligned} \Delta J &\leq -[1 - \sigma k_{v \max}^2] \left[\|r(k)\|^2 - \frac{2\gamma k_{v \max}}{(1 - \sigma k_{v \max}^2)} \|r(k)\| \right. \\ &\quad \left. - \frac{(\rho + c_0 Z_{\max}^2)}{(1 - \sigma k_{v \max}^2)} \right] - \sum_{i=1}^{n-1} [2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)] \\ &\quad \times \left\| \tilde{W}_i^T(k) \hat{\phi}_i(k) - \frac{[(1 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)) - \Gamma \|I - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)\|]}{(2 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k))} \right. \\ &\quad \times (W_i^T \hat{\phi}_i(k) + B_i k_v r(k)) \left. \right\|^2 - (1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k)) \\ &\quad \times \|\bar{e}_i(k) - ([k_v r(k) + (\alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k) + \Gamma \|I - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k)\|)] \\ &\quad \times (W_3^T \tilde{\phi}(k) + \varepsilon(k) + d(k)](1 - \alpha_3 \hat{\phi}_3^T(k) \hat{\phi}_3(k))^{-1})\|^2 \\ &\quad - \Gamma (2 - \Gamma) c_{\min} \left[\|\tilde{Z}(k)\| - \frac{(1 - \Gamma)}{(2 - \Gamma)} \frac{c_{\max}}{c_{\min}} Z_{\max} \right]^2 \end{aligned} \quad (8.36)$$

with

$$c_0 = \frac{c_{\max}}{c_{\min}} \frac{1}{(2 - \Gamma)} [(1 - \Gamma)^2 c_{\max} + \Gamma^2 (2 - \Gamma) c_{\min}] \quad (8.37)$$

a positive constant, and c_{\max} and c_{\min} denote the maximum and minimum singular value of the diagonal matrix given by

$$\begin{bmatrix} \frac{1}{\alpha_i} \|1 - \alpha_i \hat{\phi}_i^T(k) \hat{\phi}_i(k)\|^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{1}{\alpha_n} \|1 - \alpha_n \hat{\phi}_n^T(k) \hat{\phi}_n(k)\|^2 \end{bmatrix} \quad (8.38)$$

Then $\Delta J \leq 0$ as long as (8.23) and (8.25) hold and the quadratic term for $r(k)$ in (8.36) is positive, which is guaranteed when

$$\|r(k)\| > \frac{1}{(1 - \sigma k_{v \max}^2)} [\gamma k_{v \max} + \gamma^2 k_{v \max}^2 + [\rho + c_0 Z_{\max}^2] (1 - \sigma k_{v \max}^2)]^{1/2} \quad (8.39)$$

with $\gamma > 0$ and $\rho > 0$ in (8.34) and (8.35) respectively. Similarly completing the squares for $\|r(k)\|$ in (8.36), then $\Delta J \leq 0$ as long as (8.23) and (8.25) hold and the quadratic term for $\|\tilde{Z}(k)\|$ is positive, which is guaranteed when

$$\begin{aligned} \|\tilde{Z}(k)\| &> \frac{1}{\Gamma(2 - \Gamma)} [\Gamma(1 - \Gamma) c_{\max} Z_{\max} \\ &\quad + (\Gamma^2(1 - \Gamma^2) c_{\max} Z_{\max}^2 + \Gamma(2 - \Gamma) c_{\min} \theta)^{1/2}] \end{aligned} \quad (8.40)$$

where

$$\theta = \left[\Gamma^2 c_{\max} Z_{\max}^2 + \frac{\gamma^2 k_{v \max}^2}{(1 - \sigma k_{v \max})} + \rho \right] \quad (8.41)$$

Remarks:

1. For practical purposes (8.39) and (8.40) can be considered as bounds for $\|r(k)\|$ and $\|\tilde{Z}(k)\|$.
2. Note that the NN reconstruction error bound ε_N and bounded disturbances d_M increase the bounds on $\|r(k)\|$ and $\|\tilde{Z}(k)\|$ in a very interesting way. Note that small tracking error bounds may be

achieved by placing the closed-loop poles inside the unit circle and near the origin through the selection of the largest eigenvalue, $k_{v\max}$. On the other hand, the NN weight error estimates are fundamentally bounded by Z_{\max} , the known bound on the ideal weights Z . The parameter Γ offers a design tradeoff between the relative eventual magnitudes of $\|r(k)\|$ and $\|\tilde{Z}(k)\|$; a smaller Γ yields a smaller $\|r(k)\|$ and a larger $\|\tilde{Z}(k)\|$, and vice versa.

3. The effect of adaptation gains α_i , $i = 1, \dots, n$ at each layer on the weight estimation error, $\tilde{Z}(k)$, and tracking error, $r(k)$, can be easily observed by using the bounds presented in (8.39) and (8.40) through c_{\min} and c_{\max} . Large values of adaptation gains force smaller weight estimation errors and the tracking error is increased. In contrast, a large value of adaptation gain, α_n , forces smaller tracking and weight estimation errors.
4. The effect of the tracking error on the hidden-layer weights and the position of the closed-loop poles can be observed through the design parameters, κ_i , $i = 1, \dots, n$. These parameters weigh the tracking error while, in turn, drive the hidden-layer weight updates. A large value of κ_i will increase the hidden-layer weights and bounding constants γ and ρ . This in turn causes the position of the poles to move closer to the origin, resulting in the input forcing the tracking error to converge to the compact set as fast as possible.
5. It is important to note that the problem of initializing the weights occurring in other techniques in the literature does not arise because when the actual weights are initialized to zero, the conventional proportional and derivative terms stabilize the system, on an interim basis until the NN begins to learn in certain classes of nonlinear systems such as the robotic systems. Thus the NN controller requires no learning phase.

8.3 PROJECTION ALGORITHM

If the adaptation gains for an n -layer NN, $\alpha_i > 0$, $i = 1, 2, \dots, n$, are constant parameters in the update laws presented in (8.21) and (8.22), these update laws correspond to the delta rule (Rumelhart et al. 1990) also referred to as the Widrow–Hoff rule (Widrow and Lehr 1990). The theorem reveals that the delta-rule-based update tuning mechanisms at each layer has a major drawback. Using the bounds presented from (8.23), it is evident that the upper bound on the adaptation gain at each layer depends upon the number of hidden-layer neurons present in the particular layer. Specifically, if there are N_p hidden-layer neurons and the maximum value of each hidden-node output in the i th layer is taken as

unity (as for the sigmoid), then the bounds on the adaptation gain in order to ensure stability of the closed-loop system are given by

$$\begin{aligned} 0 < \alpha_i &< \frac{2}{N_p} \quad i = 1, \dots, n-1 \\ 0 < \alpha_i &< \frac{1}{N_p} \quad i = n \end{aligned} \quad (8.42)$$

In other words, the upper bound on the adaptation gain at each layer decreases with an increase in the number of hidden-layer neurons in that particular layer, so that learning must slow down for guaranteed performance. This behavior is noted in the literature (Jagannathan and Lewis 1996a), but never to our knowledge explained. This problem is fixed in the work of Jagannathan and Lewis (1996a) by using a projection algorithm (Astrom and Wittenmark 1989). To wit, replace the constant adaptation gain at each layer by

$$\alpha_i = \frac{\xi_i}{\xi_i + \|\phi_i(k)\|^2} \quad i = 1, \dots, n \quad (8.43)$$

where

$$\xi_i > 0 \quad i = 1, \dots, n \quad (8.44)$$

and

$$0 < \xi_i < 2 \quad i = 1, \dots, n-1$$

$$0 < \xi_i < 1 \quad i = n$$

are constants. Note that $\xi_i, i = 1, \dots, n$ is now the new adaptation gain at each layer. Equation 8.44 guarantees that there is no need to decrease the learning rate with the number of neurons at that layer.

Example 8.3.1 (Model Reference Adaptive Controller for Nonlinear Discrete-Time Systems): In order to illustrate the performance of the NN controller, a discrete-time MIMO nonlinear system is considered. It is demonstrated that the learning rate for the delta rule employed at each layer in fact decreases with an increase in the number of neurons in that layer. Finally, it is shown that the improved weight-tuning mechanism with the projection algorithm makes the NN weights bounded, and can allow fast tuning even for large NN.

Note that the NN controllers derived in this chapter or in others require no a priori knowledge of the dynamics of the nonlinear system being controlled, unlike in conventional adaptive control where a regression matrix must

be computed, nor is any learning phase needed. Consider the first-order MIMO discrete-time nonlinear system described by

$$\begin{bmatrix} x_{p1}(k+1) \\ x_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} \frac{x_{p2}}{1+x_{p1}^2} \\ \frac{x_{p1}}{1+x_{p2}^2} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} \quad (8.45)$$

$$y(k) = \begin{bmatrix} y_{p1}(k) \\ y_{p2}(k) \end{bmatrix} = \begin{bmatrix} x_{p1}(k) \\ x_{p2}(k) \end{bmatrix}$$

The objective is to track the output of the linear stable reference model given by

$$\begin{bmatrix} x_{m1}(k+1) \\ x_{m2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 \\ 0.1 & -0.8 \end{bmatrix} + \begin{bmatrix} \bar{r}_1(k) \\ \bar{r}_2(k) \end{bmatrix} \quad (8.46)$$

$$\bar{y}(k) = \begin{bmatrix} y_{m1}(k) \\ y_{m2}(k) \end{bmatrix} = \begin{bmatrix} x_{m1}(k) \\ x_{m2}(k) \end{bmatrix}$$

where $\bar{r}_1(k)$ and $\bar{r}_2(k)$ are the reference inputs. In other words, the aim is to determine the control inputs $u_1(k)$ and $u_2(k)$ such that

$$\lim_{k \rightarrow \infty} \|y(k) - \bar{y}(k)\| \leq \delta \quad \forall k \geq 0 \quad (8.47)$$

The elements in the diagonal gain matrix were chosen as

$$k_v = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad (8.48)$$

and a sampling interval of 10 msec was considered. A three-layer NN was selected with two input, four hidden, and two output nodes. Sigmoidal activation functions were employed in the nodes of the hidden layer. The initial conditions for the plant and the model were chosen to be $[2 - 2]^T$ and $[0.1 0.6]^T$, and the weights were initialized to zero with an initial threshold value of 3.0. No learning is performed initially to train the networks. The design parameter Γ is selected to be 0.01 for all the simulations with all the elements of the design parameter matrix, B_i , chosen to be 0.1. The upper bound on the allowed adaptation gains α_1 , α_2 , and α_3 using (8.23) for the case of the delta rule at each layer is computed to be 1.0, 1.5, and 0.5 respectively. A reference square wave of magnitude 2 units and period of 30 sec is asserted.

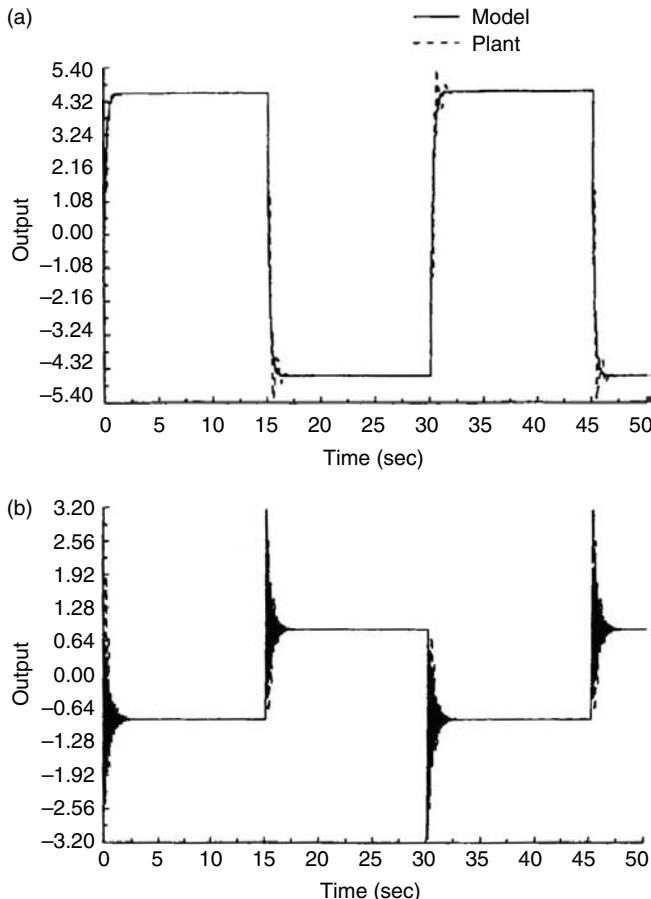


FIGURE 8.2 Response of NN controller with delta rule weight tuning and small α_3 . Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

The adaptation gains for the multilayer NN weight tuning are selected as $\alpha_1 = 0.95$, $\alpha_2 = 0.45$, and $\alpha_3 = 0.1$ for the case of delta rule (8.21) and (8.22). Figure 8.2 shows the excellent tracking response of the NN controller with the delta-rule-based weight tuning at each layer. Figure 8.3 illustrates the response of the NN controller when the delta rule is employed with the adaptation gain α_3 in the last layer changed from 0.1 to 0.6. From Figure 8.3, it is evident that the weight tuning based on the delta rule at each layer becomes unstable at $t = 1.08$ sec. This demonstrates that the adaptation gain in the case of delta rule at each layer must decrease with an increase in the hidden-layer neurons.

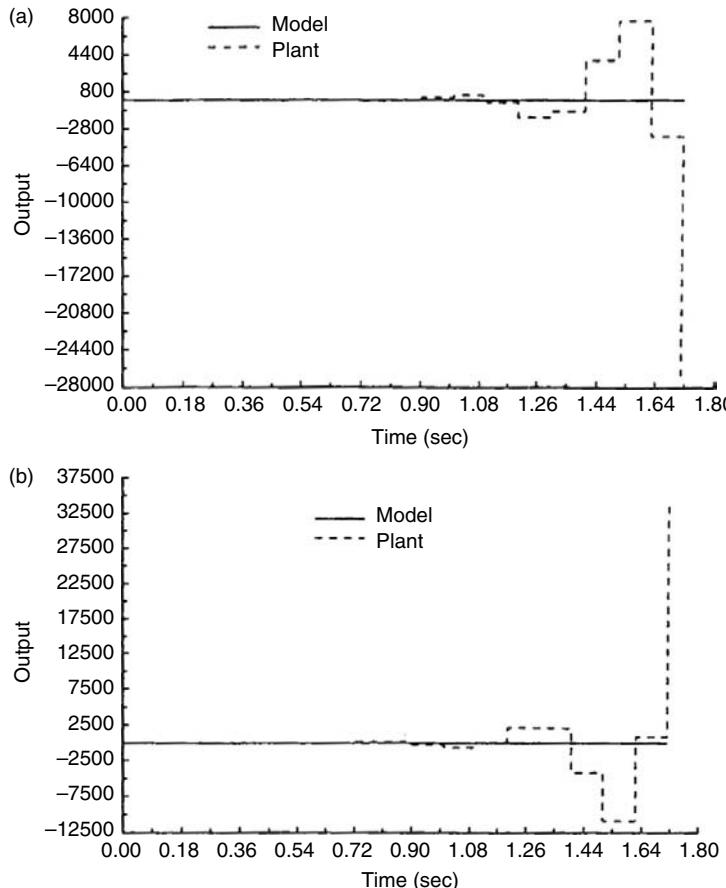


FIGURE 8.3 Response of NN controller with delta rule weight tuning and large α_3 . Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

Consider now the case where the constant learning rate parameter is replaced with the projection algorithm given by (8.43). The adaptation gains are selected to be $\xi_1 = 1.0$, $\xi_2 = 1.0$, and $\xi_3 = 0.7$ with $\zeta_1 = \zeta_2 = \zeta_3 = 0.001$ for the case of projection. Figure 8.4 presents the tracking response of the NN controller with the projection algorithm and it is clear that the controller using the delta rule at each layer performs equally well with the projection algorithm when the value of the adaptation gain is small, so that (8.23) is satisfied. However, note that from Figure 8.4 that due to large adaptation gains for the case of projection algorithm, overshoots and undershoots are observed even though the tracking performance is extremely impressive.

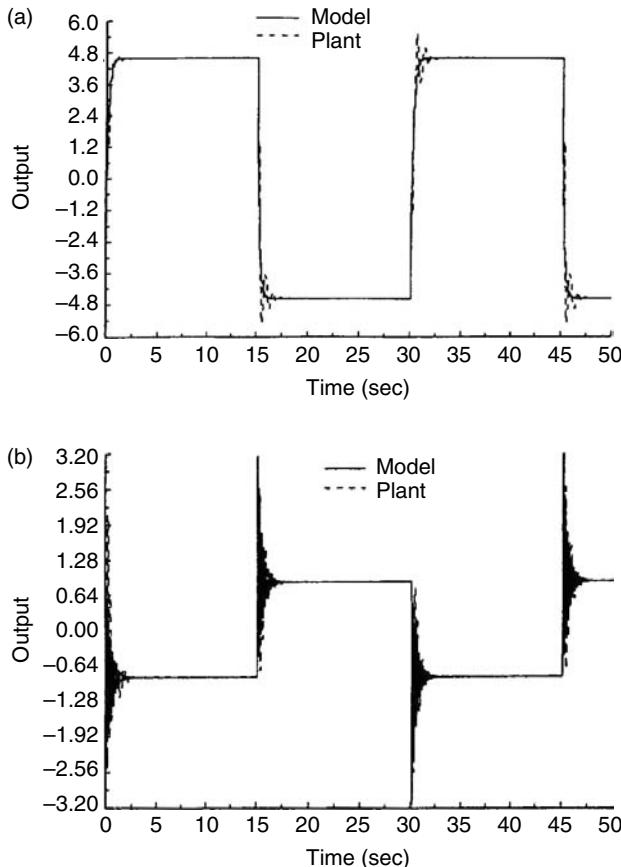


FIGURE 8.4 Response of NN controller with projection algorithm and large α_3 . Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

The performance of the NN controller was investigated while varying the adaptation gains at the output layer for the case of projection algorithm. Figure 8.5 illustrates the tracking response of the NN controller with $\xi_1 = 1.0$, $\xi_2 = 1.0$, and $\xi_3 = 0.7$ with $\zeta_1 = \zeta_2 = \zeta_3 = 0.001$. As expected, the overshoots and undershoots have been totally eliminated but there appears to be a slight degradation in the performance. In other words, at very small adaptation gains, overshoots and undershoots are not seen but there appears a slight degradation in the tracking performance with a slow and smooth convergence. On the other hand, at large adaptation gains, overshoots are observed with a good tracking performance. As the adaptation gains at the output layer are further increased,

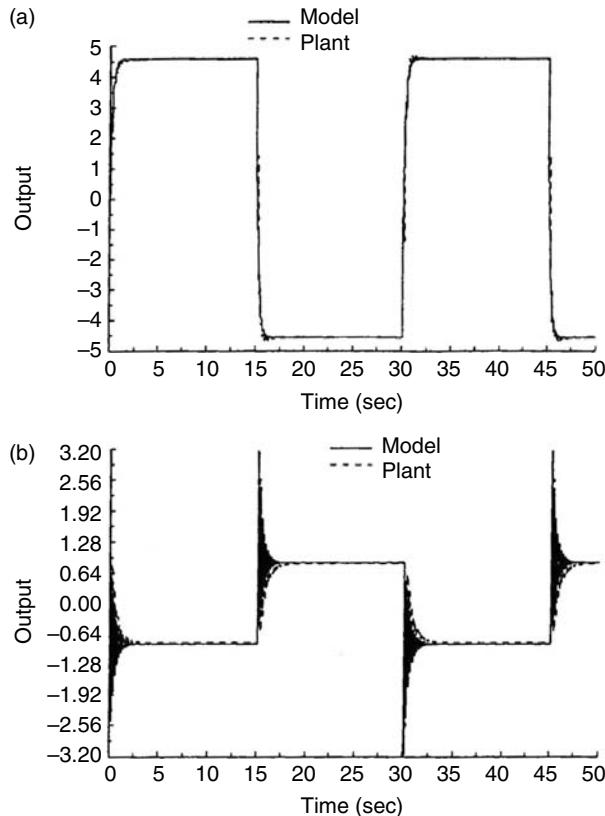


FIGURE 8.5 Response of NN controller with projection algorithm and small α_3 . Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

the oscillatory behavior continues to increase and finally the system becomes unstable. In other words, from the bounds presented in (8.25), as the adaptation gains are increased, the margin of stability continues to decrease, and, at large adaptation gains (close to 1), the system becomes unstable. Thus, the simulation results conducted corroborate the bounds presented in the previous sections.

Let us consider the case when a bounded disturbance given by

$$w(t) = \begin{cases} 0.0, & 0 \leq t < 12 \\ 0.2, & t \geq 12 \text{ (for delta rule)} \\ 0.5, & t \geq 12 \text{ (for projection algorithm)} \end{cases} \quad (8.49)$$

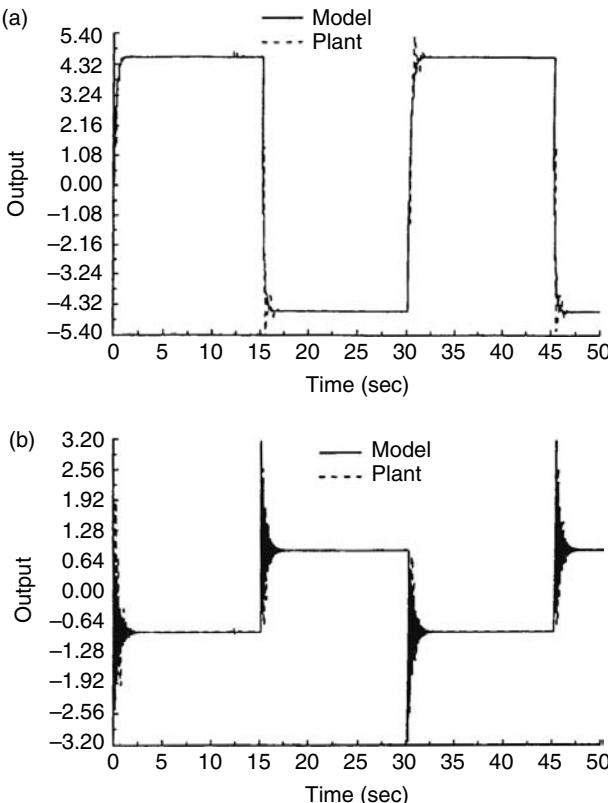


FIGURE 8.6 Response of NN controller with delta rule and small α_3 in the presence of a bounded disturbance. Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

is acting on the plant at the time instant k . Figure 8.6 and Figure 8.7 present the tracking response of NN controllers with the developed tuning based on delta rule and projection algorithm, respectively. Note the magnitude of the disturbance for the case of the projection algorithm is larger than that of the delta rule. In addition, the learning rate for the last layer of the NN in the case of delta rule is selected as 0.1, whereas for the projection algorithm it is chosen as 0.7. For both cases, it can be seen from the figures that the effect of the disturbance can be decoupled from the plant output.

Note that in the case of second output there are some oscillations observed in the output of the plant and the model at the instant when the reference input is asserted. These oscillations at the output of the plant are due to the oscillatory nature of the tracking signal. These oscillations are caused due to the improper

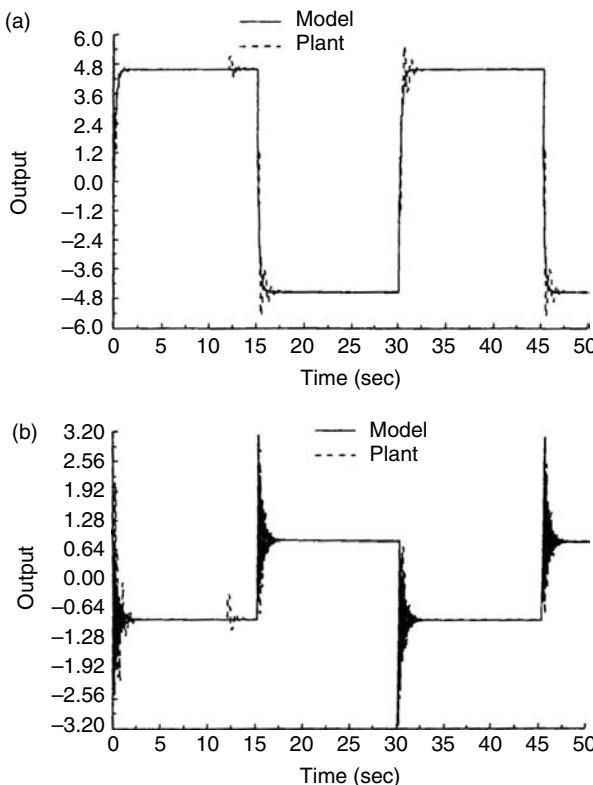


FIGURE 8.7 Response of NN controller with projection algorithm and large α_3 in the presence of a bounded disturbance. Model and plant output: (a) \bar{y}_1 and y_1 ; (b) \bar{y}_2 and y_2 .

choice of the reference model and can be totally eliminated by a proper selection of the parameters in the reference model.

8.4 CONCLUSIONS

A multilayer NN controller was presented for the discrete-time MRAC of nonlinear dynamical systems. The NN weights are tuned online using a modification of the delta rule and have an additional term. In other words, the NN controller exhibits a learning-while-functioning feature instead of learning-then-control. No PE condition is needed and furthermore no certainty equivalence assumption is required. It is assumed that the plant is controllable and state vector of the plant is accessible. The NN controller offers guaranteed performance even when a bounded disturbance is acting on the plant. It was found that the

adaptation gain in the case of delta rule at each layer must decrease with an increase in the number of hidden-layer neurons in that layer so that learning must slow down for large NN, a problem often encountered in the NN literature. The constant learning rate parameters employed in these weight-tuning updates may be modified to obtain a projection algorithm so that the learning rate is independent of the number of hidden-layer neurons.

It is further found by simulation that at low adaptation gains, a smooth and slow convergence is observed with a slight degradation in tracking performance. On the other hand, at large adaptation gains oscillatory behavior is seen with a good tracking performance and faster convergence. Simulation results are also presented when a bounded and a constant disturbance was acting on the system. It can be inferred from these results that the effect of the disturbance can be decoupled from the output of the plant.

REFERENCES

- Åström, K.J. and B. Wittenmark, *Adaptive Control*, Addison-Wesley Company, Reading, MA, 1989.
- Erzberger, H., Analysis and design of model following systems by state space techniques, *Proceedings of the Joint Automatic Control Conference*, Ann Arbor, pp. 572–581, 1968.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Autom. Contr.*, 41, 1693–1699, 1996b.
- Jagannathan, S. and Lewis, F.L., Robust implicit self-tuning regulator: convergence and stability, *Automatica*, 32, 1629–1644, 1996a.
- Jagannathan, S., Lewis, F.L., and Pastravano, O.C., Discrete-time model reference adaptive control of nonlinear dynamical systems using neural networks, *Int. J. Contr.*, 64, 217–239, 1996.
- Lewis, F.L., Abdallah, C.T., and Dawson, D.M., *Control of Robot Manipulators*, Macmillan, New York, 1993.
- Narendra, K.S. and Annaswamy, A.M., A new adaptive law for robust adaptation without persistent excitation, *IEEE Trans. Autom. Contr.*, 32, 134–145, 1987.
- Narendra, K.S. and Annaswamy, A.M., *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Narendra, K.S. and Parthasarathy, K.S., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, 4–27, 1990.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning internal representations by error propagation, in *Readings in Machine Learning*, J.W. Shavlik, Ed.: Morgan Khauffman, San Mateo, CA, pp. 115–137, 1990.
- Sussman, H.J., Uniqueness of the weights for minimal feedforward nets with given input–output map, *Neural Netw.*, 5, 589–593, 1992.
- Widrow, B. and Lehr, M., 30 years of adaptive neural networks: perceptrons, madaline, and backpropagation, *Proc. Inst. Elect. Electron. Eng.*, 78, 1415–1442, 1990.

PROBLEMS

SECTION 8.2

8.2-1: *MRAC using NN.* Consider the first-order MIMO discrete-time nonlinear system described by

$$\begin{bmatrix} x_{p1}(k+1) \\ x_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} \frac{x_{p2} + x_{p1}}{1 + x_{p1}^2} \\ \frac{x_{p2} + x_{p1}}{1 + x_{p2}^2} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$y(k) = \begin{bmatrix} y_{p1}(k) \\ y_{p2}(k) \end{bmatrix} = \begin{bmatrix} x_{p1}(k) \\ x_{p2}(k) \end{bmatrix}$$

The objective is to track the output of the linear stable reference model given by

$$\begin{bmatrix} x_{m1}(k+1) \\ x_{m2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 \\ 0.1 & -0.8 \end{bmatrix} \begin{bmatrix} \bar{r}_1(k) \\ \bar{r}_2(k) \end{bmatrix}$$

$$\bar{y}(k) = \begin{bmatrix} y_{m1}(k) \\ y_{m2}(k) \end{bmatrix} = \begin{bmatrix} x_{m1}(k) \\ x_{m2}(k) \end{bmatrix}$$

where $\bar{r}_1(k)$ and $\bar{r}_2(k)$ are the reference inputs. Design a NN controller.

8.2-2: *MRAC using NN.* Consider the first-order MIMO discrete-time nonlinear system described by

$$\begin{bmatrix} x_{p1}(k+1) \\ x_{p2}(k+1) \end{bmatrix} = \begin{bmatrix} \frac{x_{p2} + x_{p1}}{1 + x_{p1}^2} \\ \frac{1}{1 + x_{p2}^2} \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$y(k) = \begin{bmatrix} y_{p1}(k) \\ y_{p2}(k) \end{bmatrix} = \begin{bmatrix} x_{p1}(k) \\ x_{p2}(k) \end{bmatrix}$$

The objective is to track the output of the linear stable reference model given by

$$\begin{bmatrix} x_{m1}(k+1) \\ x_{m2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 \\ 0.1 & -0.8 \end{bmatrix} \begin{bmatrix} \bar{r}_1(k) \\ \bar{r}_2(k) \end{bmatrix}$$

$$\bar{y}(k) = \begin{bmatrix} y_{m1}(k) \\ y_{m2}(k) \end{bmatrix} = \begin{bmatrix} x_{m1}(k) \\ x_{m2}(k) \end{bmatrix}$$

where $\bar{r}_1(k)$ and $\bar{r}_2(k)$ are the reference inputs. Design a NN controller.

8.2-3: MRAC using NN. Consider the first-order MIMO discrete-time nonlinear system described in Problem 8.2-1.

Now the objective is to track the output of the linear reference model given by

$$\begin{bmatrix} x_{m1}(k+1) \\ x_{m2}(k+1) \end{bmatrix} = \begin{bmatrix} 0.6 & 0.2 \\ 0.1 & 0.8 \end{bmatrix} + \begin{bmatrix} \bar{r}_1(k) \\ \bar{r}_2(k) \end{bmatrix}$$
$$\bar{y}(k) = \begin{bmatrix} y_{m1}(k) \\ y_{m2}(k) \end{bmatrix} = \begin{bmatrix} x_{m1}(k) \\ x_{m2}(k) \end{bmatrix}$$

where $\bar{r}_1(k)$ and $\bar{r}_2(k)$ are the reference inputs. Design a NN controller and discuss the results.

9 Neural Network Control in Discrete-Time Using Hamilton–Jacobi–Bellman Formulation

In the literature, there are many methods of designing stable controllers for nonlinear systems. However, stability is only a bare minimum requirement in a system design. Previous chapters discuss the design of controllers for various classes of nonlinear discrete-time systems using neural networks (NNs). Ensuring optimality guarantees the stability of the nonlinear system; however, optimal control of nonlinear systems is a difficult and challenging area. If the system is modeled by linear dynamics and the cost functional to be minimized is quadratic in state and control, then optimal control is a linear feedback of states, where the gains are obtained by solving a standard Riccati equation (Lewis 1992). On the other hand, if the system is modeled by nonlinear dynamics or the cost functional is nonquadratic, the optimal state feedback control will depend upon obtaining the solution to the Hamilton–Jacobi–Bellman (HJB) equation (Saridis and Lee 1979), which is generally nonlinear. The HJB equation is difficult to solve directly because it involves solving either nonlinear partial difference equations or nonlinear partial differential equations.

To overcome the difficulty in solving the HJB equation, recursive methods are employed to obtain the solution of HJB equation indirectly. Recursive methods involve iteratively solving the generalized HJB (GHJB) equation, which is linear in the cost function of the system, and then updating the control law. It has been demonstrated (Saridis and Lee 1979) in the literature that if the initial control is admissible and the GHJB equation can be solved exactly, the updated control will converge to the optimal control, which is the unique solution to the HJB equation.

There has been a great deal of effort to address this problem in the literature in continuous-time. Approximate HJB solution has been confronted using many techniques by many authors (see Saridis and Lee 1979; Lyshevski 1990; Beard 1995; Bernstein 1995; Bertsekas and Tsitsiklis 1996; Beard et al. 1997; Beard and Saridis 1998; Han and Balakrishnan 2002; Xin and Balakrishnan 2002; Lewis and Abu-Khalaf 2003). In this chapter, we focus on HJB solution using the so-called GHJB equation in discrete-time. For linear systems with quadratic performance indices, the HJB equation becomes the algebraic Riccati equation. Kleinman (1968) pointed out that the solution of the Riccati equation can be obtained by successively solving a sequence of Lyapunov equations, which is linear in the cost functional of the system and thus it is easier to solve when compared to a Riccati equation, which is nonlinear in the cost functional. This idea has been extended (Saridis and Lee 1979) to the case of nonlinear continuous-time systems where a recursive method is used to obtain the optimal control of continuous system by successively solving the GHJB equation and then updating the control if an admissible initial control is given. Although the GHJB equation is linear and easier to solve than HJB equation, no general solution for GHJB has been demonstrated. Galerkin's spectral approximation method is employed in Beard et al. (1997) to find approximate but close solutions to the GHJB in every iteration.

Beard (Beard and Saridis 1998) employed a series of polynomial functions as basic functions to solve the approximate GHJB equation in continuous-time. However, this method requires the computation of a large number of integrals. Interpolating wavelets were used as the basic functions in Park and Tsotoras (2003). Lewis and Abu-Khalaf (2003) employed nonquadratic performance functionals to solve constrained control problems for general affine nonlinear continuous-time systems using based on the work of Lyshevski (1990). In addition, it was also shown how to formulate the associated Hamilton–Jacobi–Isaac (HJI) equation using special nonquadratic supply rates to obtain the nonlinear state feedback H_∞ control (Abu-Khalaf and Lewis 2002).

Since NN can effectively extend adaptive control techniques to nonlinearly parameterized systems, in Miller et al. (1990), Werbos first proposed using NN-based optimal control laws by solving the HJB equation. NN were used by Parisini and Zoppoli (1998) to derive optimal control laws for discrete-time stochastic nonlinear systems. In Lin and Brynes (1996), H_∞ control of discrete-time nonlinear system is presented. In Munos et al. (1999), gradient descent approaches to NN-based solutions of HJB equation are covered. Although many papers have discussed GHJB method for continuous-time systems, there is no reported work on the GHJB method for nonlinear discrete-time systems (Chen and Jagannathan 2005). Discrete-time version of the approximate GHJB equation-based control is important since all the controllers are typically implemented using embedded digital hardware. In this chapter, we will apply

the idea of GHJB equation in discrete-time from Chen and Jagannathan (2005) and set up the practical method for obtaining the nearly optimal control of discrete-time nonlinear systems by assuming that the system dynamics are accurately known. We will use successive approximation techniques in the least-squares sense to solve the GHJB in discrete-time using a quadratic functional. Subsequently, an NN is used to approximate the GHJB equation to show that the result is a closed-loop control based on a NN that has been tuned a priori in off-line mode.

9.1 OPTIMAL CONTROL AND GENERALIZED HJB EQUATION IN DISCRETE-TIME

Consider an affine in the control nonlinear discrete-time dynamic system of the form

$$x(k+1) = f(x(k)) + g(x(k))u(x(k)) \quad (9.1)$$

where $x(k) \in \Omega \subset \mathbb{R}^n$, $u : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$. Assume that $f + gu$ is Lipschitz continuous on a set Ω in \mathbb{R}^n enclosing the origin, and that the system (9.1) is controllable in the sense that there exists a continuous control on Ω that asymptotically stabilizes the system. It is desired to find a control function $u : \mathbb{R}^n \rightarrow \mathbb{R}^m$, which minimizes the generalized quadratic cost functional

$$J(x(0); u) = \sum_{k=0}^{\infty} (x(k)^T Q x(k) + u(x(k))^T R u(x(k))) + \phi(x(\infty)) \quad (9.2)$$

where $Q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a semidefinite monotonically increasing function, R is a symmetric positive definite matrix, $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ is a final state punishment function that is positive definite.

Control objective: The objective is to select the feedback control law $u(x(k))$ in order to minimize the cost functional value.

Remark 1: It is important to note that the control u must both stabilize the system on Ω and make the cost functional value finite so that the control is admissible (Beard 1995).

Definition 9.1.1 (Admissible Controls): Let Ω_u denote the set of admissible controls. A control function $u : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is defined to be admissible with

respect to the state penalty function $(x(k))^T Q(x(k))$ and control energy penalty function $(u(x(k)))^T Q(u(x(k)))$ on Ω , denoted as $u \in \Omega_u$, if:

- u is continuous on Ω
- $u(0) = 0$
- u stabilizes system (9.1) on Ω
- $J(x(0); u) = \sum_{k=0}^{\infty} ((x(k))^T Q x(k)) + u(x(k))^T R u(x(k)))$
 $+ \phi(x(\infty)) < \infty, \quad \forall x(0) \in \Omega$

Remark 2: The admissible control guarantees that the control is stable; but in general any stable control cannot guarantee that it is admissible. For example, consider the nonlinear discrete-time system

$$x(k+1) = \frac{1}{\sqrt{k+1}} + x(k) + u \quad (9.3)$$

A feedback control is given as $u = -x$ and the system solution will be $x(k) = 1/\sqrt{k}$ for $k > 0$ and $x(0)$ for $k = 0$. If we take $V(x) = x^2$ as Lyapunov function, the difference of the Lyapunov function is $V(x(k+1)) - V(x(k)) = (-1/((k+1)k))$ for $k > 1$ and $\frac{1}{2} - x(0)$ for $k = 1$. For $|x(0)| < 0.5$, the nonpositive difference of the Lyapunov function can guarantee that the system is Lyapunov stable. As $k \rightarrow \infty$, $x(k) \rightarrow 0$. This system with this feedback control is asymptotically stable. However, $\|u(k)\|^2 = \|x(k)\|^2$ and the sum $\sum_{k=0}^{\infty} \|u(k)\|^2 = x^2(0) + \sum_{k=1}^{\infty} 1/k$ is infinite. We can conclude that this feedback control is stable but not admissible. Hence, only systems that decay sufficiently fast will be considered here.

Given an admissible control and the state of the system at every instant of time, the performance of this control is evaluated through a cost functional. If the solution of the dynamic system $x(k+1) = f(x(k)) + g(x(k))u(x(k))$ is known and given the cost functional, the overall cost is the sum of the cost value calculated at each time step k . However, for complex nonlinear discrete-time systems, the closed-form solution, $x(k)$, is difficult to determine and the solution can depend upon the initial conditions. Therefore a suitable cost function, which is independent of the solution of the nonlinear dynamic system, $x(k)$, is necessary. In general it is very difficult to select the cost function; however, Theorem 9.1.1 will prove that there exists a positive definite function, $V(x)$, referred as value function, whose initial value, $V(x(0))$, is equal to the cost functional value of J given an initial admissible control and the state of the system.

Theorem 9.1.1: Assume $u_1(x) \in \Omega_u$ is an admissible control law arbitrarily selected. If there exists a positive definite continuously differentiable value

function $V(x)$ on Ω satisfying the following:

$$\begin{aligned} & \frac{1}{2}(f(x) + g(x)u_1(x) - x)^T \cdot \nabla^2 V(x) \cdot (f(x) + g(x)u_1(x) - x) \\ & + \nabla V(x)^T(f(x) + g(x)u_1(x) - x) + x^T Qx + u_1(x)^T R u_1(x) = 0 \end{aligned} \quad (9.4)$$

$$V(x(\infty)) = \phi(x(\infty)) \quad (9.5)$$

where $\nabla V(x)$ and $\nabla^2 V(x)$ are the gradient vector and Hessian matrix of $V(x)$. Then $V(x(j), u_1)$ is the value function of the system defined in (9.1) for all $j = 0, \dots, \infty$ applying the feedback control $u_1(x)$ and

$$V(x(0); u_1) = J(x(0); u_1) \quad (9.6)$$

Proof: The proof uses a linearization notion but one can relax this requirement (Chen and Jagannathan 2005) and the reader is referred to Lin and Brynes (1996) for details. Assume that $V(x(k), u_1) > 0$ exists and continuously differentiable. Then

$$V(x(\infty); u_1) - V(x(j); u_1) = \sum_{k=j}^{\infty} \Delta V(k) \quad (9.7)$$

where $\Delta V(k) = V(x(k+1)) - V(x(k))$ is the difference function of $V(x)$. Note that $V(x(k))$ is a continuously differentiable function therefore expanding the function $V(x)$ using Taylor series about the point of $x(k)$ renders

$$\begin{aligned} V(x(k+1)) &= V(x(k)) + \nabla V(k)^T(x(k+1) - x(k)) \\ &+ \frac{1}{2}(x(k+1) - x(k))^T \nabla^2 V(k)(x(k+1) - x(k)) + \dots \end{aligned} \quad (9.8)$$

where $\nabla V(k)$ is the gradient vector defined as

$$\nabla V(k) = \left. \frac{\partial V(x)}{\partial x} \right|_{x=x(k)} = \left[\left. \frac{\partial}{\partial x_1} V(x), \frac{\partial}{\partial x_2} V(x), \dots, \frac{\partial}{\partial x_n} V(x) \right]^T \right|_{x=x(k)} \quad (9.9)$$

and $\nabla^2 V(k)$ is the Hessian matrix defined as:

$$\nabla^2 V(k) = \left[\begin{array}{cccc} \frac{\partial^2 V(x)}{\partial x_1^2} & \frac{\partial^2 V(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 V(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 V(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 V(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 V(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 V(x)}{\partial x_n \partial x_1} & \frac{\partial^2 V(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 V(x)}{\partial x_n^2} \end{array} \right]_{x=x(k)} \quad (9.10)$$

By assuming a sufficiently small sampling interval, the first three terms of the Taylor series expansion can be considered by ignoring the higher than second order terms to get

$$\begin{aligned} \Delta V(k) \approx & \nabla V(k)^T (x(k+1) - x(k)) + \frac{1}{2} (x(k+1) - x(k))^T \\ & \times \nabla^2 V(k) (x(k+1) - x(k)) \end{aligned} \quad (9.11)$$

From (9.7) and (9.11), using (9.1) we get

$$\begin{aligned} & V(x(\infty); u_1) - V(x(j); u_1) \\ &= \sum_{k=j}^{\infty} \left[\nabla V(k)^T (f(x(k)) + g(x(k))u(x(k)) - x(k)) \right. \\ & \quad \left. + \frac{1}{2} (f(x(k)) + g(x(k))u(x(k)) - x(k))^T \nabla^2 V(k) (f(x(k)) \right. \\ & \quad \left. + g(x(k))u(x(k)) - x(k)) \right] \end{aligned} \quad (9.12)$$

For convenience, we denote

$$\Delta x(k) = f(x(k)) + g(x(k))u(x(k)) - x(k), \quad u(k) = u(x(k)) \quad (9.13)$$

Adding (9.2) on both sides of (9.12) and rewriting (9.12) as

$$\begin{aligned} J(x(j); u_1) - V(x(j); u_1) \\ = \sum_{k=j}^{\infty} [\frac{1}{2} \Delta x(k)^T \nabla^2 V(k) \Delta x(k) + \nabla V(k)^T \Delta x(k) + (x(k))^T Q(x(k)) \\ + u(k)^T R u(k)] + [\phi(x(\infty)) - V(x(\infty))] \end{aligned} \quad (9.14)$$

Because $x(k) \in \Omega$, applying (9.4) and (9.5) into (9.14) renders

$$V(x(j); u_1) = J(x(j); u_1) \quad \text{for } j = 0, \dots, \infty \quad (9.15)$$

More specially for $j = 0$, $V(x(0); u_1) = J(x(0); u_1)$

Remark 3: One can demonstrate the proof even without using the Taylor series expansion. An optimal control function $u^*(x)$ for a nonlinear discrete-time system is the one that minimize the value of function $V(x(0); u^*)$.

Since the Hessian matrix function $\nabla^2 V(x(k))$ is used in the approximation of difference function of $\Delta V(k)$, it is necessary to investigate the property of this matrix function. Lemma 9.2.1 will play an important role in the proof of Theorem 9.2.2.

Lemma 9.1.1: The Hessian matrix $\nabla^2 V(x(k))$ is a semi-positive definite matrix function for any $x(k) \in \Omega$.

Proof: Given $V(x(k); u_1) = J(x(k); u_1) = \sum_{j=k}^{\infty} x(j)^T Q x(j) + u(x(j))^T R u(x(j))$, we get $\nabla^2 V(x(k)) = 2Q + 2(\nabla^2 u(k))^T R (\nabla^2 u(k))$ where $\nabla^2 u(k) = (\partial^2 u(x)/\partial x^2)|_{x=x(k)}$. For any nonzero vector $v \in R^n$, we have $v^T \nabla^2 V(x(k)) v = 2v^T Q v + 2v^T (\nabla^2 u(k))^T R (\nabla^2 u(k)) v = 2v^T Q v + 2(\nabla^2 u(k) v)^T R (\nabla^2 u(k) v)$. Since R , Q are positive definite matrices and $v^T Q v \geq 0$, $v^T (\nabla^2 u(k) v)^T R (\nabla^2 u(k) v) \geq 0$, we can obtain $v^T \nabla^2 V(x(k)) v \geq 0$. So $\nabla^2 V(x(k))$ is a semi-positive definite matrix function for $x(k) \in \Omega$.

Definition 9.1.2 (GHJB Equation in Discrete-Time): The GHJB equation can be defined as

$$\frac{1}{2} \Delta x^T \nabla^2 V(x) \Delta x + \nabla V(x)^T \Delta x + x^T Q x + u(x)^T R u(x) = 0 \quad (9.16)$$

$$V(0) = 0 \quad (9.17)$$

where $\Delta x = f(x) + g(x)u(x) - x$.

In this chapter, the infinite-time optimal control problem for the nonlinear discrete-time system (9.1) is attempted. The cost functional of the infinite-time problem for the discrete-time system is defined as

$$J(x(0); u) = \sum_{k=0}^{\infty} (x(k)^T Q x(k) + u(x(k))^T R u(x(k))) \quad (9.18)$$

The GHJB equation (9.16) with the boundary condition (9.17) can be used for infinite-time problems, because as $N \rightarrow \infty$, $x(\infty) = 0$, $V(0) = V(x(\infty)) = \phi(x(\infty)) = 0$. So if an admissible control is specified, for any infinite-time problem, we can solve the GHJB equation to obtain the value function $V(x)$ which in turn can be used in the cost functional, J , along with $V(x(0))$ to calculate the cost of admissible control.

We already know how to evaluate the performance of the current admissible control, but that is not our final goal. Our objective is to improve the performance of the system over time by updating the control so that a near-optimal controller results. Besides deriving an updated control law, it is required that the updated control function renders admission control inputs to the nonlinear discrete-time system while ensuring that the performance is enhanced over time. The updated control function is obtained by minimizing a certain pre-Hamiltonian function. In fact, Theorem 9.1.2 demonstrates that if the control function is updated by minimizing the pre-Hamiltonian function defined in (9.19), then the system performance can be enhanced over time while guaranteeing that the updated control function is admissible for the original nonlinear system (9.1). Next, the pre-Hamiltonian function for the discrete-time system is introduced.

Definition 9.1.3 (Pre-Hamiltonian Function): A suitable pre-Hamiltonian function for the nonlinear system (9.1) is defined by

$$H(x, V(x), u(x)) = \frac{1}{2} \Delta x^T \nabla^2 V(x) \Delta x + \nabla V(x)^T \Delta x + x^T Q x + u(x)^T R u(x) \quad (9.19)$$

It is important to note that the pre-Hamiltonian is a nonlinear function of the state, value, and control functions. If a control function $u^{(i)} \in \Omega_u$ and value function, $V^{(i)}$, satisfies $\text{GHJB}(V^{(i)}, u^{(i)}) = 0$, an updated control function u^{i+1} can be obtained by differentiating the pre-Hamiltonian function (9.19) associated with the value function $V^{(i)}$. In other words, the updated control function can be obtained by solving

$$\frac{\partial H(x, V^{(i)}(x), u^{(i+1)})}{\partial u^{(i+1)}} = 0 \quad (9.20)$$

and it is given by

$$\begin{aligned} u^{(i+1)}(x) = & -[g^T(x)\nabla^2 V^{(i)}(x)g(x) + 2R]^{-1}g^T(x) \\ & \times (\nabla V^{(i)}(x) + \nabla^2 V^{(i)}(x)(f(x) - x)) \end{aligned} \quad (9.21)$$

The next theorem demonstrates that the updated control function is indeed admissible for the nonlinear discrete-time system described by (9.1).

Theorem 9.1.2 (Improved Control): If $u^{(i)} \in \Psi(\Omega)$, and $V^{(i)}$ satisfies $\text{GHJB}(V^{(i)}, u^{(i)}) = 0$ with the boundary condition $V^{(i)}(0) = 0$, then the updated control function derived in (9.21) by using the pre-Hamiltonian results in an admissible control for the system (9.1) on Ω . Moreover, if $V^{(i+1)}$ is the unique positive definite function satisfying $\text{GHJB}(V^{(i+1)}, u^{(i+1)}) = 0$, then

$$V^{(i+1)}(x(0)) \leq V^{(i)}(x(0)) \quad (9.22)$$

Proof of Admissibility: First of all, we should investigate the Lyapunov stability of the system with the control $u^{(i+1)}$. Taking the difference of $V^{(i)}(k)$ along the system $(f, g, u^{(i+1)})$ trajectories to obtain

$$\begin{aligned} \Delta V^{(i)}(k) = & V^{(i)}(k+1) - V^{(i)}(k) \approx \nabla V^{(i)}(k)(f(k) + g(k)u^{(i+1)}(k) - x(k)) \\ & + \frac{1}{2}(f(k) + g(k)u^{(i+1)}(k) - x(k))^T \nabla^2 V^i(k) \\ & \times (f(k) + g(k)u^{(i+1)}(k) - x(k)) \end{aligned} \quad (9.23)$$

Rewriting the GHJB equation $\text{GHJB}(V^{(i)}, u^{(i)}) = 0$

$$\begin{aligned} \frac{1}{2}(f(k) + g(k)u^{(i)}(k) - x(k))^T \nabla^2 V^i(k)(f(k) + g(k)u^{(i)}(k) - x(k)) \\ + \nabla V^{(i)T}(f(k) + g(k)u^{(i)}(k) - x(k)) + x(k)^T Qx(k) + u(k)^{(i)T} Ru^{(i)}(k) = 0 \end{aligned} \quad (9.24)$$

Substituting (9.24) into (9.23), Equation 9.23 can be rewritten as

$$\begin{aligned} \Delta V^{(i)}(k) = & -\frac{1}{2}u^{(i)}(k)^T(g(k)^T \nabla^2 V^i(k)g(k) + 2R)u^{(i)}(k) - x(k)^T Qx(k) \\ & + (\nabla V^{(i)}(k)^T + (f(k) - x(k))^T \nabla^2 V^i(k))g(k)(u^{(i+1)}(k) - u^{(i)}(k)) \\ & + \frac{1}{2}u^{(i+1)}(k)^T(g(k)^T \nabla^2 V^i(k)g(k))u^{(i+1)}(k) \end{aligned} \quad (9.25)$$

Substituting (9.20) into (9.25), the difference is obtained as

$$\begin{aligned}\Delta V^{(i)}(k) &= -x(k)^T Q x(k) - u^{(i+1)}(k)^T R u^{(i+1)}(k) - \frac{1}{2}(u^{(i+1)}(k) - u^{(i)}(k))^T \\ &\quad \times (g(k)^T \nabla^2 V^i(k) g(k) + 2R)(u^{(i+1)}(k) - u^{(i)}(k))\end{aligned}\quad (9.26)$$

Since $g(k)^T \nabla^2 V^i(k) g(k) + 2R$, R and Q are positive definite matrixes, we can get

$$\Delta V^{(i)}(k) \leq -x(k)^T Q x(k) \leq -\lambda_{\min}(Q) \|x(k)\|^2 \quad (9.27)$$

This implies that the difference of $V^{(i)}(k)$ along the system $(f, g, u^{(i+1)})$ trajectories is negative for $x(k) \neq 0$. Thus $V^{(i)}(k)$ is a Lyapunov function for $u^{(i+1)}$ on Ω and the system with feedback control $u^{(i+1)}$ is locally asymptotically stable.

Second, we need to prove that the cost function of the system with the updated control $u^{(i+1)}$ is finite. Since $u^{(i)}$ is an admissible control, from Definition 9.1.1 and (9.15) we can have

$$V^{(i)}(x(0)) = J(x(0); u^{(i)}) < \infty \quad \text{for } x(0) \in \Omega \quad (9.28)$$

The cost function for $u^{(i+1)}$ can be written as

$$J(x(0); u^{(i+1)}) = \sum_{k=0}^{\infty} ((x(k)^T Q x(k)) + u^{(i+1)}(k)^T R u^{(i+1)}(k)) \quad (9.29)$$

where $x(k)$ is the trajectory of system with admission control $u^{(i+1)}$. From (9.26) and (9.29), we have

$$\begin{aligned}V^{(i)}(x(\infty)) - V^{(i)}(x(0)) &= \sum_{k=0}^{\infty} \Delta V^{(i)}(k) \\ &= \sum_{k=0}^{\infty} -x(k)^T Q x(k) - u^{(i+1)}(k)^T R u^{(i+1)}(k) - \frac{1}{2}(u^{(i+1)}(k) - u^{(i)}(k))^T \\ &\quad \times (g(k)^T \nabla^2 V^i(k) g(k) + 2R)(u^{(i+1)}(k) - u^{(i)}(k)) \\ &= -J(x(0); u^{(i+1)}) - \frac{1}{2} \sum_{k=0}^{\infty} (u^{(i+1)}(k) - u^{(i)}(k))^T \\ &\quad \times (g(k)^T \nabla^2 V^i(k) g(k) + 2R)(u^{(i+1)}(k) - u^{(i)}(k))\end{aligned}\quad (9.30)$$

Since $x(\infty) = 0$ and $V^{(i)}(0) = 0$, we get $V^{(i)}(x(\infty)) = 0$. Rewriting (9.30) we can have

$$\begin{aligned} J(x(0); u^{(i+1)}) &= V^{(i)}(x(0)) - \frac{1}{2} \sum_{k=0}^{\infty} (u^{(i+1)}(k) - u^{(i)}(k))^T \\ &\quad \times (g(k)^T \nabla^2 V^i(k) g(k) + 2R)(u^{(i+1)}(k) - u^{(i)}(k)) \end{aligned} \quad (9.31)$$

From (9.28) and (9.31), considering that $g(k)^T \nabla^2 V^i(k) g(k) + 2R$ is a positive definite matrix function we can obtain

$$J(x(0); u^{(i+1)}) \leq V^{(i)}(x(0)) < \infty \quad (9.32)$$

Third, since $V^{(i)}$ is continuously differentiable and $g : \Re^m \rightarrow \Re^n$ is a Lipschitz continuous function on the set Ω in \Re^n , the new control law $u^{(i+1)}$ is continuous. Since $V^{(i)}$ is positive definite function, it attains a minimum at the origin, and thus, $\nabla V^{(i)}(x)$ and $\nabla^2 V^{(i)}(x)$ must vanish at the origin. This implies that $u^{(i+1)}(0) = 0$.

Finally, following Definition 9.1.1, one can conclude that the updated control function $u^{(i+1)}$ is admissible on Ω .

Proof of the Improved Control: To show the second part of Theorem 9.1.2, we need to prove that $V^{(i)}(x(0)) \leq V^{(i+1)}(x(0))$ which means the cost function will be reduced by updating the feedback control. Because $u^{(i+1)}$ is an admissible control, there exists a positive definite function $V^{(i+1)}$ such that $\text{GHJB}(V^{(i+1)}, u^{(i+1)}) = 0$ on $x \in \Omega$. According to Theorem 9.1.1, we can get

$$V^{(i+1)}(x(0)) = J(x(0); u^{(i+1)}) \quad (9.33)$$

From Equation 9.31 and Equation 9.33, we know that

$$\begin{aligned} V^{(i+1)}(x(0)) - V^{(i)}(x(0)) &= -\frac{1}{2} \sum_{k=0}^{\infty} (u^{(i+1)}(k) - u^{(i)}(k))^T (g(k)^T \nabla^2 V^i(k) g(k) + 2R) \\ &\quad \times (u^{(i+1)}(k) - u^{(i)}(k)) \leq 0 \end{aligned} \quad (9.34)$$

Theorem 9.1.2 suggests that after solving the GHJB equation and updating the control function by using (9.21), the system performance can be improved. If the control function is iterated successively, the updated control will converge close

to the solution of HJB, which then renders the optimal control function. The GHJB becomes the HJB equation on substitution of the optimal control function $u^*(x)$. The HJB equation can now be defined in discrete-time as follows:

Definition 9.1.4 (HJB Equation in Discrete-Time): The HJB equation in discrete-time can be expressed as

$$\begin{aligned} & \frac{1}{2}(f(x) + g(x)u^*(x) - x)^T \nabla^2 V^*(x)(f(x) + g(x)u^*(x) - x) + \nabla V^*(x)^T \\ & \times (f(x) + g(x)u^*(x) - x) + x^T Qx + u^*(x)^T R u^*(x) = 0 \end{aligned} \quad (9.35)$$

$$V^*(0) = 0 \quad (9.36)$$

where the optimal control function is given by

$$\begin{aligned} u^*(x) = & -[g^T(x)\nabla^2 V^*(x)g(x) + 2R]^{-1}g^T(x)(\nabla V^*(x) \\ & + \nabla^2 V^*(x)(f(x) - x)) \end{aligned} \quad (9.37)$$

Note V^* is the unique optimal solution to the HJB equation (9.35). It is important to note that the GHJB is linear in the value function derivative while the HJB equation is nonlinear in the value function derivative, similar to the case of GHJB in continuous-time. Solving the GHJB equation requires solving linear partial difference equations, while the HJB equation solution involves nonlinear partial difference equations, which may be difficult to solve. This is the reason for introducing the successive approximation technique using GHJB. In the successive approximation method, one solves (9.16) for $V(x)$ given a stabilizing control $u(k)$ then finds an improved control based on $V(k)$ using (9.21). In the following, Corollary 9.1.1 indicates that if the initial control function is admissible, then repetitive application of (9.16), (9.21) is a contraction map, and that the sequence of solutions $V^{(i)}$ converges to the optimal HJB solution $V^*(k)$.

Corollary 9.1.1 (Convergence of Successive Approximations): Given an initial admissible control, $u^0(x) \in \Psi(\Omega)$, by iteratively solving GHJB equation and updating the control function using (9.21), the sequence of solutions $V^i(x)$ will converge to the optimal HJB solution $V^*(x)$.

Proof: From the proof of Theorem 9.1.2, it is clear that after iteratively solving the GHJB equation and updating the control, the sequence of solutions $V^{(i)}$ is a decreasing sequence with a lower bound. Since $V^{(i)}$ is a positive definite function, $V^{(i)} > 0$ and $V^{(i+1)} - V^{(i)} \leq 0$, the sequence of solutions $V^{(i)}$ will converge to a positive definite function $V^{(i)} = V^{(i+1)} = V^d$ when $i \rightarrow \infty$.

Due to the uniqueness of solutions of the HJB equation (Saridis and Lee 1979), now it is necessary to show that $V^d = V^*$.

When $V^{(i)} = V^{(i+1)} = V^d$, from (9.38) we can only get $u^{(i)} = u^{(i+1)}$. Using (9.34) and taking $u^{(i)} = u^{(i+1)}$ to get

$$\begin{aligned} u^{(i)}(x) &= u^{(i+1)}(x) \\ &= -[g^T(x)\nabla^2 V^{(i)}(x)g(x) + 2R]^{-1}g^T(k)(\nabla V^{(i)}(x) + \nabla^2 V^{(i)}(x) \\ &\quad \times (f(x) - x)) \end{aligned} \quad (9.38)$$

The GHJB equation for $u^{(i)}$ can now be expressed as

$$\begin{aligned} \frac{1}{2}(f(x) + g(x)u^{(i)}(x) - x)^T \nabla^2 V^{(i)}(x)(f(x) + g(x)u^{(i)}(x) - x) + \nabla V^{(i)}(x)^T \\ \times (f(x) + g(x)u^{(i)}(x) - x) + x^T Qx + u^{(i)}(x)^T R u^{(i)}(x) = 0 \end{aligned} \quad (9.39)$$

$$V^{(i)}(0) = 0 \quad (9.40)$$

From (9.39) and (9.40), we can conclude that this set of equations is nothing but the well-known HJB equation, which is presented in Definition 9.1.4. This in turn implies that $V^{(i)}(k) \rightarrow V^*$ and $u^{(i)} \rightarrow u^*$ as $i \rightarrow \infty$.

Remark 4: The algebraic Riccati equation (ARE) associated with optimal control for linear discrete-time system can be derived from the HJB equation in discrete-time.

Consider the following linear discrete-time system and cost function defined in (9.21)

$$x(k+1) = Ax(k) + Bu(k) \quad (9.41)$$

We take $V^*(x) = x^T Px$, where P is a symmetric positive definite matrix. The gradient vector and Hessian matrix of $V^*(x)$ can be derived as $\nabla^2 V^*(x) = 2P$ and $\nabla V^*(x) = 2Px$. The HJB equations (9.35) and (9.37) can be rewritten as

$$\begin{aligned} (Ax + Bu^*(x) - x)^T P(Ax + Bu^*(x) - x) + 2x^T P(Ax + Bu^*(x) - x) \\ + x^T Qx + u^*(x)^T R u^*(x) = 0 \end{aligned} \quad (9.42)$$

$$u^*(x) = -[2B^T PB + 2R]^{-1}B^T(2Px + 2P(Ax - x)) \quad (9.43)$$

After simplifying (9.42) and (9.43), we can obtain

$$P = Q + A^T PA - A^T PB(R + B^T PB)^{-1} B^T PA \quad (9.44)$$

$$u^*(x) = -[B^T PB + R]^{-1} B^T Px \quad (9.45)$$

Equation 9.44 is nothing but ARE (Lewis 1992) for linear discrete-time system and Equation 9.45 is the optimal control of linear discrete-time system.

9.2 NN LEAST-SQUARES APPROACH

In the previous section, we have described that by recursively solving the GHJB equation and updating the control function, we could improve the closed-loop performance of control laws that are known to be admissible. Furthermore, we can get arbitrarily close to the optimal control by iterating the GHJB solution, a sufficient number of times. Although the GHJB equation is in theory easier to solve than the HJB equation, there is no general closed-form solution to this equation. Beard et al. (1997) used Galerkin's spectral method to get an approximate solution to (9.16) in continuous-time at each iteration and the convergence is shown in the overall run. This technique does not set the GHJB equation to zero for every iteration but sets it to a residual error instead. The Galerkin spectral method requires the computation of a large number of integrals in order to minimize this residual error.

The purpose of this section is to show how we approximate the solution of the GHJB equation in discrete-time, using NN such that the controls, which result from the solution, are in feedback form. It is well known that NN can be used to approximate smooth functions on a prescribed compact set (Lewis et al. 1999). We approximate $V(x)$ with a NN

$$V_L(x) = \sum_{j=1}^L w_j \sigma_j(x) = W_L^T \bar{\sigma}_L(x) \quad (9.46)$$

where the activation function vector $\sigma_j(x) : \Omega \rightarrow \Re$, is continuous, $\sigma_j(0) = 0$ and the NN weights are w_j , and L is the number of hidden-layer neurons. The vectors $\bar{\sigma}_L(x) \equiv [\sigma_1(x), \sigma_2(x), \dots, \sigma_L(x)]^T$ and $W_L \equiv [w_1, w_2, \dots, w_L]^T$ are the vectors of the activation function and the NN weight matrix, respectively. The NN weights will be tuned to minimize the residual error in a least-squares sense over a set of points within the stability region of the initial stabilizing control. Least squares solution (Beard 1995) attains the lowest possible residual error with respect to the NN weights.

For the GHJB(V, u) = 0, V is replaced by V_L having a residual error as

$$\text{GHJB} \left(V_L = \sum_{j=1}^L w_j \sigma_j, u \right) = e_L(x) \quad (9.47)$$

To find the least-squares solution, the method of weighted residuals is used (Finlayson 1972). The weights w_j are determined by projecting the residual error onto $\partial(e_L(x))/\partial W_L$ and setting the result to zero $\forall x \in \Omega$, that is,

$$\left\langle \frac{\partial(e_L(x))}{\partial W_L}, e_L(x) \right\rangle = 0 \quad (9.48)$$

When expanded, the above equation becomes

$$\begin{aligned} & \langle \nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L \Delta x, \nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L \Delta x \rangle W_L \\ & + \langle x^T Q x + u^T R u, \nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L \Delta x \rangle = 0 \end{aligned} \quad (9.49)$$

where $\nabla \bar{\sigma}_L = [\partial \sigma_1(x)/\partial x, \partial \sigma_2(x)/\partial x, \dots, \partial \sigma_L(x)/\partial x]^T$, $\nabla^2 \bar{\sigma}_L = [\partial^2 \sigma_1(x)/\partial x^2, \partial^2 \sigma_2(x)/\partial x^2, \dots, \partial^2 \sigma_L(x)/\partial x^2]^T$, and $\Delta x = f + gu - x$. In order to proceed, the following technical results are needed.

Lemma 9.2.1: If the set $\{\sigma_j(x)\}_1^L$ is linearly independent and $u \in \Omega_u$, then the set

$$\{\nabla \sigma_j^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \sigma_j \Delta x\}_1^L \quad (9.50)$$

is also linearly independent.

Proof: Calculating the $\sigma(x(j))$ along the system trajectories (f, gu) for $x(j) \in \Omega$ by using the similar formulation of (9.12), we have

$$\sigma_j(x(\infty)) - \sigma_j(x(0)) = \sum_{k=0}^{\infty} \left[\nabla \sigma_j(k)^T \Delta x(k) + \frac{1}{2} \Delta x(k)^T \nabla^2 \sigma_j(k) \Delta x(k) \right] \quad (9.51)$$

Since $u \in \Omega_u$ is an admissible control, the system (f, gu) is stable and $x(\infty) = 0$. With the condition on the active function, $\sigma(0) = 0$, we have

$\sigma(x(\infty)) = 0$. Rewriting (9.50) with the above results, we have

$$\sigma(x(0)) = - \sum_{k=0}^{\infty} \left[\nabla \sigma_j(k)^T \Delta x(k) + \frac{1}{2} \Delta x(k)^T \nabla^2 \sigma_j(k) \Delta x(k) \right] \quad (9.52)$$

Extending (9.52) into the vector formulation

$$\bar{\sigma}_L(x(0)) = - \sum_{k=0}^{\infty} \left[\nabla \bar{\sigma}_L(k)^T \Delta x(k) + \frac{1}{2} \Delta x(k)^T \nabla^2 \bar{\sigma}_L(k) \Delta x(k) \right] \quad (9.53)$$

Now suppose that Lemma 9.2.1 is not true. Then there exists a nonzero $C \in \Re^L$ such that

$$C^T (\nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L(k) \Delta x) \equiv 0 \quad \text{for } x \in \Omega \quad (9.54)$$

From (9.53) and (9.54), we have

$$C^T \bar{\sigma}_L(x(0)) = - \sum_{k=0}^{\infty} \left[C^T (\nabla \bar{\sigma}_L(k) \Delta x(k) + \frac{1}{2} \Delta x(k)^T \nabla^2 \bar{\sigma}_L(k) \Delta x(k)) \right] \equiv 0 \\ \text{for } x(0) \in \Omega \quad (9.55)$$

which contradicts the linear independence of $\{\sigma_j(x)\}_1^L$. So the set (9.50) must be linearly independent.

From Lemma 9.2.1, Equation 9.49 can be rewritten, after defining $\{\nabla \sigma_j^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \sigma_j \Delta x\}_1^L \stackrel{\Delta}{=} \{\theta_j\}_1^L \stackrel{\Delta}{=} \bar{\theta}$, as

$$W_L = -\langle \bar{\theta}, \bar{\theta} \rangle^{-1} \langle Q + u^T R u, \bar{\theta} \rangle \quad (9.56)$$

Because of Lemma 9.2.1, the term $\langle \bar{\theta}, \bar{\theta} \rangle$ is full rank, and is thus invertible. Therefore, a unique solution for W_L exists.

From (9.56), we need to calculate the inner product of $\langle f(x), g(x) \rangle$. In Hilbert space, we define the inner product as

$$\langle f(x), g(x) \rangle = \int_{\Omega} f(x) g(x) dx \quad (9.57)$$

Executing the integration in (9.57) is computationally expensive. However, the integration can be approximated to a suitable degree by using the Riemann

definition of integration so that the inner product can be obtained. This in turn results in a nearly optimal and computationally tractable algorithm.

Lemma 9.2.2 (Riemann Approximation of Integrals): An integral can be approximated as

$$\int_a^b f(x)dx = \lim_{\|\delta x\| \rightarrow 0} \sum_{i=1}^n f(\bar{x}_i) \cdot \delta x \quad (9.58)$$

where $\delta x = x_i - x_{i-1}$ and f is bounded on $[a, b]$ (Burk 1998).

Introducing a mesh on Ω , with mesh size equal to δx , which is taken very small, we can rewrite some terms in (9.49) as follows:

$$X = [\nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L \Delta x|_{x=x_1} \cdots \nabla \bar{\sigma}_L \Delta x + \frac{1}{2} \Delta x^T \nabla^2 \bar{\sigma}_L \Delta x|_{x=x_p}]^T \quad (9.59)$$

$$Y = \begin{bmatrix} x^T Q x + u(x)^T R u(x)|_{x=x_1} \\ \vdots \\ x^T Q x + u^T(x) R u(x)|_{x=x_p} \end{bmatrix} \quad (9.60)$$

where p in x_p represents the number of points of the mesh, which increases as the mesh size is reduced.

Using Lemma 9.2.2, we can rewrite (9.49) as

$$XW_L + Y = 0 \quad (9.61)$$

This implies that we can calculate

$$W_L = -(X^T X)^{-1} (XY) \quad (9.62)$$

An interesting observation is that Equation 9.62 is the standard least-squares method (LSM) of estimation for a mesh on Ω . Note that the mesh size δx should be such that the number of points p is greater or equal to the order of the approximation L and the activation functions should be linearly independent. These conditions guarantee a full rank for $(X^T X)$. The suboptimal control of nonlinear discrete-time system is summarized in Table 9.1.

TABLE 9.1**Suboptimal GHJB-Based Control of Nonlinear Discrete-Time Systems**

The suboptimal control of nonlinear discrete-time system can be obtained off-line as:

1. Define a NN as $V = \sum_{j=1}^L w_j \sigma_j(x)$ to approximate smooth function of $V(x)$.
 2. Select an admissible feedback control law u_1 .
 3. Find $V^{(1)}$ associated with u_1 to satisfy GHJB by applying LSA to obtain the NN weights W^1 using (9.62)
 4. Update the control as
- $$u^{(2)}(x) = -[g^T(x)\nabla^2 V^{(1)}(x)g(x) + 2R]^{-1} g^T(x)(\nabla V^{(1)}(x) + \nabla^2 V^{(1)}(x)(f(x) - x))$$
5. Find $V^{(2)}$ associated with u_2 to satisfy GHJB by using LSA to obtain W^2 .
 6. If $V^1(0) - V^2(0) \leq \varepsilon$, where ε is a small positive constant, then $V^* = V^{(1)}$ and stop. Otherwise, go back to step 4 by increasing the index by one.

After we obtain V^* , the optimal state feedback control, which can be implemented online, can be described as

$$u^*(x) = -[g^T(x)\nabla^2 V^*(x)g(x) + 2R]^{-1} g^T(x)(\nabla V^*(x) + \nabla^2 V^*(x)(f(x) - x))$$

9.3 NUMERICAL EXAMPLES

The power of the technique is demonstrated in the case of HJB by using three examples. First, we consider a linear discrete-time system to compare the performance of the proposed approach to that of the standard solution obtained by solving the Riccati equation. This comparison will present that the proposed approach works for a linear system and renders an optimal solution. Second, we will use a general nonlinear practical system and a real-world two-link planar revolute–revolute (RR) robot arm system to demonstrate that the proposed approach renders a suboptimal solution for nonlinear discrete-time systems.

In all of the examples that we present in this section, the basis functions required will be obtained from even polynomials so that the NN can approximate the positive definite function $V(x)$. If the dimension of the system is n and the order of approximation is M , then we use all of the terms in expansion of the polynomial (Beard 1995)

$$\sum_{j=1}^{M/2} \left(\sum_{k=1}^n x_k \right)^{2j} \quad (9.63)$$

The resulting basis functions for a two-dimensional system is

$$\{x_1^2, x_1 x_2, x_2^2, x_1^4, x_1^3 x_2, \dots, x_2^M\} \quad (9.64)$$

Example 9.3.1 (Linear Discrete-Time System): Consider the linear discrete-time system given in state space formulation

$$x(k+1) = Ax(k) + Bu(k) \quad (9.65)$$

where

$$A = \begin{bmatrix} 0 & -0.8 \\ 0.8 & 1.8 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad (9.66)$$

Define the cost functional

$$J(u; x(0)) = \sum_{k=0}^N (x_1(k)^2 + x_2(k)^2 + u(k)^2) \quad (9.67)$$

Define the NN with the activation functions containing polynomial functions up to the sixth order of approximation by using $n = 2$ and $M = 6$. From (9.63), the NN can be constructed as

$$\begin{aligned} V(x_1, x_2) = & w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1^4 + w_5 x_2^4 + w_6 x_1^3 x_2 + w_7 x_1^2 x_2^2 \\ & + w_8 x_1 x_2^3 + w_9 x_1^6 + w_{10} x_2^6 + w_{11} x_1^5 x_2 + w_{12} x_1^4 x_2^2 + w_{13} x_1^3 x_2^3 \\ & + w_{14} x_1^2 x_2^4 + w_{15} x_1 x_2^5 \end{aligned} \quad (9.68)$$

Select the initial control law $u_1(k) = -0.5x_1(k) + 0.3x_2(k)$, which is admissible and the updating rule

$$u_{i+1} = -[B^T \nabla^2 V^{(i)}(k) B + 2]^{-1} B^T (\nabla V^{(i)}(k) + \nabla^2 V^{(i)}(k)(A - I)x(k)) \quad (9.69)$$

where u_i and v^i satisfy the GHJB equation

$$\begin{aligned} \frac{1}{2} (Ax + Bu^{(i)}(x) - x)^T \nabla^2 V^{(i)}(x) (Ax + Bu^{(i)}(x) - x) + \nabla V^{(i)}(x)^T \\ \times (Ax + Bu^{(i)}(x) - x) + x^T x + u^{(i)}(x)^2 = 0 \end{aligned} \quad (9.70)$$

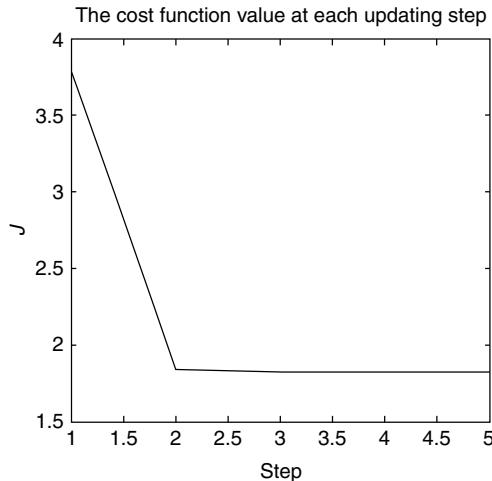


FIGURE 9.1 Cost function J^i with steps.

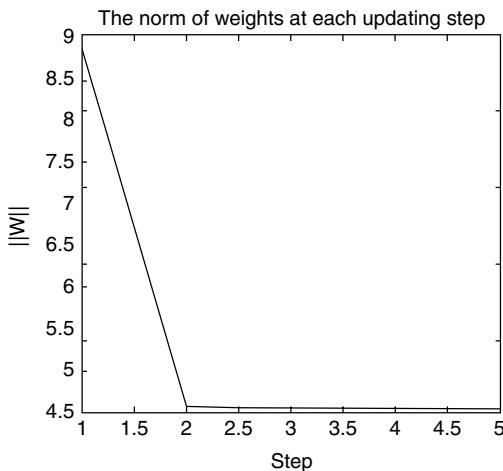


FIGURE 9.2 NN weight norm at each updating step.

In the simulation, the mesh size Δx is selected as 0.01, the AS region is chosen for the states as $-0.5 \leq x_1 \leq 0.5$, and $-0.5 \leq x_2 \leq 0.5$. The small positive approximation error constant is selected as $\varepsilon = 0.00001$. The initial states are selected as $x_1(0) = x_2(0) = 0.5$ with $N = 100$. After updating five times, the optimal value function V^* and the optimal control u^* are obtained. During the updating progress, Figure 9.1 shows the cost functional value at each updating step and Figure 9.2 shows the norm of weights of NN at each

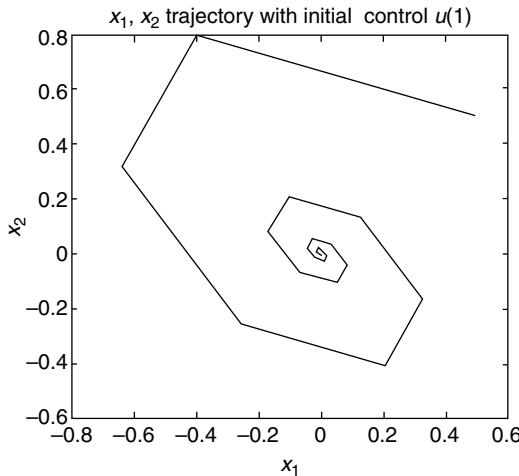


FIGURE 9.3 State trajectory with initial control u_1 .

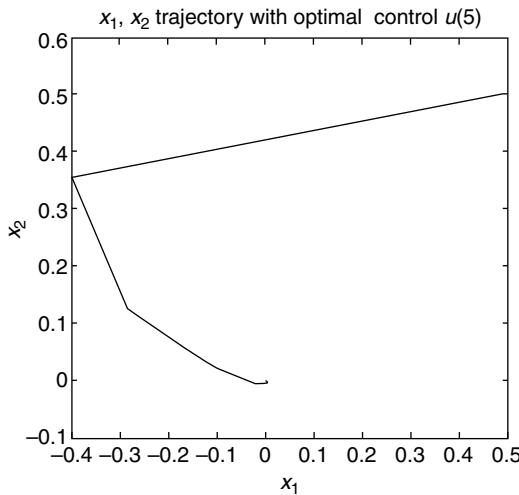


FIGURE 9.4 GHJB-based control.

updating step. From these plots, it is clear that the cost functional value continues to decrease until it reaches a minimum and afterward, it remains constant.

We implement admissible control initially without any update and optimal control on the system. Figure 9.3 shows the (x_1, x_2) trajectory with an initial admissible control whereas Figure 9.4 illustrates the (x_1, x_2) trajectory with

TABLE 9.2
Cost Value with Admissible Controls

Initial control u_1	Optimal cost	
	value	$\ W^5\ $
x_2	1.826	4.5449
$x_1 + x_2$	1.826	4.5449
$x_1 + 2x_2$	1.826	4.5449

the GHJB-based optimal control. From these figures, we can conclude that the updated control is not only an admissible control but also converges to the optimal control. Table 9.2 presents that with different initial admissible controls, the final NN weights, the optimal cost functional values, and the updated control function will converge to the unique optimal control. Hence, this method is independent on the selection of the initial admissible control for the linear discrete-time systems. In order to evaluate whether the proposed method converges to the optimal control obtained from classical optimal control methods, we use the Riccati equation in discrete-time to solve the linear quadratic regulator (LQR) optimal control problem for this system (Lewis 1992; Lewis and Syrmos 1995). Riccati equations in discrete-time are given by Finlayson (1972).

$$P(i) = A^T P(i+1)(I + BR^{-1}B^T P(i+1))^{-1}A + Q \quad (9.71)$$

$$P(N) = Q(N) = I \times 10^5 \quad (9.72)$$

$$u(i) = -(R + B^T P(i+1)B)^{-1}B^T P(i+1)Ax \quad (9.73)$$

Figure 9.5 displays that the optimal (x_1, x_2) trajectory generated by solving Riccati equation whereas Figure 9.6 depicts the error between the control inputs obtained from the proposed and the Riccati methods. Table 9.3 shows the optimal cost functional value obtained from the two methods. Comparing Figure 9.5 with Figure 9.3, and Figure 9.6 and Table 9.3, we observe that the trajectories and the optimal control inputs are the same. We can conclude that for linear discrete-time system, the updated control associated with GHJB equation will converge to the optimal control.

Example 9.3.2 (Nonlinear Discrete-Time System): Consider the nonlinear discrete-time system given by

$$x(k+1) = f(x(k)) + g(x(k))u(k) \quad (9.74)$$

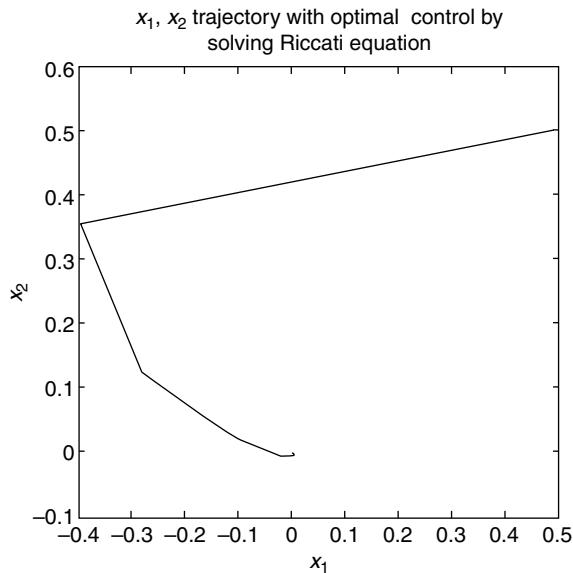


FIGURE 9.5 State trajectory using Riccati equation.

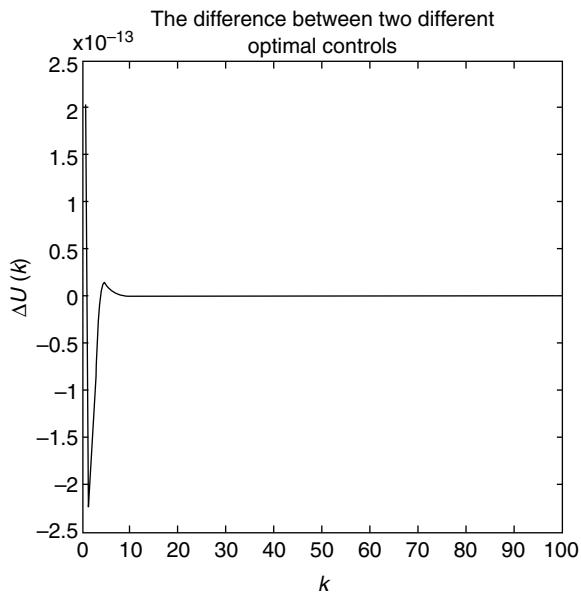


FIGURE 9.6 Control input difference.

TABLE 9.3
Comparison of Control Methods

Control method	Optimal cost value
Successively updating	1.826
Solving Riccati equation	1.826

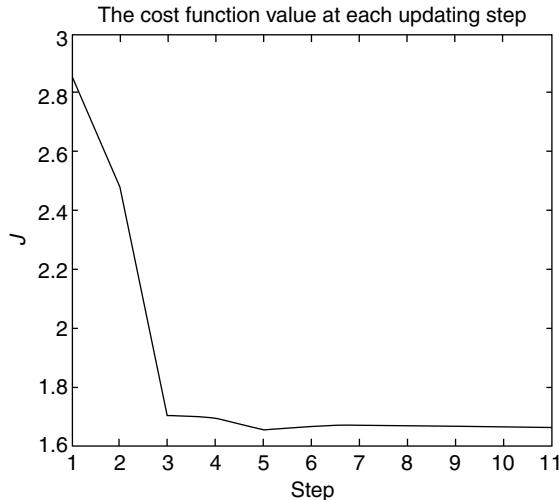


FIGURE 9.7 Cost functional value of J^i at each step.

where

$$f(x(k)) = \begin{bmatrix} -0.8x_2(k) \\ \sin(0.8x_1(k) - x_2(k)) + 1.8x_2(k) \end{bmatrix}, \quad g(x(k)) = \begin{bmatrix} 0 \\ -x_2(k) \end{bmatrix} \quad (9.75)$$

Define the cost function as

$$J(u; x(0)) = \sum_{k=0}^{\infty} (x_1(k)^2 + x_2(k)^2 + u(k)^2) \quad (9.76)$$

Select the initial control law as $u_1 = x_1 + 1.5x_2$ and the NN is also selected using (9.68). The simulation parameters and cost function are defined similar to Example 9.3.1. Figure 9.7 displays the cost functional value at each

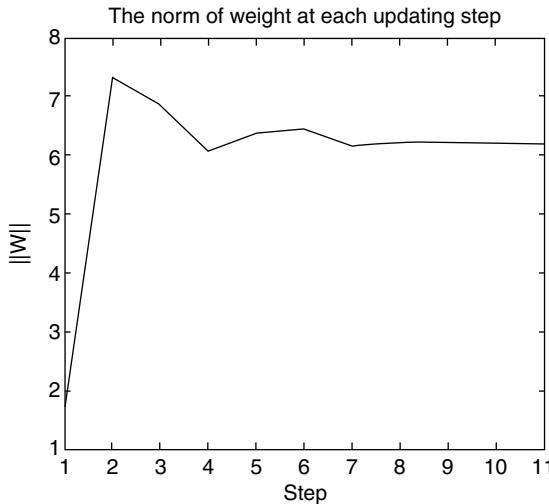


FIGURE 9.8 NN weight norm.

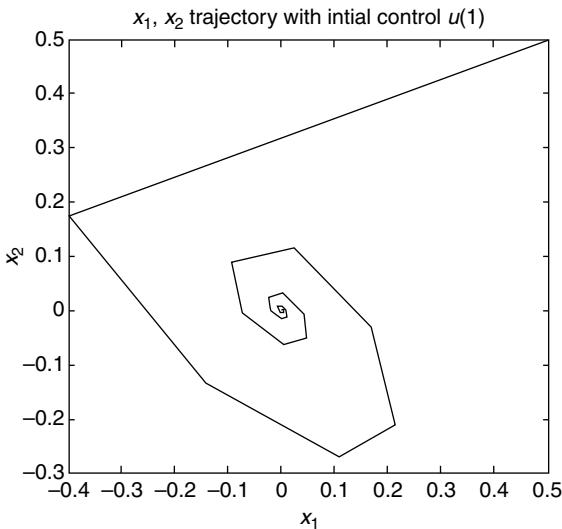


FIGURE 9.9 State trajectory with initial admissible control.

time step and Figure 9.8 depicts the norm of NN weights. After updating eight times, we get the suboptimal control u^* off-line and then the suboptimal control is implemented with several initial conditions. Figure 9.9 shows the state trajectory (x_1, x_2) with initial admissible control and without any

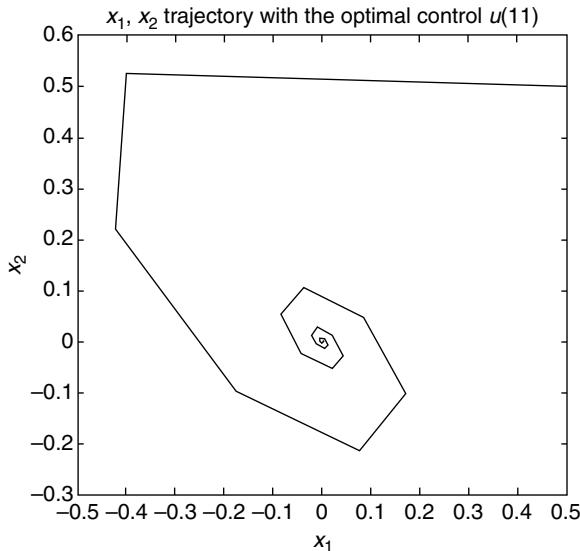


FIGURE 9.10 State trajectory with optimal control.

TABLE 9.4
GHJB-Based Nearly Optimal Control
with Initial Admissible Control Values

Initial control u_1	Optimal cost value	$\ W^{11}\ $
$x_1 + 1.5x_2$	1.6692	6.1985
x_1	1.6692	6.1984
x_2	1.6691	6.1993
$0.3x_1 + x_2$	1.6691	6.1993

update. By contrast, Figure 9.10 illustrates the state trajectory (x_1, x_2) by solving the GHJB-based control with successive approximation. Different values of initial admissible controls are used to obtain the nearly optimal control result. Table 9.4 shows that, with different initial admissible controls, the final NN norm on the weights and the optimal cost functional value are almost the same demonstrating the validity of the proposed GHJB-based solution although the state trajectory is different from the initial admissible control.

Example 9.3.3 (Two-Link Planar RR Robot Arm System): A two-link planar RR robot arm used extensively for simulation in the literature is shown in

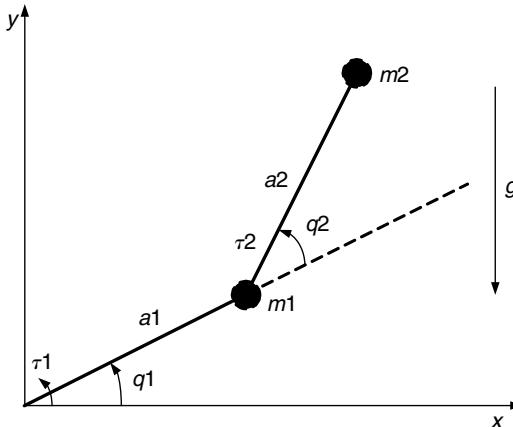


FIGURE 9.11 Two-link planar robot arm.

Figure 9.11. This arm is simple enough to simulate, yet has all the nonlinear effects common to general robot manipulators. The discrete-time dynamics of the two-link robot arm system is obtained by discretizing the continuous-time dynamics. In simulation, we apply the GHJB-based nearly optimal control method to solve the nonlinear quadratic regulator problem. In other words, we seek a suboptimal control to move the arm to the desired position while minimizing the cost functional value.

The continuous-time dynamics model of two-link planar RR robot is given (Lewis et al. 1999)

$$\begin{bmatrix} \alpha + \beta + 2\eta \cos q_2 & \beta + \eta \cos q_2 \\ \beta + \eta \cos q_2 & \beta \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -\eta(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2) \sin q_2 \\ \eta\dot{q}_1^2 \sin q_2 \end{bmatrix} + \begin{bmatrix} \alpha e_1 \cos q_1 + \eta e_1 \cos(q_1 + q_2) \\ \eta e_1 \cos(q_1 + q_2) \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \quad (9.77)$$

where $\alpha = (m_1 + m_2)a_1^2$, $\beta = m_2a_2^2$, $\eta = m_2a_1a_2$, $e_1 = g/a_1$. We define the state and control variables as: $x_1 = q_1$, $x_2 = q_2$, $x_3 = \dot{q}_1$, $x_4 = \dot{q}_2$, and $u = [\tau_1 \ \tau_2]^T$. For simulation purposes, the parameters are selected as: $m_1 = m_2 = 1 \text{ kg}$, $a_1 = a_2 = 1 \text{ m}$, $g = 10 \text{ m/sec}^2$, then $\alpha = 2$, $\beta = 1$, $\eta = 1$, $e_1 = 10$. Rewriting the continuous-time dynamics as state equation, we get

$$\dot{x} = f(x) + g(x)u \quad (9.78)$$

where

$$f(x) = \begin{bmatrix} x_3 \\ x_4 \\ \frac{-(2x_3x_4 + x_4^2 - x_3^2 - x_3^2 \cos x_2) \sin x_2 + 20 \cos x_1 - 10 \cos(x_1 + x_2) \cos x_2}{\cos^2 x_2 - 2} \\ \frac{(2x_3x_4 + x_4^2 + 2x_3x_4 \cos x_2 + x_4^2 \cos x_2 + 3x_3^2 + 2x_3^2 \cos x_2) \sin x_2 + 20[\cos(x_1 + x_2) - \cos x_1](1 + \cos x_2) - 10 \cos x_2 \cos(x_1 + x_2)}{\cos^2 x_2 - 2} \end{bmatrix} \quad (9.79)$$

and

$$g(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{2 - \cos^2 x_2} & \frac{-1 - \cos x_2}{2 - \cos^2 x_2} \\ \frac{-1 - \cos x_2}{2 - \cos^2 x_2} & \frac{3 + 2 \cos x_2}{2 - \cos^2 x_2} \end{bmatrix} \quad (9.80)$$

The control objective is moving the arm from an initial state $x(0) = [pi/3 \ pi/6 \ 0 \ 0]$ to the final state $x_d = [pi/2 \ 0 \ 0 \ 0]$ with the cost function defined as

$$J = \sum_0^{\infty} ((x(t) - x_d)^T (x(t) - x_d) + u(t)^T u(t)) dt \quad (9.81)$$

First, we will convert the continuous-time dynamics system into discrete-time. Let us consider a discrete-time system with a sampling period Δt and denote a time function $f(t)$ at $t = k\Delta t$ as $f(k)$, where k is a sampling number. If the sampling period Δt is sufficiently small compared to the time constant of the system, the response evaluated by discrete-time methods will be reasonably accurate (Lewis 1992). Therefore, we use the following approximation for the derivative of $f(k)$ as

$$\dot{f}(k) \cong \frac{1}{\Delta t} (f(k+1) - f(k)) \quad (9.82)$$

Using this relation and with the sampling interval of $\Delta t = 1$ msec, the continuous-time dynamics system can be converted to an equivalent discrete-time nonlinear system as

$$x(k+1) = f'(x(k)) + g'(x(k))u \quad (9.83)$$

where

$$f'(x(k)) = \begin{bmatrix} 0.001x_3(k) + x_1(k) \\ 0.001x_4(k) + x_2(k) \\ \frac{-(2x_3x_4 + x_4^2 - x_3^2 - x_3^2 \cos x_2) \sin x_2 + 20 \cos x_1 - 10 \cos(x_1 + x_2) \cos x_2}{1000(\cos^2 x_2 - 2)} + x_3(k) \\ \frac{(2x_3x_4 + x_4^2 + 2x_3x_4 \cos x_2 + x_4^2 \cos x_2 + 3x_3^2 + 2x_3^2 \cos x_2) \sin x_2 + 20[\cos(x_1 + x_2) - \cos x_1](1 + \cos x_2) - 10 \cos x_2 \cos(x_1 + x_2)}{1000(\cos^2 x_2 - 2)} + x_4(k) \end{bmatrix} \quad (9.84)$$

and

$$g'(x(k)) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{1000(2 - \cos^2 x_2(k))} & \frac{-1 - \cos x_2(k)}{1000(2 - \cos^2 x_2(k))} \\ \frac{-1 - \cos x_2(k)}{1000(2 - \cos^2 x_2(k))} & \frac{3 + 2 \cos x_2(k)}{1000(2 - \cos^2 x_2(k))} \end{bmatrix} \quad (9.85)$$

with cost functional value in discrete-time chosen as

$$J = \sum_{k=0}^N ((x(k) - x_d)^T Q (x(k) - x_d) + u(x(k))^T R u(x(k))) \quad (9.86)$$

where $Q = 0.001 \times I^4$ and $R = 0.001 \times I^4$. The solution to the problem is almost the same as the linear system example except that we move the original point of axis to $x_d = [pi/2 \ 0 \ 0 \ 0]$ and use the new axis as $x'_1(k) = x_1(k) - pi/2$. The NN to approximate the GHJB equation is selected as a polynomial function up to a fourth order approximation, which means that $n = 4$ and $M = 4$. From

(9.51), the NN can be constructed as

$$\begin{aligned}
 V(x'_1, x_2, x_3, x_4) = & w_1 x'_1^2 + w_2 x_2^2 + w_3 x_3^2 + w_4 x_4^2 + w_5 x'_1 x_2 + w_6 x'_1 x_3 \\
 & + w_7 x'_1 x_4 + w_8 x_2 x_3 + w_9 x_2 x_4 + w_{10} x_3 x_4 + w_{11} x'_1^4 \\
 & + w_{12} x_2^4 + w_{13} x_3^4 + w_{14} x_4^4 + w_{15} x'_1^3 x_2 + w_{16} x'_1^3 x_3 \\
 & + w_{17} x'_1^3 x_4 + w_{18} x_2^3 x'_1 + w_{19} x_2^3 x_3 + w_{20} x_2^3 x_4 + w_{21} x_3^3 x'_1 \\
 & + w_{22} x_3^3 x_2 + w_{23} x_3^3 x_4 + w_{24} x_4^3 x'_1 + w_{25} x_4^3 x_2 + w_{26} x_4^3 x_3 \\
 & + w_{27} x'_1^2 x_2 x_3 + w_{28} x'_1^2 x_2 x_4 + w_{29} x'_1^2 x_3 x_4 + w_{30} x_2^2 x'_1 x_3 \\
 & + w_{31} x_2^2 x'_1 x_4 + w_{32} x_2^2 x_3 x_4 + w_{33} x_3^2 x'_1 x_2 + w_{34} x_3^2 x'_1 x_4 \\
 & + w_{35} x_3^2 x_2 x_4 + w_{36} x_4^2 x'_1 x_2 + w_{37} x_4^2 x'_1 x_3 + w_{38} x_4^2 x_2 x_3 \\
 & + w_{39} x'_1 x_2 x_3 x_4
 \end{aligned} \tag{9.87}$$

Associated gradient vector is derived as

$$\nabla V = \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \frac{\partial V}{\partial x_3}, \frac{\partial V}{\partial x_4} \right]^T$$

$$\nabla^2 V = \begin{bmatrix} \frac{\partial^2 V}{\partial x_1^2} & \frac{\partial^2 V}{\partial x_1 \partial x_2} & \frac{\partial^2 V}{\partial x_1 \partial x_3} & \frac{\partial^2 V}{\partial x_1 \partial x_4} \\ \frac{\partial^2 V}{\partial x_2 \partial x_1} & \frac{\partial^2 V}{\partial x_2^2} & \frac{\partial^2 V}{\partial x_2 \partial x_3} & \frac{\partial^2 V}{\partial x_2 \partial x_4} \\ \frac{\partial^2 V}{\partial x_3 \partial x_1} & \frac{\partial^2 V}{\partial x_3 \partial x_2} & \frac{\partial^2 V}{\partial x_3^2} & \frac{\partial^2 V}{\partial x_3 \partial x_4} \\ \frac{\partial^2 V}{\partial x_4 \partial x_1} & \frac{\partial^2 V}{\partial x_4 \partial x_2} & \frac{\partial^2 V}{\partial x_4 \partial x_3} & \frac{\partial^2 V}{\partial x_4^2} \end{bmatrix} \tag{9.88}$$

Select the initial admissible control law as

$$u_1 = [-500x'_1 - 500x_3, -200x_2 - 200x_4]^T \tag{9.89}$$

Control function updating rule is taken as

$$u_{i+1} = -[g'^T \nabla^2 V^{(i)}(k) g' + 2R]^{-1} g'^T (\nabla V^{(i)}(k) + \nabla^2 V^{(i)}(k)(f'(k) - x(k))) \tag{9.90}$$

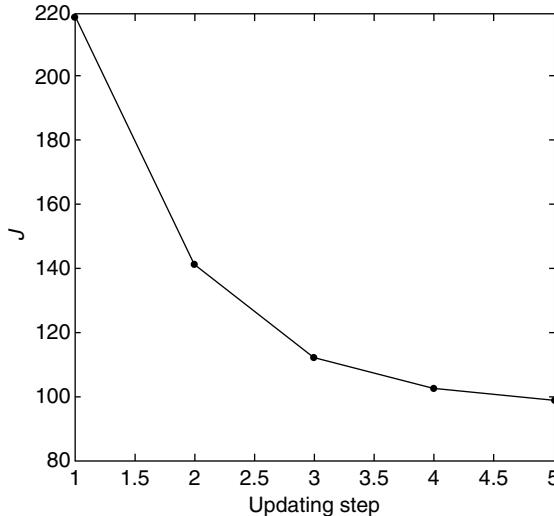


FIGURE 9.12 Cost functional, J^i , at each updating step.

The u_i and V^i satisfy the GHJB equation

$$\begin{aligned} & \frac{1}{2}(f'(x) + g'(x)u^{(i)}(x) - x^T)\nabla^2 V^{(i)}(x)(f'(x) + g'(x)u^{(i)}(x) - x) \\ & + \nabla V^{(i)}(x)^T(f'(x) + g'(x)u^{(i)}(x) - x) + x^T Qx + u^{(i)T} R u^{(i)} = 0 \quad (9.91) \end{aligned}$$

In the simulation, the mesh size δx is selected as 0.2, the AS region is chosen as $0 \leq x_1 \leq 2$, $-1 \leq x_2 \leq 1$, $-1 \leq x_3 \leq 1$, and $-1 \leq x_4 \leq 1$. The small positive constant is selected as $\varepsilon = 0.01$ with $N = 2000$. We use the GHJB method to obtain the nearly optimal control. After updated five times, the control has converged to the nearly optimal control u^* . Figure 9.12 shows the cost functional value over time whereas, Figure 9.13 shows the norm of NN weights at each updating time.

After we get the optimal control, we implement the initial admissible and optimal controls on the two-link planar robot arm system. Figure 9.14a displays the state trajectory (x_1, x_2) with initial admissible control and GHJB-based sub-optimal control. Similarly, Figure 9.14b illustrate the state trajectory (x_3, x_4) with initial admissible and GHJB-based optimal control. From these trajectory figures, we know that the robot arm has moved from the initial location to the final goal. On the other hand, Figure 9.15a, b depict the initial control and suboptimal control inputs achieved by successive approximation. Table 9.5 shows that with different initial admissible controls, the converged norm of NN

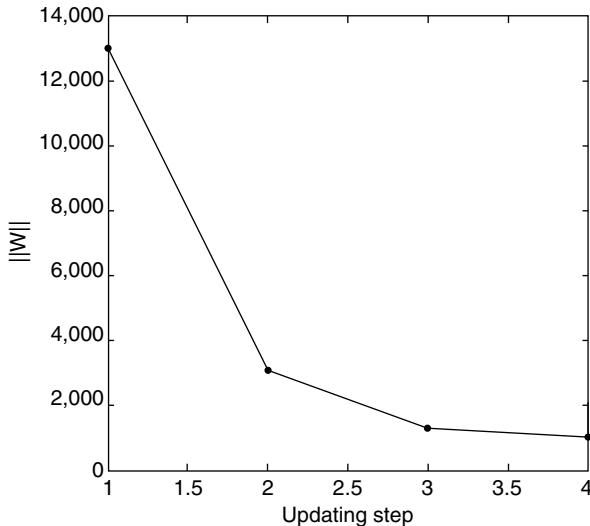


FIGURE 9.13 Norm of the NN weights.

weights and the optimal cost functional values are almost close. It is important to note that with different admissible control function values, the successive approximation-based updated controls will converge to a unique improved control and the improved cost function values are almost the same. Since a small function approximation error value is used in solving the GHJB equation, the approximation-based GHJB solution renders the suboptimal control, which is quite close to the optimal control solution.

From Figure 9.14b and Figure 9.15b, the trajectory with nearly optimal control is a little longer than the trajectory with initial admissible control even though the cost functional value with optimal control is significantly less for GHJB-based control. This is due to the trade-off observed between the trajectory selection and the control input. The selection of the weighting Q and R matrices will dictate the selection. If we are more interested in perfect trajectory, we can select a higher $\|Q\|$ or reduce $\|R\|$. If we are more interested in saving control energy, we can select a lower $\|Q\|$ or increase $\|R\|$. For example, if we select $Q = 1000 \times I^4$ and $R = 0.001 \times I^4$, Figure 9.16a, b displays that the results obtained are different from those of Figure 9.14a, b. It is important to note that the trajectory in Figure 9.16a is close to a straight line but at the expense of the control input.

In Table 9.5, optimal cost values J^* with different initial control are not exactly same as that of the previous two examples, but are still reasonable due to the selection of the mesh size of 0.2. By decreasing the mesh size, one can

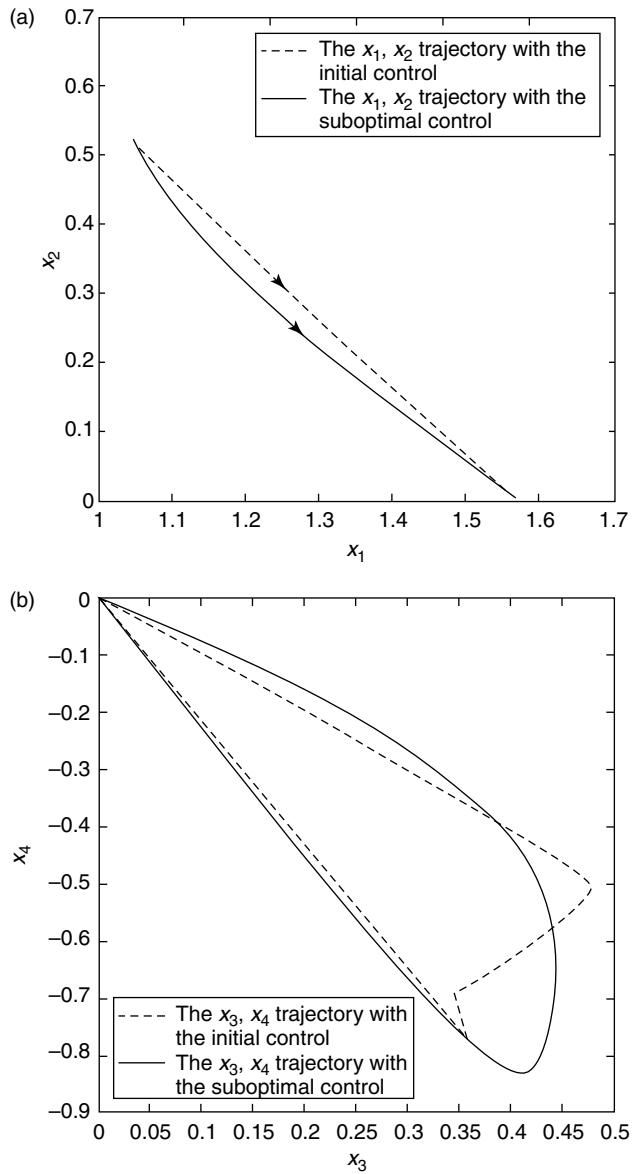


FIGURE 9.14 State trajectory: (a) (x_1, x_2) and (b) (x_3, x_4) .

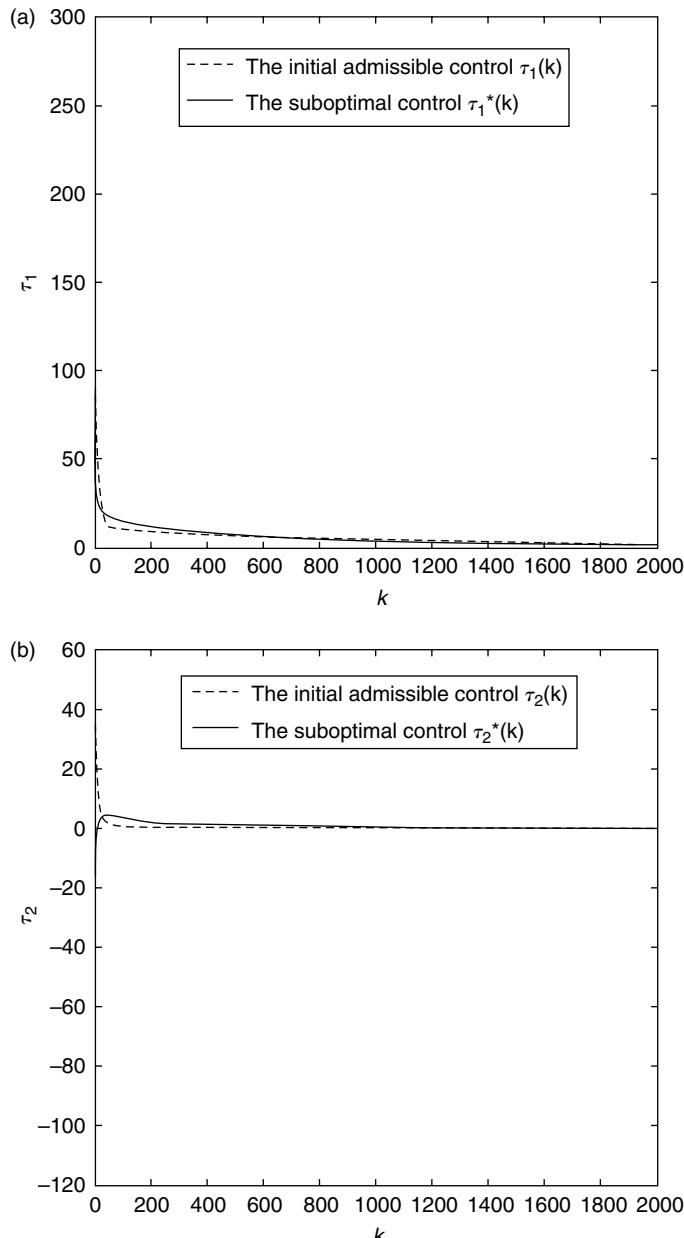


FIGURE 9.15 (a) Initial control τ_1 and suboptimal control τ_1^* . (b) The initial control τ_2 and suboptimal control τ_2^* .

TABLE 9.5
GHJB-Based Solution with Admissible Control Values

Initial control u_1	Optimal cost value J^*	$\ W^*\ $
$[-500x'_1 - 500x_3, -200x_2 - 200x_4]^T$	98.745	912.1
$[-200x'_1 - 300x_3, -200x_2 - 200x_4]^T$	97.726	928.32
$[-200x'_1 - 400x_3 - 200x_4, -200x_2 - 200x_4]^T$	97.252	999.09
$[-300x'_1 - 400x_3, -300x_2 - 300x_4]^T$	97.779	954.77
$[-300x'_1 - 200x_3 - 200x_4, -200x'_1 - 200x_2 - 200x_3 - 200x_4]^T$	98.294	968.04

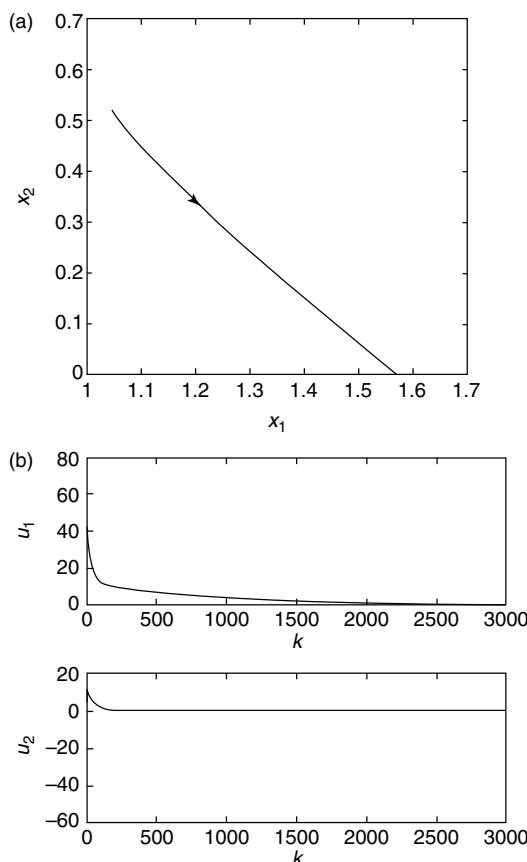


FIGURE 9.16 (a) State trajectory (x_1, x_2) with suboptimal control. (b) Suboptimal control input.

increase the accuracy of convergence in the cost functional. In the previous second order system examples, the mesh size is selected as 0.01, which is quite small. But in the fourth-order robot system, a mesh size of 0.2 is chosen as a trade-off between accuracy and computation. Decreasing the mesh size requires more memory to store the values due to an increase in computation even though the cost functional will converge to a unique minimum.

9.4 CONCLUSIONS

In this chapter, HJB, GHJB, and pre-Hamiltonian functions in discrete-time are introduced. A systematic method of obtaining the optimal control for general affine nonlinear discrete-time system is proposed. Given an admissible control, the updated control through NN successive approximation of the GHJB equation renders an admissible control. For the LQR problem, the updated control will converge to the optimal control. For nonlinear discrete-time system, the updating control law will converge to an improved control, which renders a suboptimal control.

REFERENCES

- Abu-Khalaf, M. and Lewis, F.L. Nearly optimal HJB solution for constrained input system using a neural network least-squares approach, *Proc. IEEE Conf. Decis. Contr.*, 1, 943–948, 2002.
- Beard, R.W., *Improving the closed-loop performance of nonlinear systems*, Ph.D. Thesis, Rensselaer Polytechnic Institute, 1995.
- Beard, R.W. and Saridis, G.N., Approximate solutions to the time-invariant Hamilton–Jacobi–Bellman equation, *J. Optimiz. Theory Appl.*, 96, 589–626, 1998.
- Beard, R.W., Saridis, G.N., and Wen, J.T., Galerkin approximations of the generalized Hamilton–Jacobi–Bellman equation, *Automatica*, 33, 2159–2177, 1997.
- Bernstein, D.S., Optimal nonlinear, but continuous, feedback control of systems with saturating actuators, *Int. J. Contr.*, 62, 1209–1216, 1995.
- Bertsekas, D.P. and Tsitsiklis, J.N., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
- Burk, F., *Lebesgue Measure and Integration*, John Wiley & Sons, New York, NY, 1998.
- Chen, Z. and Jagannathan, S., Generalized Hamilton–Jacobi–Bellmann formulation based neural network control of affine nonlinear discrete-time systems, *Proc. IEEE Conf. Decis. Contr.*, 3, 4123–4128, 2005.
- Finlayson, B.A., *The Method of Weighted Residuals and Variational Principles*, Academic Press, New York, NY, 1972.
- Han, D. and Balakrishnan, S.N., State-constrained agile missile control with adaptive critic based neural networks, *IEEE Trans. Contr. Syst. Technol.*, 10, 481–489, 2002.

- Kleinman, D., On a iterative technique for Riccati equation computations, *IEEE Trans. Autom. Contr.*, 13, 114–115, 1968.
- Lewis, F.L., *Applied Optimal Control and Estimation*, Texas Instruments, Upper Saddle River, NJ, 1992.
- Lewis, F.L. and Abu-Khalaf, M., A Hamilton–Jacobi setup for constrained neural network control, *Proceedings of the IEEE International Symposium on Intelligent Control*, Houston, pp. 1–15, 2003.
- Lewis, F.L. and Syrmos, V.L., *Optimal Control*, John Wiley & Sons, New York, NY, 1995.
- Lewis, F.L., Jagannathan, S., and Yesilderek, A., *Neural Network Control of Robot Manipulator and Nonlinear Systems*, Taylor & Francis Inc., UK, 1999.
- Lin, W. and Brynes, C.I., H_∞ -control of discrete-time nonlinear systems, *IEEE Trans. Autom. Contr.*, 41, 494–510, 1996.
- Lyshevski, S.E., *Control Systems Theory with Engineering Applications*, Birkhauser, Boston, MA, 1990.
- Miller, W.T., Sutton, R., and Werbos, P., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990.
- Munos, R., Baird, L.C., and Moor, A.W., Gradient descent approaches to neural-net-based solutions of the Hamilton–Jacobi–Bellman equation, *Proc. Int. Joint Conf. Neural Netw.*, 3, 2152–2157, 1999.
- Parisini, T. and Zoppoli, R., Neural approximations for infinite-horizon optimal control of nonlinear stochastic systems, *IEEE Trans. Neural Netw.*, 9, 1388–1408, 1998.
- Park, C. and Tsiotras, P., Approximations to optimal feedback control using a successive wavelet collocation algorithm, *Proc. Am. Contr. Conf.*, 3, 1950–1955, 2003.
- Saridis, G.N. and Lee, C.S., An approximation theory of optimal control for trainable manipulators, *IEEE Trans. Syst. Man, Cybern.*, 9, 152–159, 1979.
- Xin, Ming and Balakrishnan, S.N., A new method for suboptimal control of a class of nonlinear systems, *Proc. IEEE Conf. Decis. Contr.*, 3, 2756–2761, 2002.

PROBLEMS

SECTION 9.6

9.6-1: Consider the linear discrete-time system given in state space formulation

$$x(k+1) = Ax(k) + Bu(k)$$

where

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

Define the cost functional

$$J(u; x(0)) = \sum_{k=0}^{N-1} (x_1(k)^2 + x_2(k)^2 + u(k)^2)$$

Implement GHJ-based NN controller and compare it with Riccati solution.

9.6-2: Consider the nonlinear discrete-time system given by

$$x(k+1) = f(x(k)) + g(x(k))u(k)$$

where

$$\begin{aligned} f(x(k)) &= \begin{bmatrix} 1.004x_1(k) - 0.004x_2(k) \\ -0.001x_1(k) + x_2(k) + 0.002x_1(k)x_2(k) + 0.002x_2^2(k) \end{bmatrix} \\ g(x(k)) &= \begin{bmatrix} 0 \\ -0.001x_2(k) \end{bmatrix} \end{aligned}$$

Define the cost function:

$$J(u; x(0)) = \sum_{k=0}^{\infty} (x_1(k)^2 + x_2(k)^2 + u(k)^2)$$

Select an initial stabilizing control law. Use GHJB to update the NN control by selecting an appropriate mesh size.

9.6-3: A two-link planar RR robot arm used extensively for simulation in the literature is shown in Figure 9.11. Derive a GHJB-based controller to track a desired sinusoidal trajectory.

10 Neural Network Output Feedback Controller Design and Embedded Hardware Implementation

This chapter shows how to implement neural network (NN) controllers on actual industrial systems. An embedded real-time control system developed at Embedded Systems and Networking Systems Laboratory (ESNL) of the University of Missouri-Rolla is described. The hardware and software interfaces are outlined. Next a real world problem is described — namely, control of a spark ignition (SI) engine-operating lean and with high exhaust gas recirculation (EGR) levels.

This complex engine system presents severe difficulties in terms of cyclic dispersion in heat release due to operation at lean conditions or with EGR levels causing misfires and ultimately leading to high emissions. Using an experimentally verified mathematical model, a novel NN output feedback controller is derived both for lean engine mode and engine with high EGR levels. Minimizing cyclic dispersion for the SI engines is introduced in Example 6.2.2 using total fuel and air in the given cylinder as state feedback variables. Recall that an SI engine is modeled as a nonlinear discrete-time system in nonstrict feedback form and designing suitable controllers becomes a problem due to the causal nature of certain signals. Moreover, the total air and fuel present in a given cylinder is not known beforehand and therefore a state feedback controller cannot be implemented in practice. By contrast, the proposed output feedback controller consists of an observer to estimate the total fuel and air in the cylinder prior to combustion by using a two-layer NN. Similarly, two two-layer NNs are employed for the output feedback controller since a backstepping approach will be utilized to design the controller. This output feedback controller will be implemented on engines to reduce emissions.

Current automotive three-way catalysts require an engine to operate at stoichiometric conditions with the addition of EGR to reduce engine-out NO_x .

Noncatalytic SI engine designs (e.g., generator sets and other industrial applications) could make use of a combination of both lean operation and high levels of EGR to reduce engine-out NO_x as well as improve fuel efficiency. Hence, an NN controller for operating an engine with high levels of EGR as well as for lean operation can be deployed on several classifications of engines including engines with three-way catalysts, engines without three-way catalysts, as well as potentially on diesel engines or nontraditional, lean operating engines such as direct injection stratified charge and homogeneous charge compression ignition engines since the NN can learn the dynamics online.

10.1 EMBEDDED HARDWARE-PC REAL-TIME DIGITAL CONTROL SYSTEM

Implementation of advanced control schemes has long been an expensive and time-consuming task. Many commercial products are available that implement standard proportional, integral, and derivative (PID) servocontrol loops, but anything other than PID control often requires the use of custom embedded hardware boards and development systems. The development costs can be prohibitive for projects with modest budgets. Many systems that traditionally use PID control schemes can potentially benefit from advanced control schemes. Among these are the spark engine controls during lean and with high EGR levels, hybrid engines, and fuel flexible engines. Besides the engine systems, there are many other nonlinear systems such as autonomous vehicles, automotive systems, power grids, and so on.

The real-time control system (RTCS) was developed at UMR's ESNL to facilitate the implementation of advanced control schemes. RTCS was developed under two National Science Foundation (NSF) grants. Based on two inexpensive processing systems, one an embedded PC104 card and the other a PC and nicknamed embedded hardware, its premise is simple: use the PC to implement user interfaces to the controller and the embedded hardware for real-time control. The cost of these two hardware modules is relatively cheap, a major consideration in the design of RTCS. The PC hardware is used mainly for user interface and can be removed under normal production leaving the embedded controller board alone.

10.1.1 HARDWARE DESCRIPTION

Hardware requirements and selection are discussed in succeeding parts and the details are taken from Vance (2005) and Vance et al. (2005). There are two main parts to the hardware selection: the embedded PC that performs the rigorous

computations and the engine–PC interface controller, which handles interrupts from the shaft encoders, records pressure measurements, and handles the fuel injection timing. First selection of the PC is discussed and then selection of components for the controller interface.

Operation of the PC software is reviewed, describing how pressure data gives heat release information from the cycle. Heat release use and engine control input calculation are discussed. Requirements of the embedded PC are satisfaction of the limited calculation time window of 17.667 msec at 1000 rpm and communication capabilities to interface with the engine. Ultimately, a PC770 model single-board computer from Octagon Systems is selected with networking capabilities, PC104 expansion, and dedicated digital I/O. The embeddable PC has 128-MB RAM, 256-MB flash memory, and an 800-MHz Intel P3.

This PC770 single-board computer, depicted in Figure 10.1, allows for communication with other computers and real-time data analysis with multiple hardware expansion possibilities. Implementation of an EGR algorithm in addition to the lean combustion algorithm will be facilitated by the PC770's virtuosity.

An analog-to-digital (A/D) conversion takes place every $83.3 \mu\text{sec}$, which translates to a sampling rate of 12 kHz. In addition, a fast conversion is



FIGURE 10.1 PC770 embedded computer. (Used with permission from Octagon Systems.)

preferred so that pressure data can be sent to the PC as soon as possible. A Texas Instruments part capable of 100 kilo-samples per second with a typical conversion time of 1 μ sec is used.

An 8051 variant — Atmel AT89C51RB2 microcontroller — populates the engine–PC controller interface board. It synchronizes crank-angle interrupts and pressure measurements and keeps track of the current location within the engine cycle. The microcontroller acts as a buffer for measured pressure that needs to be forwarded to the PC via additional RAM located on the chip. Timer 0 located on the device is used to measure half crank angles for the given engine speed. Fuel pulse widths are accurately controlled via timer 1.

The microcontroller operates at 40 MHz with a peripheral clock of 1/12 the core frequency. The period of timer increments is 300 nsec. The timers are 16-bit so the maximum time span of the Atmel microcontroller for one such timer is 19.6605 msec, which is larger than the allotted fuel injection window of 18.1667 msec.

10.1.2 SOFTWARE DESCRIPTION

The embedded PC implements artificial NNs in software running on a Linux kernel operating system. Linux was chosen because of its small overhead compared to Microsoft Windows and the use of a complete operating system does not hinder calculation time on a fast machine and simplifies usage and development. The NN software was written in C and compiled by the GNU C compiler. Programming in C offers features to quickly store controller data to file for later analysis and display real-time events on-screen.

10.2 SI ENGINE TEST BED

Experiments were performed on a 0.497-l single-cylinder Ricardo Hydra research engine with identical cylinder geometry to the Ford Zetec engine. A production fuel injection system was used along with a modified Ford injector driver for control. The test engine runs on one of four cylinders; the other three blocked off to eliminate the nonlinearities that would be introduced by the other bifurcating combustion cycles. A pressure transducer in this cylinder provides a signal for a charge amplifier, which in turn has an analog output that is read by an A/D converter. The engine was maintained at 1000 rpm for all experiments using a motoring dynamometer, even when engine behavior was very erratic under high levels of simulated EGR.

The test engine, which is illustrated in Figure 10.2, has configured shaft encoders (see Figure 10.3) for crank-angle degree and start of cycle that give active-low TTL pulses at crank angle and cycle start, respectively. These signals

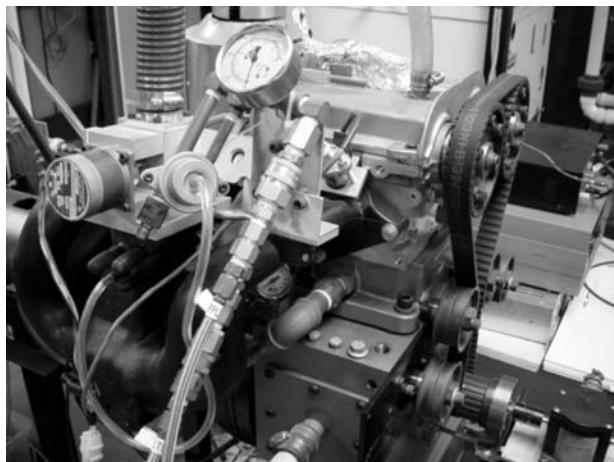


FIGURE 10.2 CFR engine.

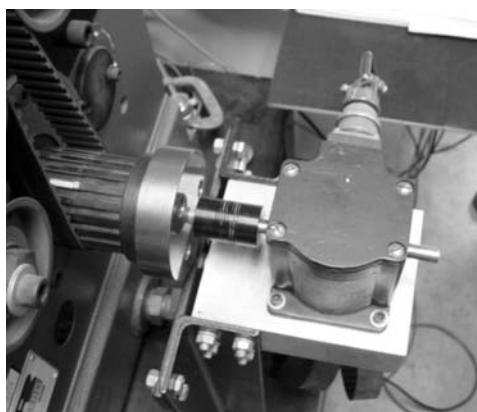


FIGURE 10.3 Shaft encoder.

are interfaced with the CMOS inputs of a microcontroller so they can be used to correlate control events to position within the cycle.

A fuel injector driver circuit is arranged such that it will accept an active-low TTL input. When the input signal is driven low, the fuel injector opens allowing fuel to pass into the cylinder due to fuel pressure. A timer is configured to count how much time passes for fuel injection. The fuel injector signal is pulled high thereby shutting off fuel flow when the timer expires. This timer value is changed for every cycle according to the controller's calculations.

10.2.1 ENGINE-PC INTERFACE HARDWARE OPERATION

The interface between the embedded computer and the engine handles the precise timing of pressure measurements and fuel injection. An 8051 variant microcontroller is used to achieve these goals. External hardware interrupts of the 8051 detect crank-angle degree signals and start of cycle signals. A/D conversion, PC communication, and fuel injection timing are undertaken by the microcontroller.

Pressure measurements are taken by starting A/D conversion of the charge amplifiers analog output upon the detection of a crank angle or half crank angle. Once the A/D conversion is finished either signaling an interrupt, the digitized pressure value is recorded into external memory of the microcontroller. For the given pressure recording window of 345 to 490°, 290 pressure measurements are taken when both crank angle and half crank angle are used.

A timer is used to detect half crank-angle degrees. The timer is loaded with the value that would correspond with half of the crank-angle period — 0.0833 msec at 1000-rpm engine speed. That value is dependent upon the clock frequency and how many machine cycles the timer requires to increment. When the timer overflows, it causes an interrupt whereby a pressure measurement from the cylinder via the charge amplifier is converted by the A/D and stored.

Pressure measurements can be sent to the PC as soon as they are recorded. When not recording pressure, the microcontroller can send pressure measurements to the PC by putting them into a latch one by one and alerting the PC that a new pressure measurement is available. This increase in throughput allows for more calculation time for the PC since most of the pressure measurements have already been sent by the time the last pressure is recorded. Once the last pressure measurement is latched onto the output and the PC notifies the microcontroller that it has been recorded, a start-calculation signal is sent to the PC. The engine-PC interface hardware then waits for the PC to return a new value to load into its fuel injection timer.

When the PC has finished its calculations required to process the new inputs through the NN, it returns a newly computed fuel input to the interface hardware in the form of a fuel pulse width that can be loaded into timer registers. Fuel injection begins when the corresponding output on the microcontroller is set low because of crank angle 596° detection. Fuel injection stops when the timer overflows causing an interrupt.

Figure 10.4 shows a block diagram of the controller interface with signals drawn between components. BNC connectors connect the board to the engine sensors, transducer, and a fuel injector. A 26-pin stand-off and parallel cable connect the interface to the digital I/O port of the embedded PC.

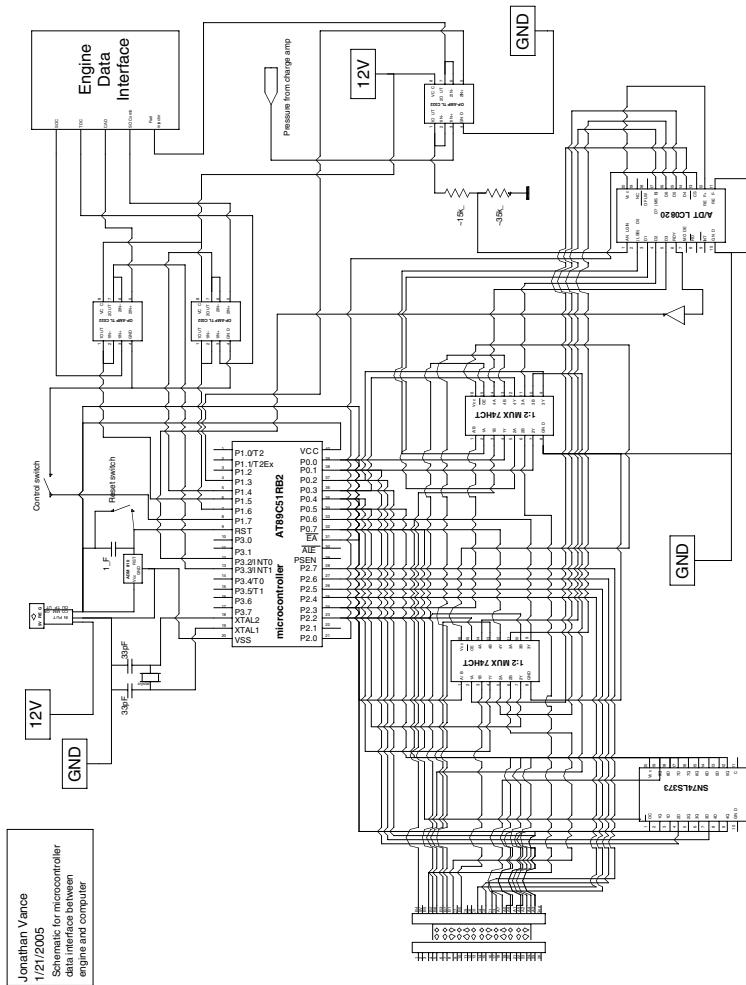


FIGURE 10.4 Engine-PC interface schematic.

10.2.2 PC OPERATION

The embedded PC implements artificial NNs in software. Communication between the PC and engine-interface controller happens through a dedicated digital I/O port. 8-bit buses are created on the digital I/O port for communicating with the 8051 hardware in parallel. A two-way handshaking protocol is used that maximizes throughput since the slowest of the microcontroller and the digital I/O chip is the limiting factor.

The PC receives all of the pressure measurements and finally a start signal from the engine-interface microcontroller circuit via digital I/O. The pressures are then integrated by a composite trapezoid rule function. This integration of pressure corresponds to the heat release for the given cycle from which those pressures were recorded. The heat release value is then scaled to be within the range of heat release values created by the Daw engine model used for simulation and suitable for input to the NN.

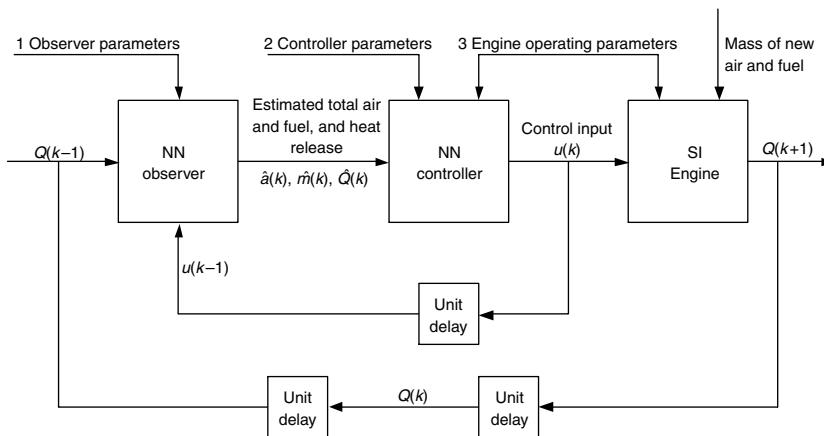
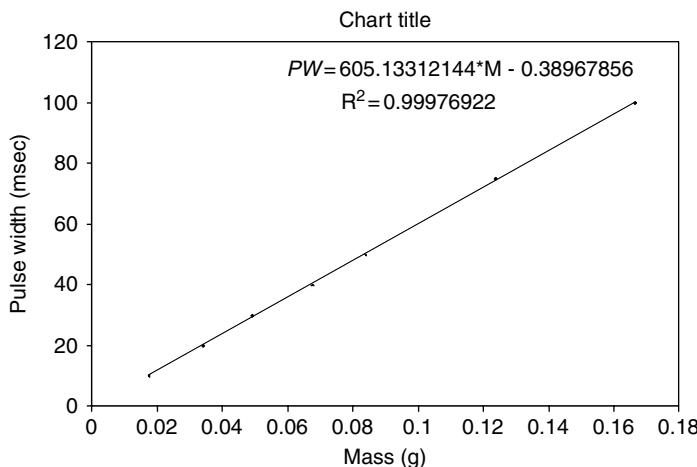
Before control is handed to the NN, the program initializes the weights and outputs by calling an initialization routine. Proper initialization is necessary to guarantee that the controller will converge on the nonlinear system. Initialization can be random from the uniformly distributed random number generator in the C standard library or static values can be used for initialization.

The calculated value for heat release is passed to the NN controller. It calculates the errors of estimated heat release to the measured heat release for the cycle. The hidden-layer weights are updated online according to a newly developed update rule based on Lyapunov analysis. The details of the observer and controller development for lean operation and with high EGR levels are described in Section 10.3 and Section 10.4, respectively. A block diagram representation of the output feedback controller is shown in Figure 10.5. Based on available data the observer NN estimates the fuel, air, and heat release of the next engine cycle. The controller NN output is updated and the NN controller software program returns fuel.

Fuel output of the controller function is actually a deviation from nominal fuel needed to operate the engine at the desired operating point. Thus, controller output of fuel in grams is added to the nominal fuel in grams to find a new total value of fuel in grams to be injected into the cylinder for the next cycle to minimize cyclic dispersion. The flow rate of the fuel injector must be known in order to calculate the time required to let an amount of fuel in grams pass into the cylinder.

Figure 10.6 shows the injector flow in terms of seconds per gram. Using the equation $PW = 605.13312144 * M - 0.38967856$, one can input fuel in grams to get a pulse width in milliseconds.

Another function uses this value in milliseconds to find the values to be loaded into the microcontroller timer. For a 16-bit timer this function

**FIGURE 10.5** Block diagram representation of the controller.**FIGURE 10.6** Cooperative fuel research (CFR) engine injector flow.

looks like

$$y = (\text{Timer}_{\text{period}} * ((2^{16}) - 1)) * x + ((2^{16}) - 1)$$

where x is the fuel pulse width in milliseconds and y is the timer load value. Naturally, the timer load value must be converted from decimal to hexadecimal to load the timer registers.

For the 16-bit timer of the 8051, a low-timer byte and high-timer byte are sent from the PC via the digital I/O lines again using the two-way handshaking protocol. The PC asserts that the new fuel pulse width has been sent and the microcontroller is ready to inject fuel for the next cycle.

10.2.3 TIMING SPECIFICATIONS FOR THE CONTROLLER

The controller operates on a fuel research engine running at 1000 rpm which is $16 \frac{2}{3}$ rotations per second. The crank shaft completes two rotations for every engine cycle, and, thus, there are $8 \frac{1}{3}$ engine cycles per second. A shaft encoder on the crank creates a low TTL pulse for every crank-angle degree. Another sensor on the cam shaft sends a low TTL pulse at the start of each cycle corresponding to the first crank-angle degree. Since the crank has two rotations for every cycle, there are 720° per cycle. At $8 \frac{1}{3}$ engine cycles per second, a crank-angle degree event happens 6000 times per second or 0.1667 msec elapse per crank angle.

In-cylinder pressure measurements must be taken during combustion to obtain the heat release per engine cycle. The test engine starts combustion at 345° after start of cycle and combustion is completed when the exhaust valve opens at 490° . A pressure transducer signal from the cylinder is read from the charge amplifier at every crank-angle degree and half crank-angle degree. These pressure measurements are sent to a PC for integration. The pressure measurement window is 145° or 24.1667 msec.

Calculations cannot begin until the PC has received all pressure measurements after 490° and calculations must be finished by 596° after start of cycle when the controller-loaded fuel injector timer begins counting for the injected fuel pulse width. The calculation window is 106° wide or 17.667 msec. In this time all engine-to-PC-to-engine communication must be complete.

The new fuel input calculated by the NN controller must be received by the fuel injector timer soon enough to inject all of the fuel onto the back of the closed intake valve to ensure proper vaporization of the new fuel. The intake valve opens at 705° . Fuel is chosen to be injected at the same crank angle for every cycle at 596° . This allows for a fuel injection window of 18.16671 msec before the intake valve opens. Fuel pressure and injector flow rate ensure that this is enough time to inject the calculated amount of fuel.

Figure 10.7 shows the timing events in terms of degrees and in seconds. Start of cycle is labeled SOC and top dead center is labeled TDC. The pressure window is shown in milliseconds on the second plot as well as the calculation window and the fuel injection window. These timing requirements are used to select the hardware necessary to perform engine control in real-time with the lean combustion algorithm artificial NN.

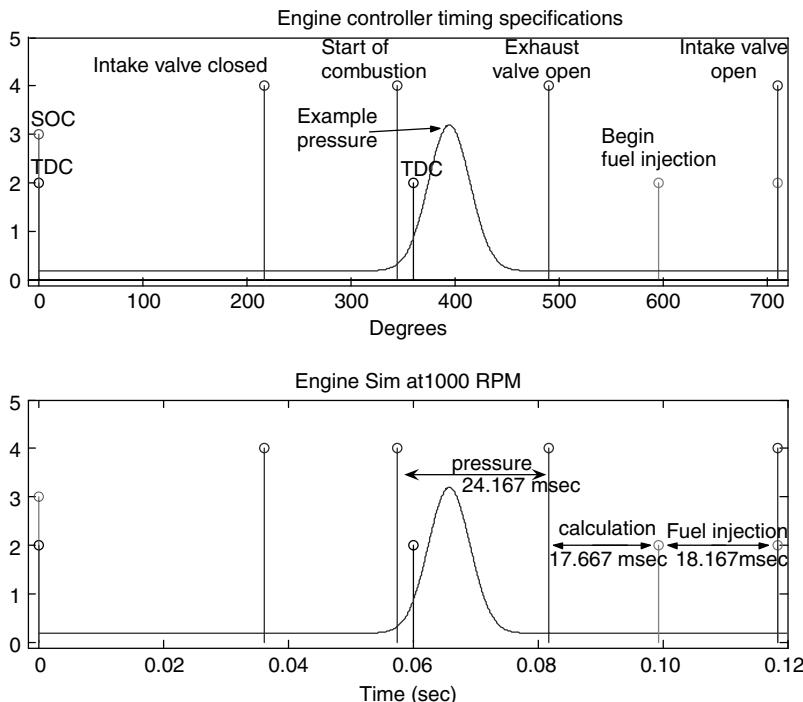


FIGURE 10.7 Engine controller timing requirements.

10.2.4 SOFTWARE IMPLEMENTATION

A software implementation of the artificial NN controller is used for simplicity and ease of development. Software allows the algorithm to be changed quickly and learning rates and adaptation parameters can be adjusted on the spot between each engine test run. An embeddable computer with an Intel Pentium III processor running at 800 MHz is used to perform software computations and data acquisition. Table 10.1 describes the hardware performance. This choice is supported by estimations made from a similar 20-node NN algorithm for EGR.

For testing purposes the total time period of calculation and communication with the engine interface is calculated by reading dummy pressure variables from a digital I/O port, performing an iteration of the NN function on the calculated heat release and outputting the new fuel to another digital I/O port. A corresponding digital I/O signal acts as a busy signal while the software is working. For the 800-MHz Pentium III these periods were recorded in the following figure for a varying number of controller nodes with observer nodes set to 100.

TABLE 10.1
Hardware Performance

Operation	486 clocks	#operations	#clocks
Addition	10	398	3,980
Subtraction	10	384	3,840
Multiplication	16	415	6,640
Division	73	387	28,251
Exponent	70	372	26,040
Total			68,751
CPU (MHz)	100		
Time (sec)	0.00068751		

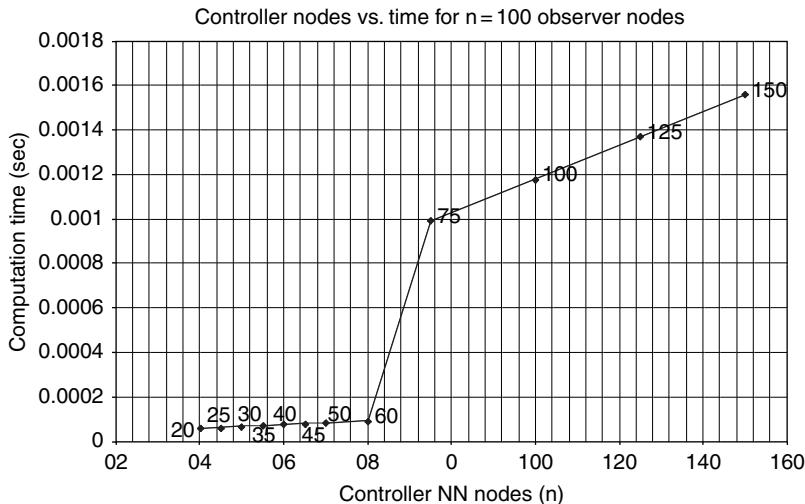


FIGURE 10.8 Lean NN controller runtimes with number of neurons.

As seen in Figure 10.8, increasing the number of controller nodes from 60 to 75 leads to a sharp increase in calculation time. This is due to the processor having to use memory external to its cache. Nonetheless, even at 100 controller nodes and 100 observer nodes, calculations are complete within 1.2 msec, well below the available time of 17.667 msec. Next, the details of the output feedback controller development, simulation and experimental results for lean engine operation and with high EGR levels are described.

10.3 LEAN ENGINE CONTROLLER DESIGN AND IMPLEMENTATION

Today's automobiles utilize sophisticated microprocessor-based engine control systems to meet stringent Federal regulations governing fuel economy and the emission of carbon monoxide (CO), oxides of nitrogen (NO_x), and hydrocarbons (HC). The engine control system can be classified into three categories (Dudek and Sain 1989): the spark advance (SA) control, the air-fuel ratio (A/F) control, and the EGR control. Current engine control systems operate at stoichiometric conditions. However, the more recent concerns about global warming due to greenhouse gases such as carbon dioxide are now shifting the objective of automotive combustion control. Current efforts are tailored to decrease the total amount of emissions and to reduce the fuel consumption.

To address these two requirements, lean combustion control technology receives increased preference (Inoue et al. 1993). One of the difficulties of operating an engine at lean conditions, as already mentioned in Chapter 6, is that the engine exhibits cyclic dispersion (Daw et al. 1998) in heat release. As a result, the engine produces partial and total misfires with unsatisfactory performance and therefore control schemes are necessary. Lean engine operation is usually specified using the equivalence ratio φ . In order to be extremely lean, the engine must operate at $\varphi \leq 0.75$.

While NO_x emissions peak at a slightly lean equivalence ratio of approximately 0.9, NO_x emissions fall with a continued decreasing equivalence ratio. In fact, operation of SI engines at very lean equivalence ratios can dramatically lower the requirements placed on catalytic reduction of NO_x using exhaust after-treatment devices. Major constraints to practical lean operation have been the large number of partial burns and ultimately complete cycle misfires. Partial burns result in significant increase in unburned hydrocarbons and a single misfire is capable of destroying modern catalytic converters. The engine operating very lean exhibits cyclic dispersion in heat release (Daw et al. 1996; Davis et al. 1999) though the weak mixture shows good tolerance to knocking. Moreover, the cyclic dispersion impacts overall performance, operator satisfaction and it makes the lean combustion controller design very difficult.

With the adoption of a three-way catalyst, the engine must work within a narrow window near the stoichiometric air fuel ratio, which influences the peak combustion temperature, the heat transfer losses, the amount of disassociation, and the maximum compression ratio that can assure knock free operation (Martin et al. 1988; Van Nieuwstadt 2000; Itoyama et al. 2002). At these near-stoichiometric conditions, charge dilution with recirculated exhaust gas (EGR) is commonly used as a means to reduce the engine-out NO_x emissions from the internal combustion engines under normal operation. Moreover diesel engines and some nonautomotive SI engines (e.g., generator sets, industrial use

engines, etc.) currently do not use catalysts and therefore dilution with EGR is used extensively as a method of reducing NO_x.

The majority of the EGR introduced into the intake charge is inert and acts as a heat sink during the combustion process, thus lowering the flame front temperature and not only reducing thermal NO_x but also impacting flame speed. The amount of EGR that can be introduced is governed by the stable operating limit that relates to cyclic dispersion in the combustion process caused by lower flame speeds (Heywood 1988), very similar to the problems encountered at lean operation. The proposed lean engine operation is therefore applicable for engines that do not use catalysts. In addition, it is found in Sutton and Drallmeier (2000) that engines exhibit cyclic dispersion in heat release when dilution with EGR is used. Therefore, if a controller is developed first for lean engine operation, then it could be applied to control the engine during dilution with EGR after minor modifications.

Cyclic dispersion has been examined at length and has been identified as a limiting factor in increasing fuel efficiency and reducing emissions (Kantor 1984; Daily 1988; Martin et al. 1988; Inoue et al. 1993; Ozdor et al. 1994; Daw et al. 1996; Itoyama et al. 2002) either with very lean operation or with high levels of EGR (Lunsden et al. 1997; Sutton and Drallmeier 2000). Recent studies (Wagner et al. 1998a, 1998b) on SI engines have been conducted on the development of cyclic dispersion under lean combustion conditions and the influence of EGR (Lunsden et al. 1997; Sutton and Drallmeier 2000). In the lean combustion studies, investigators (Wagner et al. 1999) have found that the cyclic dispersion exhibits both stochastic and deterministic effects with residual fuel and air as the feedforward mechanism. The stochastic effects are rooted in random fluctuations in intake port dynamics, fuel-air charge mixing, and mass of injected fuel. Due to the complex dynamics exhibited by engines operating very lean and with high levels of EGR and these dynamics being nonlinear, unknown, and sensitive to parameter fluctuations, current nonlinear adaptive control schemes (He and Jagannathan 2005) are not suitable. Finally, controller implementations based on continuous-time analysis, if not properly done tend to have performance problems (Jagannathan and Lewis 1996).

Various control schemes have been developed in the literature for lean combustion control. Inoue et al. (1993) designed a lean combustion engine control system using a combustion pressure sensor. Davis et al. (1999) developed a feedback control approach to reduce the cyclic dispersion at lean conditions. However, only the fuel intake system is controlled without considering the air intake. Consequently, significant cyclic dispersion is still left though a control scheme is used. He and Jagannthan (2003) proposed a nonlinear backstepping controller to keep a stable operation of the SI engine at extremely lean conditions ($\varphi \leq 0.75$) by altering the fuel intake (control variable) based on the air intake. All these methods require the precise mathematical model of the

engine and its dynamics. However, the analysis of cyclic dispersion process is difficult due to the variations in the delivery of air and fuel into the cylinder, the fluid dynamics effects during engine intake, exhaust strokes, residual gas fraction, and many other uncertain phenomena. Differences between model and real engine dynamics could jeopardize the controller performance.

In Chapter 6, a novel controller is developed by assuming that the states such as total mass of fuel and air in the cylinder prior to combustion are available for measurement. In this chapter, a direct adaptive NN output feedback controller is proposed for stable operation of the SI engine at extreme lean conditions. A nonlinear system of the following equations (see Section 3.1) $x_1(k+1) = f_1(x_1(k), x_2(k)) + g_1(x_1(k), x_2(k))x_2(k) + d'_1(k)$ and $x_2(k+1) = f_2(x_1(k), x_2(k)) + g_2(x_1(k), x_2(k))u(k) + d'_2(k)$ is used to describe the engine dynamics at lean operation, where $f_1(x_1(k), x_2(k))$, $g_1(x_1(k), x_2(k))$, $f_2(x_1(k), x_2(k))$, and $g_2(x_1(k), x_2(k))$ are unknown nonlinear functions. The control objective then is to reduce the cyclic dispersion in heat release by reducing variation in equivalence ratios ($\varphi(k) = (1/R)x_2(k)/x_1(k)$, where R is a constant) by measuring heat release as the output parameter.

Controlling such a class of nonstrict feedback nonlinear systems is extremely difficult because the control input cannot directly influence the state $x_1(k)$. Moreover, the objective here is to show the boundedness of both the states close to their respective targets so that the actual equivalence ratio is close to its target and bounded tightly, then the cyclic dispersion can be reduced significantly. It is important to note that an observer needs to be used to obtain the total mass of air and fuel estimates. However, the imprecise knowledge of the nonlinear dynamics of the engine makes the design of the observer and the controller more difficult. Design of the observer for an uncertain nonlinear nonstrict feedback system and proving stability of the closed-loop system are even more difficult and challenging since the separation principle normally used to design output feedback control schemes for linear systems is not valid for nonlinear systems. In fact, even an exponentially decaying state estimation error for nonlinear systems can lead to instability in finite time (Krstic et al. 1995).

It is important to note that in adaptive backstepping (Jagannathan 2001), the controlled system is limited to strict feedback nonlinear systems and the state $x_1(k)$ is bounded tightly to its desired target. Moreover, the linearity in the parameter assumption is necessary. By contrast, this assumption is relaxed with the proposed scheme and boundedness of both the states close to their respective targets is demonstrated.

To make the controller practical, heat release-based NN-based output feedback controller is proposed to realize the stable operation of SI engine at lean conditions. Several output feedback controller designs in discrete-time are proposed for the single-input-single-output (SISO) nonlinear systems

(Yeh and Kokotovic 1995). In particular, a backstepping-based adaptive output feedback controller scheme is presented (Yeh and Kokotovic 1995) for the control of a class of strict feedback nonlinear systems, where a rank condition is required to ensure the boundedness of all signals and linear in the unknown parameter (LIP) assumption is utilized. The results presented in this chapter are taken from Vance (2005) and Vance et al. (2005), and in this work the above-mentioned limitations are relaxed.

Two NNs are employed to learn the unknown nonlinear dynamics for the nonstrict feedback nonlinear system since the residual gas fraction and combustion efficiency are considered unknown. Even for the nonstrict feedback nonlinear system, one can use backstepping approach to design the control input (injected fuel) to the total fuel system. The total fuel is then treated as the virtual control signal to the air system and boundedness of both the states tightly to their respective targets will be demonstrated by suitably designing the actual and virtual control inputs. This selection leads to a tight bound on equivalence ratio error by minimizing its variations. As a result, the cyclic dispersion in heat release will be reduced. Exact knowledge of the engine dynamics is not required and therefore the NN controller is model-free. The stability analysis of the closed-loop control system is presented and the boundedness of the closed-loop signals is demonstrated. The NN weights are tuned online with no off-line learning phase required.

The proposed NN controller design is even applicable to a class of nonlinear systems that have a similar structure to the engine dynamics. In other words, this approach from Vance et al. (2005) is not limited to the control of nonstrict feedback nonlinear systems and can be used even for strict feedback nonlinear systems.

10.3.1 ENGINE DYNAMICS

The SI engine dynamics according to the Daw model can be expressed in the following form (Daw et al. 1996):

$$x_1(k+1) = AF(k) + F(k)x_1(k) - R \cdot F(k)CE(k)x_2(k) + d_1(k) \quad (10.1)$$

$$x_2(k+1) = (1 - CE(k))F(k)x_2(k) + (MF(k) + u(k)) + d_2(k) \quad (10.2)$$

$$y(k) = x_2(k)CE(k) \quad (10.3)$$

$$\varphi(k) = R \frac{x_2(k)}{x_1(k)} \quad (10.4)$$

$$CE(k) = \frac{CE_{\max}}{1 + 100^{-(\varphi(k) - \varphi_m)/(\varphi_u - \varphi_l)}} \quad (10.5)$$

and

$$\varphi_m = \frac{\varphi_u - \varphi_l}{2} \quad (10.6)$$

where $x_1(k)$ and $x_2(k)$ are the total mass of air and fuel, respectively, in the cylinder before the k th burn, $y(k)$ is the heat release at k th instant, $CE(k)$ is the combustion efficiency, and $0 < CE_{\min} < CE(k) < CE_{\max}$, CE_{\max} is the maximum combustion efficiency — a constant, $F(k)$ is the residual gas fraction and $0 < F_{\min} < F(k) < F_{\max}$, $AF(k)$ is the mass of fresh air fed per cycle, R is the stoichiometric A/F ratio, approximately 15.13, $MF(k)$ is the mass of fresh fuel per cycle, $u(k)$ is the change in mass of fresh fuel per cycle, $\varphi(k)$ is the input equivalence ratio, φ_m , φ_u , φ_l are constant system parameters, and $d_1(k)$ and $d_2(k)$ are unknown but bounded disturbances. Since $y(k)$ varies cycle by cycle, the engine performance degrades and ultimately becomes unsatisfactory. In the above engine dynamics, both $F(k)$ and $CE(k)$ are unknown nonlinear functions of $x_1(k)$ and $x_2(k)$.

Remark 1: States $x_1(k)$ and $x_2(k)$ are typically not measurable and the output of $y(k)$ is only available. The control objective is to operate an engine at lean conditions ($0 < \varphi(k) < 1$) with only heat release information — to stabilize $y(k)$ around a target heat release value y_d .

Remark 2: We notice that, in (10.3), the available system output $y(k)$ is an unknown nonlinear function of both $x_1(k)$ and $x_2(k)$, unlike in all past literatures (Yeh and Kokotovic 1995; He and Jagannathan 2005), where $y(k) = x_1(k)$ or $y(k)$ is considered a known linear combination of system states. This relationship makes the observer design more challenging. Substituting (10.3) into both (10.1) and (10.2), we get

$$x_1(k+1) = AF(k) + F(k)x_1(k) - R \cdot F(k)y(k) + d_1(k) \quad (10.7)$$

$$x_2(k+1) = F(k)(x_2(k) - y(k)) + (MF(k) + u(k)) + d_2(k) \quad (10.8)$$

For real engine operation, the fresh air, $AF(k)$, the fresh fuel, $MF(k)$, and the residual gas fraction, $F(k)$, can all be viewed as nominal values plus some small and bounded disturbances such as

$$AF(k) = AF_0 + \Delta AF(k) \quad (10.9)$$

$$MF(k) = MF_0 + \Delta MF(k) \quad (10.10)$$

and

$$F(k) = F_0 + \Delta F(k) \quad (10.11)$$

where AF_0 , MF_0 , and F_0 are known nominal fresh air, fresh fuel, and residual gas fraction values, respectively. Here the terms $\Delta AF(k)$, $\Delta MF(k)$, and $\Delta F(k)$ are small, unknown but bounded disturbances for fresh air, fresh fuel, and residual gas fraction, respectively. Their bounds are given by

$$0 \leq |\Delta AF(k)| \leq \Delta AF_m \quad (10.12)$$

$$0 \leq |\Delta MF(k)| \leq \Delta MF_m \quad (10.13)$$

and

$$0 \leq |\Delta F(k)| \leq \Delta F_m \quad (10.14)$$

where ΔAF_m , ΔMF_m , and ΔF_m are their respective upper bounds for $\Delta AF(k)$, $\Delta MF(k)$, and $\Delta F(k)$.

Combining (10.9) through (10.11) with (10.7) and (10.8), and rewriting (10.7) and (10.8) to get

$$\begin{aligned} x_1(k+1) = & AF_0 + F_0 x_1(k) - R \cdot F_0 \cdot y(k) + \Delta AF(k) + \Delta F(k)x_1(k) \\ & - R\Delta F(k)y(k) + d_1(k) \end{aligned} \quad (10.15)$$

$$\begin{aligned} x_2(k+1) = & F_0(x_2(k) - y(k)) + (MF_0 + u(k)) + \Delta F(k)(x_2(k) - y(k)) \\ & + \Delta MF(k) + d_2(k) \end{aligned} \quad (10.16)$$

Now, at the k th step using (10.3), let us predict the future heat release, $y(k+1)$

$$y(k+1) = x_2(k+1)CE(k+1) = f_3(x_1(k), x_2(k), y(k), u(k)) \quad (10.17)$$

where $f_3(x_1(k), x_2(k), y(k), u(k))$ is an unknown nonlinear function of states, output, and the input.

10.3.2 NN OBSERVER DESIGN

A two-layer NN will be employed to predict the heat release value at the subsequent time step and the heat release prediction error is utilized to design the system observer. From (10.17), $y(k+1)$ can be approximated by using a one-layer NN as

$$y(k+1) = w_1^T \phi_1(v_1^T z_1(k)) + \varepsilon_1(z_1(k)) \quad (10.18)$$

where the input to the NN is taken as $z_1(k) = [x_1(k), x_2(k), y(k), u(k)]^T \in R^4$, the matrix $w_1 \in R^{n_1}$ and $v_1 \in R^{4 \times n_1}$ represent the target output- and hidden-layer weights, $\phi_1(\cdot)$ represents the hidden-layer activation function, n_1 denotes

the number of the nodes in the hidden layer, and $\varepsilon(z_1(k)) \in R$ is the functional approximation error. It is demonstrated in Igelnik and Pao (1995) that, if the hidden-layer weight, v_1 , is chosen initially at random and held constant and the number of hidden-layer nodes is sufficiently large, the approximation error $\varepsilon(z_1(k))$ can be made arbitrarily small over the compact set since the activation function forms a basis.

For simplicity we define

$$\phi_1(z_1(k)) = \phi_1(v_1^T z_1(k)) \quad (10.19)$$

and

$$\varepsilon_1(k) = \varepsilon(z_1(k)) \quad (10.20)$$

Given (10.19) and (10.20), (10.18) is rewritten as

$$y(k+1) = w_1^T \phi_1(z_1(k)) + \varepsilon_1(k) \quad (10.21)$$

Since states $x_1(k)$ and $x_2(k)$ are not measurable, $z_1(k)$ is not available. Using the estimated values for the states $\hat{x}_1(k)$, $\hat{x}_2(k)$, and $\hat{y}(k)$ instead of $x_1(k)$, $x_2(k)$, and $y(k)$, the proposed heat release observer can be written as

$$\hat{y}(k+1) = \hat{w}_1^T(k) \phi_1(v_1^T \hat{z}_1(k)) + l_1 \tilde{y}(k) = \hat{w}_1^T(k) \phi_1(\hat{z}_1(k)) + l_1 \tilde{y}(k) \quad (10.22)$$

where $\hat{y}(k+1)$ is the predicted heat release, $\hat{w}_1(k) \in R^{n_1}$ the actual output-layer weights, $\hat{z}_1(k) = [\hat{x}_1(k), \hat{x}_2(k), \hat{y}(k), u(k)]^T \in R^4$ the NN input, $l_1 \in R$ the observer gain, and $\tilde{y}(k)$ is the heat release estimation error, which is defined as

$$\tilde{y}(k) = \hat{y}(k) - y(k) \quad (10.23)$$

and $\phi_1(\hat{z}_1(k))$ represents $\phi_1(v_1^T \hat{z}_1(k))$ for simplicity.

Using the heat release estimation error, the proposed observer is given as

$$\hat{x}_1(k+1) = AF_0 + F_0 \hat{x}_1(k) - R \cdot F_0 \cdot \hat{y}(k) + l_2 \tilde{y}(k) \quad (10.24)$$

and

$$\hat{x}_2(k+1) = F_0(\hat{x}_2(k) - \hat{y}(k)) + (MF_0 + u(k)) + l_3 \tilde{y}(k) \quad (10.25)$$

where $l_2 \in R$ and $l_3 \in R$ are observer gains. Here, it is assumed that the initial value of the actual control input $u(0)$ is bounded. Equation 10.22,

Equation 10.24, and Equation 10.25 are the proposed observer equations for the estimation of $x_1(k)$ and $x_2(k)$.

Define the state estimation errors as

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k) \quad i = 1, 2 \quad (10.26)$$

Combining (10.21) through (10.26), we obtain the estimation error dynamics as

$$\begin{aligned} \tilde{x}_1(k+1) &= F_0\tilde{x}_1(k) + (l_2 - R \cdot F_0)\tilde{y}(k) - \Delta AF(k) - \Delta F(k)x_1(k) \\ &\quad + R\Delta F(k)y(k) - d_1(k) \end{aligned} \quad (10.27)$$

$$\begin{aligned} \tilde{x}_2(k+1) &= F_0\tilde{x}_2(k) + (l_3 - F_0)\tilde{y}(k) - \Delta F(k)(x_2(k) - y(k)) \\ &\quad - \Delta MF(k) - d_2(k) \end{aligned} \quad (10.28)$$

and

$$\begin{aligned} \tilde{y}(k+1) &= \hat{w}_1^T(k)\phi_1(\hat{z}_1(k)) + l_1\tilde{y}(k) - w_1^T\phi_1(z_1(k)) - \varepsilon_1(k) \\ &= (\hat{w}_1(k) - w_1)^T\phi_1(\hat{z}_1(k)) + w_1^T(\phi_1(\hat{z}_1(k)) - \phi_1(z_1(k))) - \varepsilon_1(k) \\ &= \tilde{w}_1^T(k)\phi_1(\hat{z}_1(k)) + w_1^T(k)\phi_1(\tilde{z}_1(k)) - \varepsilon_1(k) \\ &= \xi_1(k) + w_1^T(k)\phi_1(\tilde{z}_1(k)) - \varepsilon_1(k) \end{aligned} \quad (10.29)$$

where

$$\tilde{w}(k) = \hat{w}(k) - w_1 \quad (10.30)$$

$$\xi_1(k) = \tilde{w}_1^T(k)\phi_1(\hat{z}_1(k)) \quad (10.31)$$

and for simplicity, $(\phi_1(\hat{z}_1(k)) - \phi_1(z_1(k)))$ is written as $\phi_1(\tilde{z}_1(k))$.

10.3.3 ADAPTIVE NN OUTPUT FEEDBACK CONTROLLER DESIGN

At lean condition and without any control, significant amount of cyclic dispersion in heat release can be observed which results in unsatisfactory performance in terms of poor conversion fuel efficiency. In order to allow the engine operation at lean conditions, our control objective is to reduce the cyclic dispersion in heat release — drive the actual cylinder heat release value to the operating point of y_d . Given y_d and the engine dynamics (10.1) through (10.5), we could obtain the operating point of total mass of air and fuel in the cylinder, x_{1d} and x_{21d} , respectively. By driving both the states $x_1(k)$ and $x_2(k)$ to approach their respective target values x_{1d} and x_{21d} , $y(k)$ will be closer to its desired value y_d .

10.3.3.1 Adaptive NN Backstepping Design

With the estimated states $\hat{x}_1(k)$ and $\hat{x}_2(k)$, the controller design follows the backstepping technique detailed in the subsequent steps.

Step 1: Virtual controller design

Define the system error as

$$e_1(k) = x_1(k) - x_{1d} \quad (10.32)$$

Evaluating (10.32) at the subsequent time step and combining with (10.1) to get

$$\begin{aligned} e_1(k+1) &= x_1(k+1) - x_{1d} = AF(k) + F(k)x_1(k) - x_{1d} \\ &\quad - R \cdot F(k)CE(k)x_2(k) + d_1(k) \end{aligned} \quad (10.33)$$

For simplicity, let us denote

$$f_1(k) = AF(k) + F(k)x_1(k) - x_{1d} \quad (10.34)$$

and

$$g_1(k) = R \cdot F(k)CE(k) \quad (10.35)$$

Then the system error equation can be expressed as

$$e_1(k+1) = f_1(k) - g_1(k)x_2(k) + d_1(k) \quad (10.36)$$

By viewing $x_2(k)$ as a virtual control input, a desired feedback control signal can be designed as

$$x_{2d}(k) = \frac{f_1(k)}{g_1(k)} \quad (10.37)$$

The term $x_{2d}(k)$ can be approximated by using the first NN as

$$x_{2d}(k) = w_2^T \phi_2(v_2^T x(k)) + \varepsilon_2(x(k)) = w_2^T \phi_2(x(k)) + \varepsilon_2(x(k)) \quad (10.38)$$

where the NN input is given by the state $x(k) = [x_1(k), x_2(k)]^T$, $w_2 \in R^{n_2}$, and $v_2 \in R^{2 \times n_1}$ denote the constant target output- and hidden-layer weights, n_2 is the number of hidden-layer nodes with the hidden-layer activation function $\phi_2(v_2^T x(k))$ abbreviated as $\phi_2(x(k))$, and $\varepsilon_2(x(k))$ is the approximation error.

Since both $x_1(k)$ and $x_2(k)$ are unavailable, the actual state is replaced by its estimate $\hat{x}(k)$ and used as the NN input. Consequently, the virtual control input is taken as

$$\hat{x}_{2d}(k) = \hat{w}_2^T(k) \phi_2(v_2^T \hat{x}(k)) = \hat{w}_2^T(k) \phi_2(\hat{x}(k)) \quad (10.39)$$

where $\hat{w}_2^T(k) \in R^{n_2}$ is the actual weight matrix for the first NN. Define the weight estimation error by

$$\tilde{w}_2(k) = \hat{w}_2(k) - w_2 \quad (10.40)$$

Define the error between the desired and actual virtual control inputs $x_2(k)$ and $\hat{x}_{2d}(k)$ as

$$e_2(k) = x_2(k) - \hat{x}_{2d}(k) \quad (10.41)$$

Equation 10.36 can be expressed using (10.41) for $x_2(k)$ as

$$e_1(k+1) = f_1(k) - g_1(k)(e_2(k) + \hat{x}_{2d}(k)) + d_1(k) \quad (10.42)$$

or, equivalently,

$$\begin{aligned} e_1(k+1) &= f_1(k) - g_1(k)(e_2(k) + x_{2d}(k) - x_{2d}(k) + \hat{x}_{2d}(k)) + d_1(k) \\ &= -g_1(k)(e_2(k) - x_{2d}(k) + \hat{x}_{2d}(k)) + d_1(k) \\ &= -g_1(k)(e_2(k) + \hat{w}_2^T(k) \phi_2(\hat{x}(k)) \\ &\quad - w_2^T(k) \phi_2(x(k)) - \varepsilon_2(x(k))) + d_1(k) \end{aligned} \quad (10.43)$$

Equation 10.43 can be further expressed as

$$e_1(k+1) = -g_1(k)(e_2(k) - \zeta_2(k) + w_2^T \phi_2(\tilde{x}(k)) - \varepsilon_2(x(k))) + d_1(k) \quad (10.44)$$

where

$$\zeta_2(k) = \tilde{w}_2^T(k) \phi_2(\hat{x}(k)) \quad (10.45)$$

and

$$w_2^T \phi_2(\tilde{x}(k)) = w_2^T (\phi_2(\hat{x}(k)) - \phi_2(x(k))) \quad (10.46)$$

Step 2: Design of the actual control input $u(k)$

Rewriting the error $e_2(k)$ from (10.41) as

$$\begin{aligned} e_2(k+1) &= x_2(k+1) - \hat{x}_{2d}(k+1) \\ &= (1 - CE(k))F(k)x_2(k) + (MF(k) + u(k)) - \hat{x}_{2d}(k+1) + d_2(k) \end{aligned} \quad (10.47)$$

where for simplicity, let us denote

$$f_2(k) = (1 - CE(k))F(k)x_2(k) + MF(k) \quad (10.48)$$

Equation 10.47 can be written as

$$e_2(k+1) = f_2(k) + u(k) - \hat{x}_{2d}(k+1) + d_2(k) \quad (10.49)$$

where $\hat{x}_{2d}(k+1)$ is the value of $\hat{x}_{2d}(k)$ at the subsequent time step which is required in the current time instant. Since it is not available in the current time step, using (10.37) and (10.39), one can approximate the future value by using a dynamical/recurrent NN provided it is a smooth function of measurable variables. It is important to note that $\hat{x}_{2d}(k+1)$ is a smooth nonlinear function of the state $x(k)$ and the virtual control input $\hat{x}_{2d}(k)$. Consequently, the term $\hat{x}_{2d}(k+1)$ can be approximated by using another NN since it is a one-step head predictor. Alternatively, one can use the approach presented in the backlash compensator design of Chapter 4 (Lewis et al. 2002). In any case, it is important to note that for these classes of systems, the causal problem issue does not arise since a one-step predictor is employed.

Select the desired control input by using the second NN in the controller design as

$$\begin{aligned} u_d(k) &= (-f_2(k) + \hat{x}_{2d}(k+1)) = w_3^T \phi_3(v_3^T z_3(k)) + \varepsilon_3(z_3(k)) \\ &= w_3^T \phi_3(z_3(k)) + \varepsilon_3(z_3(k)) \end{aligned} \quad (10.50)$$

where $w_3 \in R^{n_3}$ and $v_3 \in R^{3 \times n_3}$ denote the constant target output and hidden-layer weights, n_3 the number of hidden-layer nodes with the activation function $\phi_3(v_3^T z_3(k))$ abbreviated by $\phi_3(z_3(k))$, $\varepsilon_3(z_3(k))$ is the approximation error, and $z_3(k) \in R^3$ is the NN input defined by (10.51). Considering the fact that both

$x_1(k)$ and $x_2(k)$ cannot be measured, the NN input $z_3(k)$ is substituted with $\hat{z}_3(k) \in R^3$, where

$$z_3(k) = [x(k), \hat{x}_{2d}(k)]^T \in R^3 \quad (10.51)$$

and

$$\hat{z}_3(k) = [\hat{x}(k), \hat{x}_{2d}(k)]^T \in R^3 \quad (10.52)$$

Define

$$\hat{e}_1(k) = \hat{x}_1(k) - x_{1d} \quad (10.53)$$

and

$$\hat{e}_2(k) = \hat{x}_2(k) - x_{2d} \quad (10.54)$$

The actual control input is now selected as

$$u(k) = \hat{w}_3^T(k)\phi_3(v_3^T\hat{z}_3(k)) + l_4\hat{e}_2(k) = \hat{w}_3^T(k)\phi_3(\hat{z}_3(k)) + l_4\hat{e}_2(k) \quad (10.55)$$

where $\hat{w}_3(k) \in R^{n_3}$ is the actual output-layer weight vector and $l_4 \in R$ is the controller gain selected to stabilize the system. Similar to the derivation of (10.29), combining (10.49), (10.50), and (10.55) yields

$$e_2(k+1) = l_4\hat{e}_2(k) + \xi_3(k) + w_3^T\phi_3(\tilde{z}_3(k)) - \varepsilon_3(z_3(k)) + d_2(k) \quad (10.56)$$

where

$$\tilde{w}_3(k) = \hat{w}_3(k) - w_3 \quad (10.57)$$

$$\xi_3(k) = \hat{w}_3^T(k)\phi_3(\hat{z}_3(k)) \quad (10.58)$$

and

$$w_3^T\phi_3(\tilde{z}(k)) = w_3^T(\phi_3(\hat{z}_3(k)) - \phi_3(z_3(k))) \quad (10.59)$$

Equation 10.44 and Equation 10.56 represent the closed-loop error dynamics. It is necessary now to show that the estimation errors (10.23) and (10.26), the system errors (10.44) and (10.56), and the NN weight matrices $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded.

10.3.3.2 Weight Updates for Guaranteed Performance

Assumption 10.3.1 (Bounded Ideal Weights): Let w_1 , w_2 , and w_3 be the unknown output-layer target weights for the observer and two action NNs and assume that they are bounded above so that

$$\|w_1\| \leq w_{1m} \quad \|w_2\| \leq w_{2m} \quad \text{and} \quad \|w_3\| \leq w_{3m} \quad (10.60)$$

where $w_{1m} \in R^+$, $w_{2m} \in R^+$, and $w_{3m} \in R^+$ represent the bounds on the unknown target weights where the Frobenius norm is used.

Fact 10.3.1: The activation functions are bounded above by known positive values so that

$$\|\phi_i(\cdot)\| \leq \phi_{im} \quad i = 1, 2, 3 \quad (10.61)$$

where ϕ_{im} , $i = 1, 2, 3$ are the upper bounds.

Assumption 10.3.2 (Bounded NN Approximation Error): The NN approximation errors $\varepsilon_1(z_1(k))$, $\varepsilon_2(x(k))$, and $\varepsilon_3(z_3(k))$ are bounded over the compact set by ε_{1m} , ε_{2m} , and ε_{3m} , respectively.

Theorem 10.3.1 (Lean Engine Controller): Consider the system given in (10.1) to (10.3) and let the Assumptions 10.3.1 and 10.3.2 hold. The unknown disturbances are considered to be bounded by $|d_1(k)| \leq d_{1m}$ and $|d_2(k)| \leq d_{2m}$, respectively. Let the observer NN weight tuning be provided by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(\hat{z}_1(k)) (\hat{w}_1^T(k) \phi_1(\hat{z}_1(k)) + l_5 \tilde{y}(k)) \quad (10.62)$$

with the virtual control NN weight tuning given by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{x}(k)) (\hat{w}_2^T(k) \phi_2(\hat{x}(k)) + l_6 \hat{e}_1(k)) \quad (10.63)$$

and the control input NN weights be tuned by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(\hat{z}_3(k)) (\hat{w}_3^T(k) \phi_3(\hat{z}_3(k)) + l_7 \hat{e}_2(k)) \quad (10.64)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$ and $l_5 \in R$, $l_6 \in R$, and $l_7 \in R$ are design parameters. Consider the system observer given by (10.22), (10.24), and (10.25), and virtual and actual control inputs defined as (10.39) and (10.55), respectively.

The estimation errors (10.27) to (10.29), the tracking errors (10.44) and (10.56), and the NN weights $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are uniformly ultimately bounded (UUB) with the bounds specifically given by (10.A.17) through (10.A.24) provided the design parameters are selected as:

$$(a) \quad 0 < \alpha_i \|\phi_i(k)\|^2 < 1 \quad i = 1, 2, 3 \quad (10.65)$$

$$(b) \quad l_3^2 < 1 - \frac{(l_1 - R \cdot F_0)^2}{6R^2 \cdot \Delta F_m^2} - \frac{(l_2 - F_0)^2}{6\Delta F_m^2} - 4l_5^2 \quad (10.66)$$

$$(c) \quad l_6^2 < \min \left(\frac{(1 - F_0^2)}{18R^2 \cdot \Delta F_m^2}, \frac{1}{18R^2} \right) \quad (10.67)$$

$$(d) \quad l_4^2 + 6l_7^2 < \min \left(\frac{(1 - F_0^2)}{6\Delta F_m^2}, \frac{1}{3} \right) \quad (10.68)$$

Proof: See Appendix 10.A.

Remark 3: Given specific values of R , F_0 , and ΔF_m , we could derive the design parameters of l_i , $i = 1, 6, 7$. For instance, given $R = 14.6$, $F_0 = 0.14$, and $\Delta F_m = 0.02$, we can select $l_1 = 1.99$, $l_2 = 0.13$, $l_3 = 0.4$, $l_4 = 0.14$, $l_5 = 0.25$, $l_6 = 0.016$, and $l_7 = 0.1667$ to satisfy (10.66) to (10.68).

Remark 4: Given the hypotheses, this proposed NN output NN control scheme and the weight-updating rules in Theorem 10.3.1 with the parameter selection based on (10.65) through (10.68), the state $x_2(k)$ approaches the operating point x_{2d} .

Remark 5: It is important to note that in this theorem there is no persistency of excitation condition (PE), certainty equivalence (CE), and LIP assumptions for the NN observer and NN controller. In our proof, the Lyapunov function shown in the appendix consists of the observer estimation errors, system errors, and the NN estimation errors. Though the proof is exceedingly complex, it obviates the need for the CE assumption and it allows weight-tuning algorithms to be derived during the proof, not selected a priori in an ad hoc manner. Also, separation principle is not applied.

Remark 6: The NN weights are initialized at random or zero. Therefore there is no explicit off-line learning phase.

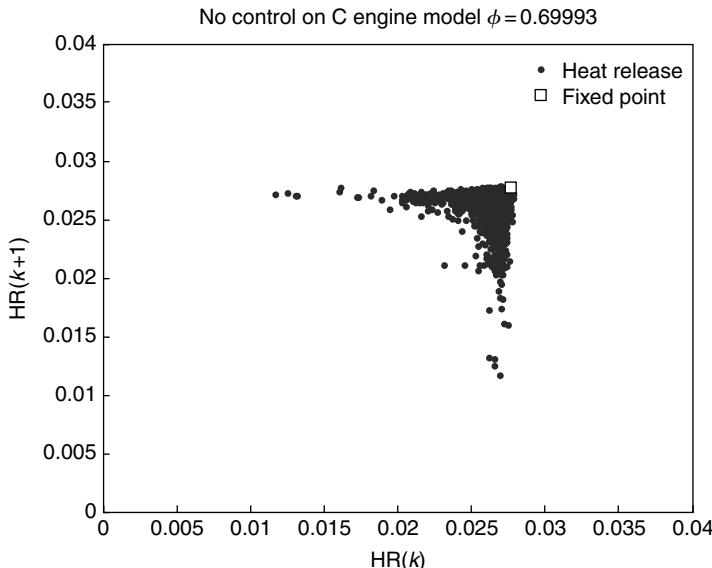


FIGURE 10.9 Uncontrolled Daw engine model results.

10.3.4 SIMULATION OF NN CONTROLLER C IMPLEMENTATION

Using the Daw et al. model to simulate lean engine performance, the parameters are selected as the following: 20,000 cycles are considered at equivalence ratio of 0.699 with $R = 15.13$ (iso-octane), residual gas fraction $F = 0.09$, mass of nominal new air = 0.52485, mass of nominal new fuel = 0.02428, the standard deviation of mass of new fuel is 0.007, cylinder volume in moles = 0.021, molecular weight of fuel = 114, molecular weight of air = 28.84, $\phi_u = 0.665$, $\phi_l = 0.645$, maximum combustion efficiency = 1, and the gains of backstepping controller are selected at 0.1 and placed diagonally. A 2% unknown perturbation was added to the residual gas fraction. The initial values of the errors e_1 and e_2 are taken as 0.9 and 0.6, respectively. Figure 10.9 shows the heat release variation per cycle via a return map without any control whereas Figure 10.10 presents the backstepping lean engine NN controller performance. It is observed that the engine exhibits minimal dispersion at extreme lean conditions with perturbation when the residual gas fraction is unknown.

Figure 10.11 presents the variation in heat release when control is initiated after 10,001th cycle. As expected, the cyclic dispersion in heat release has been minimized due to reduction in variations in combustion efficiency since residual gas fraction amounts are reduced in the cylinder. Figure 10.12 presents the controller output whereas Figure 10.13 depicts the pulse width change when

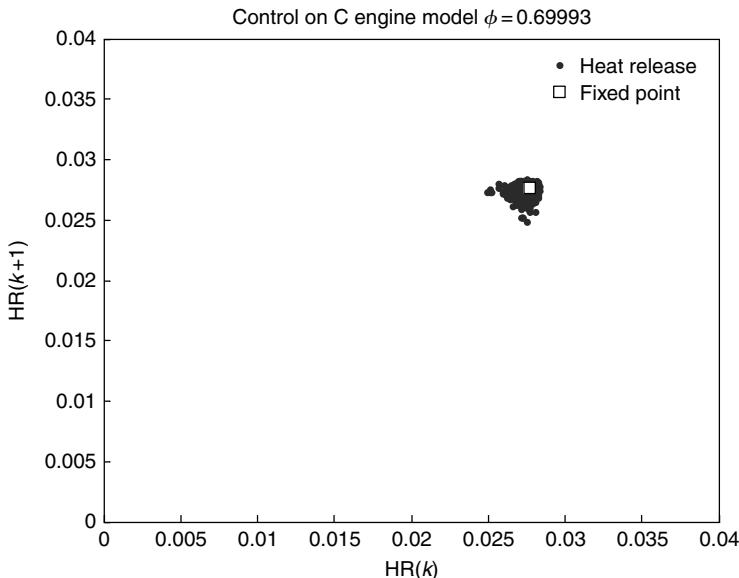


FIGURE 10.10 Controlled model using NN implementation in C.

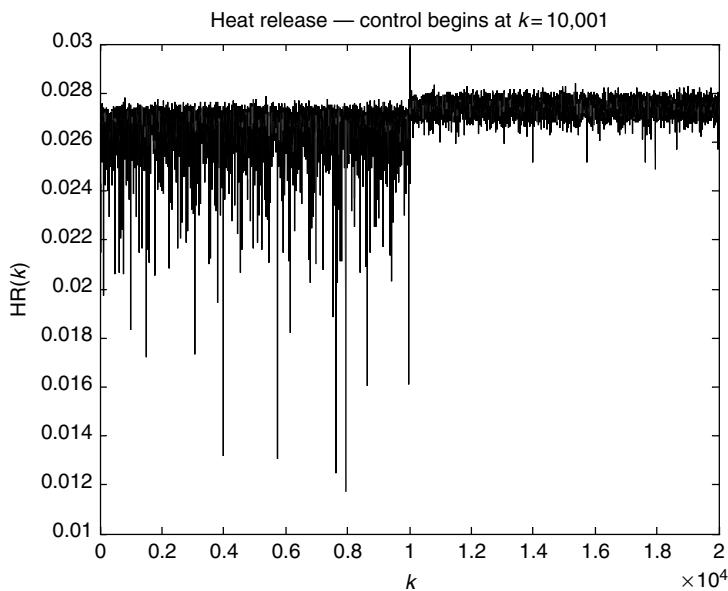


FIGURE 10.11 Heat release before and after NN control (control initiated at the 10,001th cycle).

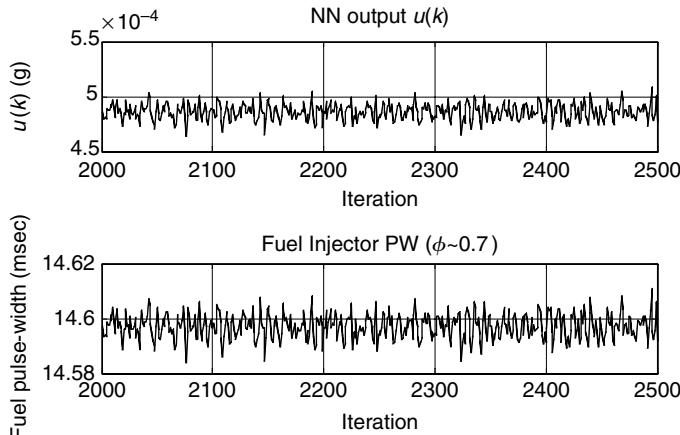


FIGURE 10.12 Controller output.

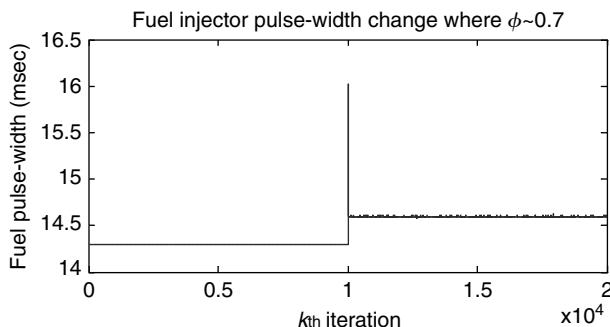


FIGURE 10.13 Pulse-width change before and after control.

the control is initiated. Figure 10.14 shows the performance of the observer in terms of the estimated heat release. From this figure, it is clear that the NN observer is able to predict heat release close to the model value.

10.3.5 EXPERIMENTAL RESULTS

A single-cylinder cooperative fuel research (CFR) engine, which has been used in several other studies of cyclic dynamics and for the proposed work shown in Figure 10.15, incorporates a modern port fuel injection system and is coupled to an electric dynamometer to control engine speed. An in-cylinder

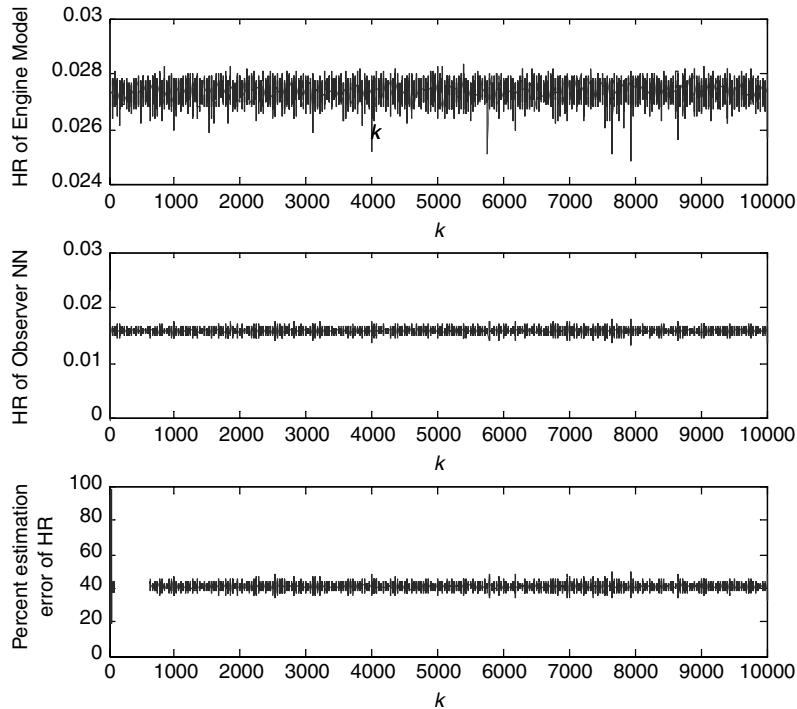


FIGURE 10.14 Comparison of estimated and modeled heat release.

pressure transducer along with high-speed data acquisition system is used to sample crank-angle resolved cylinder pressure for determining cyclic heat release. Analysis programs for heat release have been developed in previous studies (Wagner 1999). Finally the A/F ratio will be sensed in the exhaust stream using a UEGO sensor, where this sensor provides an output proportional to the oxygen content in the exhaust stream. Previous work has indicated that the UEGO sensor should have sufficient response time (Wagner 1995).

The controller will be deemed successful if the lean limit, defined by cyclic variability in heat release, can be extended (equivalence ratio ≤ 0.76). The goal is stable operation with a 5% increase in fuel conversion efficiency from stoichiometric operation. The controller code in C is included in Appendix B. Figure 10.16 depicts the cyclic dispersion in heat release using the CFR engine at an equivalence ratio of 0.707 whereas Figure 10.17 illustrates the cyclic dispersion in heat release with the proposed controller. Here heat release is given in Joules.

By using the metric coefficient of variation, with and without control, one can observe that the coefficient of variation has dropped by over 66%. It appears

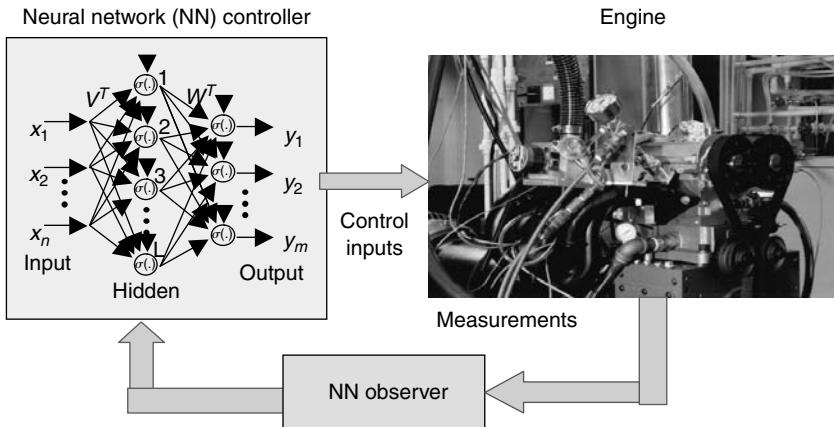


FIGURE 10.15 Output feedback engine control configuration.

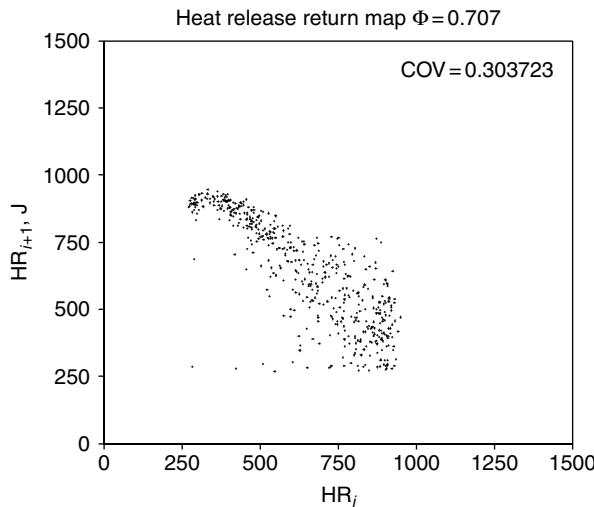


FIGURE 10.16 Cyclic dispersion in heat release when equivalence ratio is 0.707.

that the controller is trying to push the system a bit richer. This is the direct result of the uniformly ultimately boundedness of the closed-loop system where it was shown that the heat release errors will be bounded due to the boundedness of the equivalence ratio. It is important to note that the boundedness of the errors can only be demonstrated and not the asymptotic stability (AS) due to the uncertainties encountered in the engine model such as residual gas fraction,

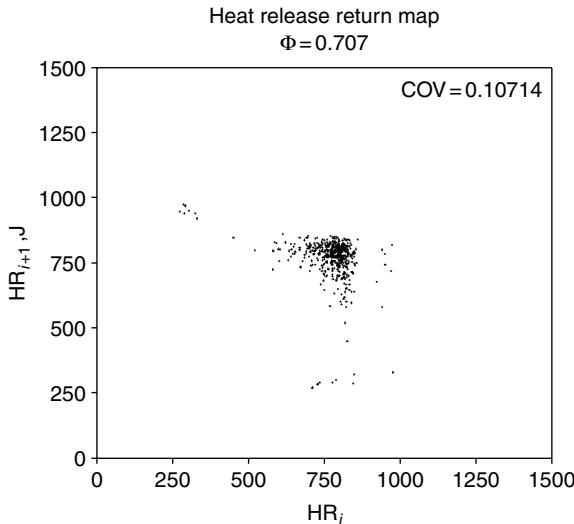


FIGURE 10.17 Cyclic dispersion with NN control.

thermal effects, variations in heat release per cycle due to combustion issues, and errors in instrumentation. Moreover, the target heat release is difficult to determine due to the uncertainty in engine combustion efficiency. Finally, the target air and fuel operating points based on a given heat release is calculated using nominal values and this has resulted in possible error in actual and target heat release. Though the NNs have the capability to learn the uncertainties, suitable inputs to approximate such uncertainties are not fed as inputs to the NNs since these are not available for measurement.

In order to evaluate the effectiveness of the closed-loop control due to the boundedness of errors and control input, another experiment was performed to control the engine at an equivalence ratio of 0.707. When the control action is initiated, it was found that the controller appears to push the system to an equivalence ratio of 0.735, a 3.5% error in equivalence ratio. A comparison has been made between the uncontrolled yet stable operation of the engine at this equivalence ratio of 0.735 with that of the controlled scenario. Figure 10.18 depicts the cyclic dispersion in heat release during uncontrolled case when the engine is allowed to operate at the controlled scenario. One can observe from the coefficient of variation metric that there is a slight improvement in cyclic dispersion. These plots indicate that the controller is able to keep the equivalence ratio within $\pm 3.5\%$ of the target which is a tight bound in general and given that a significant amount of information is not available a priori (see Figure 10.19).

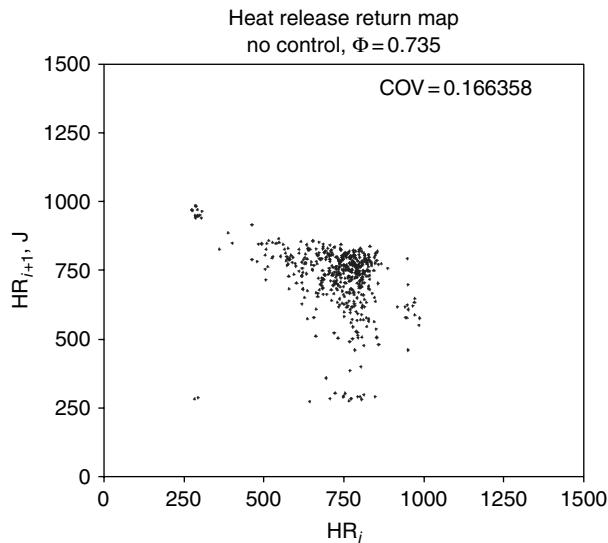


FIGURE 10.18 Cyclic dispersion without control at an equivalence ratio of 0.735.

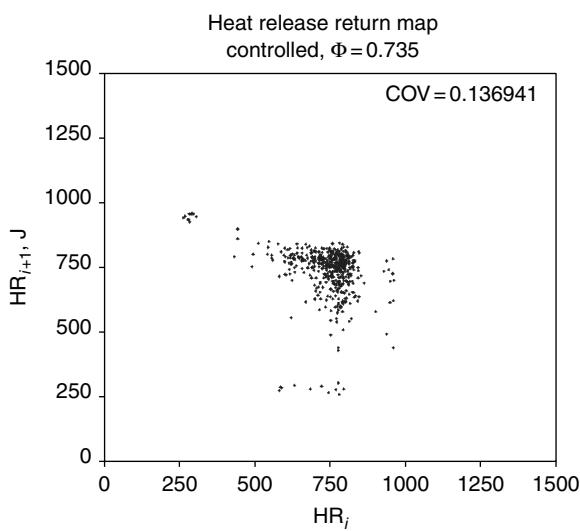


FIGURE 10.19 Cyclic dispersion with NN control.

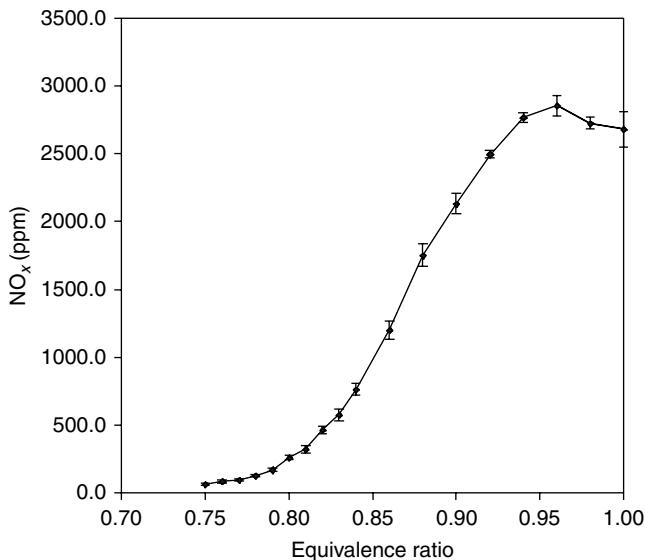


FIGURE 10.20 Reduction in NO_x .

The next set of plots illustrates the reduction in NO_x and fuel intake due to the lean engine operation (see Figure 10.20). From the nitrous oxide plot, it can be observed that the reduction is significant when an engine is forced to operate lean irrespective of open- or closed-loop control. However, a closed-loop engine operation is robust to disturbances and parameter variations such as air flow whereas open-loop control is sensitive to them. Operating engine lean will lower the fuel intake by an amount of 10% as illustrated in Figure 10.21.

Figures 10.22 through 10.24 illustrate the performance of the NN controller on the CFR engine compared to the uncontrolled scenario for different equivalence ratios. Here the heat release values are normalized and are in Joules. Using the coefficient of variation, which represents the amount of cyclic dispersion, one can show that the controller is able to reduce the cyclic dispersion by an amount equal to half, compared to the uncontrolled case. Due to the uncertainty in combustion efficiency, errors in target air and fuel operating points, air flow measurement issues and scaling factors, the fuel intake appears to increase by an amount of 3% in general which in turn causes an increase in equivalence ratio by 3% from the desired value of equivalence ratio. Comparing the uncontrolled plot with that of the controlled scenario after the inclusion of these errors appears to show limited improvement in cyclic dispersion. However, uncontrolled scenarios are normally avoided as they are sensitive to unmodeled dynamics, disturbances, and parameter changes even

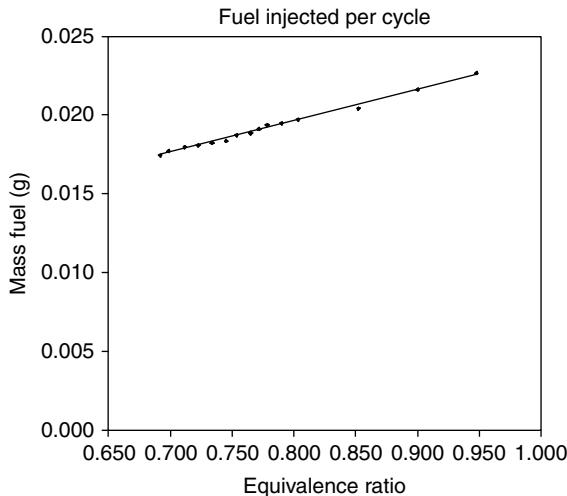


FIGURE 10.21 Fuel intake with equivalence ratio.

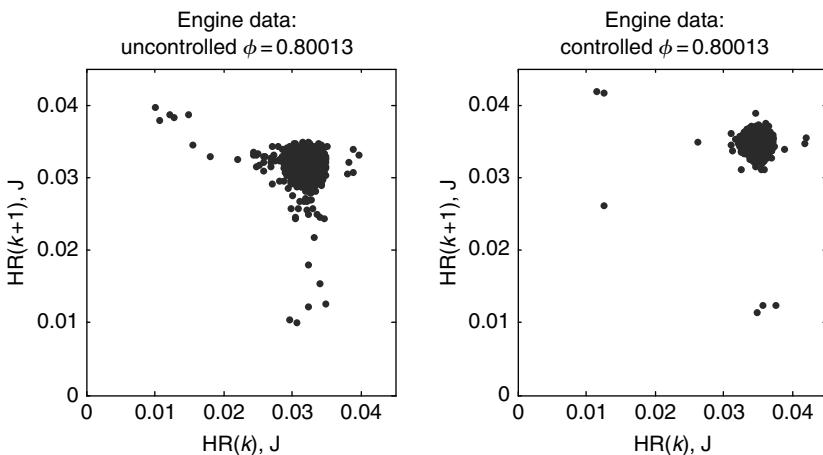


FIGURE 10.22 Performance of the NN controller for an equivalence ratio of 0.8. (a) No control. (b) With NN control.

though during the uncontrolled scenario, the engine system is stable. This is not the case in general for other nonlinear systems.

The above plots show that controllers for the lean operation push the engine system a bit richer. To overcome this issue one has to identify a different control parameter such as ignition timing or spark timing which is not related to fuel

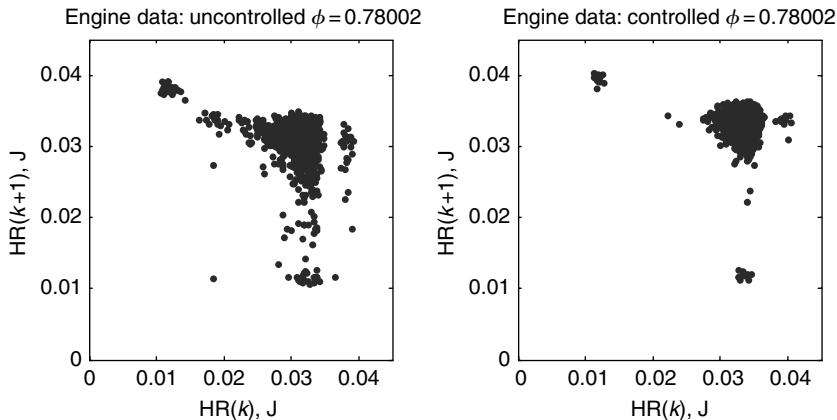


FIGURE 10.23 Performance of the NN controller for an equivalence ratio of 0.78. (a) No control. (b) With NN control.

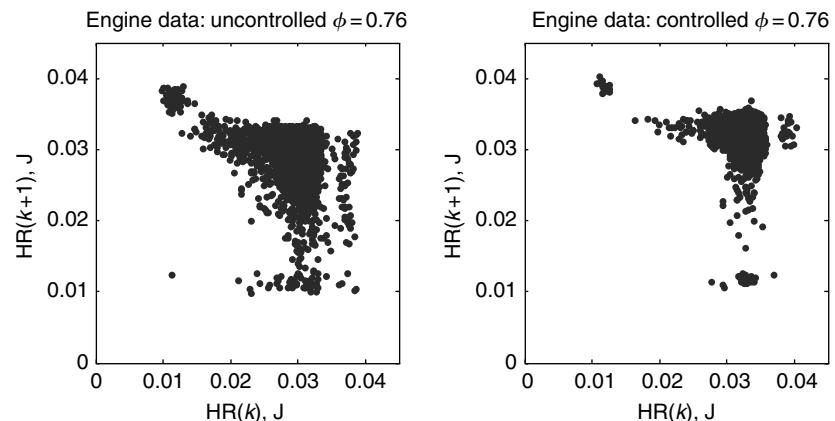


FIGURE 10.24 Performance of the NN controller for an equivalence ratio of 0.76. (a) No control. (b) With NN control.

intake since fuel intake is used both for maintaining the engine at an operating point and to minimize cyclic dispersion. Controlling the ignition or spark timing can help operate an engine at the same equivalence ratio as that of the uncontrolled case since fuel intake is used mainly for changing the operating point whereas ignition or spark timing is used for minimizing cyclic dispersion.

10.4 EGR ENGINE CONTROLLER DESIGN AND IMPLEMENTATION

Studies were performed by Sutton and Drallmeier (2000) on the onset of complex dynamic behavior in an SI engine with high levels of simulated EGR (added nitrogen) and to relate these results to the lean operation. Experiments were performed on a 0.497-l single-cylinder Ricardo Hydra research engine with identical cylinder geometry to the Ford Zetec engine. A production fuel injection system was used along with a modified Ford injector driver for control. The engine was maintained at 1200 rpm for all experiments using a motoring dynamometer, even when engine behavior was very erratic under very high levels of simulated EGR. All feedback controllers except engine speed were disabled to minimize their effect on engine dynamics. Spark timing was set at the maximum brake torque advance for stoichiometric operation and maintained at this advance for the duration of each group of experiments. Cycle-resolved heat release values were analyzed using return maps, bifurcation sequences, symbol sequence statistics, and sequences of repeating heat release values. These tools were used to relate the dynamics exhibited by the engine under high levels of simulated EGR to lean combustion dynamics observed in this engine system.

Figure 10.25 illustrates the effect of simulated EGR with a constant fuel-to-oxidizer ratio on cyclic heat release. Heat release is calculated for 1000 cycles, as EGR level is varied and all other variables are held constant. Data sets were acquired by allowing the engine to stabilize at a predetermined nitrogen

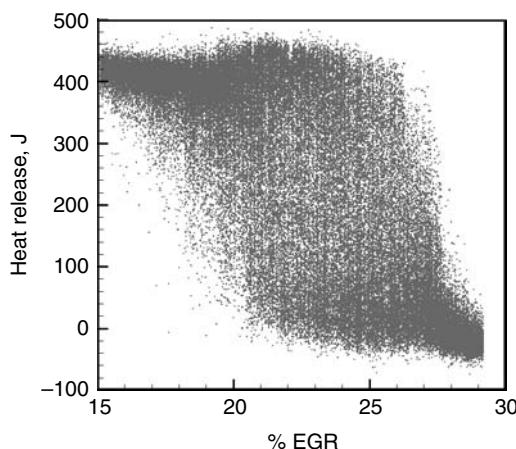


FIGURE 10.25 Bifurcation diagram for EGR at a constant fuel-to-oxidizer ratio (Sutton and Drallmeier 2000)

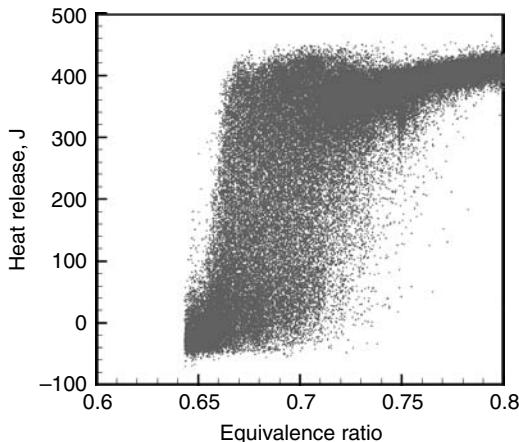


FIGURE 10.26 Bifurcation diagram for A/F ratio (Sutton and Drallmeier 2000).

flow rate, acquiring 1000 cycles of crank-angle resolved pressure data, and then incrementing the nitrogen flow and repeating the process.

The bifurcation plot illustrated in Figure 10.25 displays a number of different combustion modes similar to what was seen in for the lean engine operation in Figure 10.26. The first stage ($EGR < 20\%$) is dominated by variations that appear to be Gaussian in nature. In the second stage ($20\% < EGR < 25\%$) the engine undergoes a transition from small fluctuations in heat release to alternating high-energy and low-energy combustion events that are non-Gaussian in nature. The third stage ($25\% < EGR < 31\%$) is dominated by alternating high-energy and low-energy combustion events (period two). The fourth and final stage ($EGR > 31\%$) is characterized by a rapid decrease in heat release and a return to Gaussian behavior with variations around a value of zero. This bifurcation phenomenon appears to be similar to when the engine was operating lean as shown in Figure 10.26. Therefore, it is envisioned that by applying an NN controller similar to that of a lean operation given in Section 10.3, the cyclic dispersion caused by high levels of EGR dilution can be minimized.

The results of the previous investigations performed by the other researchers (Wagner 1998b, 1999; Sutton and Drallmeier 2000) have very important implications toward controlling cyclic variations in a number of different engine systems. Due to the similar effect of lean combustion (excess air) and high EGR (excess inert gas) on combustion processes (Lunsden 1997), it may be concluded that a control scheme could be modified to operate in a system using high levels of EGR provided such a control scheme could be developed for use under lean fueling conditions. As observed in the literature, operating an

engine with high levels of EGR could significantly reduce the NO_x (as much as 50 to 60% below current strategies) while operating the engine in lean operation would add additional benefits such as fuel economy (reduced CO_2) and in the reduction of other hydrocarbon-based exhaust gases. The NN controller scheme proposed in Section 10.4 is able to reduce the cyclic dispersion without needing the knowledge of the residual gas fraction and other engine parameters.

In fact, the next section will discuss the development of NN controller for engines operating with high EGR levels from Jagannathan et al. (2005). It will be seen that the controller developed for lean engine operation will be modified for engines operating with high EGR levels.

10.4.1 ENGINE DYNAMICS WITH EGR

A mathematical representation of the SI engine was developed by Sutton and Drallmeier (2000) and is given by

$$\begin{aligned} x_1(k+1) = & F(k)[x_1(k) - R \cdot CE(k)x_2(k) + r_{\text{O}_2}(k) + r_{\text{N}_2}(k)] \\ & + x_{1\text{new}}(k) + d_1(k) \end{aligned} \quad (10.69)$$

$$x_2(k+1) = F(k)(1 - CE(k))x_2(k) + x_{2\text{new}}(k) + u(k) + d'_2(k) \quad (10.70)$$

$$x_3(k+1) = F(k)(r_{\text{CO}_2}(k) + r_{\text{H}_2\text{O}}(k)r_{\text{N}_2}(k) + x_3(k) + EGR(k)) \quad (10.71)$$

$$y(k) = x_2(k)CE(k) \quad (10.72)$$

$$\varphi(k) = R \frac{x_2(k)}{x_1(k)} \cdot \left[1 - \gamma \frac{x_3(k) + EGR(k)}{(x_2(k) + x_1(k) + x_3(k) + EGR(k))} \right] \quad (10.73)$$

$$CE(k) = \frac{CE_{\max}}{1 + 100^{-(\varphi(k) - \varphi_m)/(\varphi_u - \varphi_l)}} \quad \varphi_m = \frac{\varphi_u + \varphi_l}{2} \quad (10.74)$$

$$r_{\text{H}_2\text{O}}(k) = \gamma_{\text{H}_2\text{O}}x_2(k)CE(k) \quad (10.75)$$

$$r_{\text{O}_2}(k) = \gamma_{\text{O}_2}x_2(k)CE(k) \quad (10.76)$$

$$r_{\text{N}_2}(k) = \gamma_{\text{N}_2}R \cdot x_2(k)CE(k) \quad (10.77)$$

$$r_{\text{CO}_2}(k) = \gamma_{\text{CO}_2}x_2(k)CE(k) \quad (10.78)$$

where $x_1(k)$, $x_2(k)$, and $x_3(k)$ are the total mass of air, fuel, and inert gases, respectively. The variable $y(k)$ is the heat release at the k th instant, $CE(k)$ the combustion efficiency satisfying $0 < CE_{\min} < CE(k) < CE_{\max}$, where CE_{\max} is the maximum combustion efficiency, $F(k)$ is the residual gas fraction satisfying $0 < F_{\min} < F(k) < F_{\max}$, $x_{1\text{new}}$ and $x_{2\text{new}}$ are the mass of fresh air and

fresh fuel fed per cycle, R is the stoichiometric A/F ratio, approximately 14.6 for gasoline, $u(k)$ is the small change in fuel per cycle, $\varphi(k)$ is the equivalence ratio, φ_m , φ_l , φ_u are system parameters, $r_{\text{H}_2\text{O}}(k)$, $r_{\text{O}_2}(k)$, $r_{\text{N}_2}(k)$, and $r_{\text{CO}_2}(k)$ are the mass of water, oxygen, nitrogen, and carbon dioxide, respectively, γ is a constant and $\gamma_{\text{H}_2\text{O}}$, γ_{O_2} , γ_{N_2} , and γ_{CO_2} are constant parameters associated with water, oxygen, nitrogen, and carbon dioxide, respectively, and $d'_1(k)$ and $d'_2(k)$ are unknown but bounded disturbances. It can be seen that the SI engine with EGR levels has highly nonlinear dynamics with $CE(k)$ and $F(k)$ normally unknown.

Remark 7: In (10.69) through (10.71), states $x_1(k)$ and $x_2(k)$ are not measurable whereas the output $y(k)$ can be measured using a pressure sensor. The control objective is to operate the engine with high EGR levels by taking $y(k)$ as the feedback parameter and without knowing precisely the engine dynamics.

Remark 8: For lean engine operation, the EGR dynamics (10.71) are not considered and there is a slight difference in total air and fuel system dynamics (10.69c) and (10.70).

Substituting (10.72) into both (10.69) and (10.70), we get

$$x_1(k+1) = F(k)[x_1(k) - R \cdot y(k) + r_{\text{O}_2}(k) + r_{\text{N}_2}(k)] + x_{1\text{new}}(k) + d_1(k) \quad (10.79)$$

$$x_2(k+1) = F(k)(x_2(k) - y(k)) + x_{2\text{new}}(k) + u(k) + d'_2(k) \quad (10.80)$$

In real engine operation, the fresh air, $x_{1\text{new}}$, fresh fuel, $x_{2\text{new}}$, and residual gas fraction, $F(k)$, can all be viewed as nominal values plus some small and bounded perturbations.

$$x_{1\text{new}}(k) = x_{1\text{new}0} + \Delta x_{1\text{new}}(k) \quad (10.81)$$

$$x_{2\text{new}}(k) = x_{2\text{new}0} + \Delta x_{2\text{new}}(k) \quad (10.82)$$

and

$$F(k) = F_0(k) + \Delta F(k) \quad (10.83)$$

where $x_{1\text{new}0}$, $x_{2\text{new}0}$, and F_0 are the known nominal fresh air, fuel, and residual gas fraction values. The bounds on the unknown bounded perturbations

$\Delta x_{1\text{new}0}$, $\Delta x_{2\text{new}0}$, and ΔF_0 are given by

$$0 \leq |\Delta x_{1\text{new}}(k)| \leq \Delta x_{1\text{new}M} \quad (10.84)$$

$$0 \leq |\Delta x_{2\text{new}}(k)| \leq \Delta x_{2\text{new}M} \quad (10.85)$$

$$0 \leq |\Delta F(k)| \leq \Delta F_M \quad (10.86)$$

Substituting these values into the system model we can get

$$\begin{aligned} x_1(k+1) = & (F_0(k) + \Delta F(k))[x_1(k) - R \cdot CE(k)x_2(k) + r_{O_2}(k) + r_{N_2}(k)] \\ & + x_{1\text{new}0} + \Delta x_{1\text{new}}(k) + d_1(k) \end{aligned} \quad (10.87)$$

$$\begin{aligned} x_2(k+1) = & (F_0(k) + \Delta F(k))(1 - CE(k))x_2(k) + x_{2\text{new}0} + \Delta x_{2\text{new}}(k) \\ & + u(k) + d'_2(k) \end{aligned} \quad (10.88)$$

10.4.2 NN OBSERVER DESIGN

First a NN is used to predict the value of the heat release for the subsequent cycle which will be used by the observer to predict the states of the system. The heat release for the next burn cycle is given by

$$y(k+1) = x(k+1)CE(k+1) \quad (10.89)$$

From (10.89), $y(k+1)$ can be approximated by using a one-layer NN as

$$y(k+1) = w_1^T \phi_1(v_1^T z_1(k)) + \varepsilon_1(z_1(k)) \quad (10.90)$$

where the NN input is taken as $z_1(k) = [x_1(k), x_2(k), y(k), u(k)]^T \in R^4$, the matrices $w_1 \in R^{4 \times n_1}$ and $v_1 \in R^{4 \times n_1}$ represent the output- and hidden-layer weights, $\phi_1(\cdot)$ represents the hidden-layer activation function, n_1 denotes the number of the nodes in the hidden layer, and $\varepsilon_1(z_1(k)) \in R$ is the functional approximation error. It has been demonstrated that, if the hidden-layer weight, v_1 is chosen initially at random and held constant and the number of hidden-layer nodes is sufficiently large, the approximation error $\varepsilon_1(z_1(k))$ can be made arbitrarily small over the compact set since the activation function forms a basis (Igelnik and Pao 1995).

For simplicity, we define

$$\phi_1(z_1(k)) = \phi_1(v_1^T z_1(k)) \quad (10.91)$$

and

$$\varepsilon_1(k) = \varepsilon_1(z_1(k)) \quad (10.92)$$

Given (10.91) and (10.92), (10.90) is rewritten as

$$y(k+1) = w_1^T \phi_1(z_1(k)) + \varepsilon_1(k) \quad (10.93)$$

Since states $x_1(k)$ and $x_2(k)$ are not measurable, $z_1(k)$ is not available. Using the estimated values $\hat{x}_1(k)$, $\hat{x}_2(k)$, and $\hat{y}(k)$ instead of the actual values for $x_1(k)$ and $x_2(k)$ and $y(k)$, respectively, the proposed heat release observer can be given as

$$\hat{y}(k+1) = \hat{w}_1^T \phi_1(v_1^T \hat{z}_1(k)) + l_1 \tilde{y}(k) = \tilde{w}_1^T(k) \phi_1(\hat{z}_1(k)) + l_1 \tilde{y}(k) \quad (10.94)$$

where $\hat{y}(k+1)$ is the predicted heat release, $\hat{w}_1(k) \in R^{n_i}$ is the actual output-layer weights, the NN input $\hat{z}_1(k) = [\hat{x}_1(k), \hat{x}_2(k), \hat{y}(k), u(k)]^T \in R^4$, $l \in R$ is the observer gain, $\tilde{y}(k)$ is the heat release estimation error defined as

$$\tilde{y}(k) = \hat{y}(k) - y(k) \quad (10.95)$$

where $\phi_1(\hat{z}_1(k))$ represents $\phi_1(v_1^T \hat{z}_1(k))$ for the purpose of simplicity.

Since the mass of water, oxygen, nitrogen, and carbon dioxide, respectively $r_{H_2O}(k)$, $r_{O_2}(k)$, $r_{N_2}(k)$, and $r_{CO_2}(k)$ are unknown, the proposed observer structure is same as that of the NN observer used during lean engine operation and it is given by

$$\hat{x}_1(k+1) = x_{1\text{new}0}(k) + F_o \hat{x}_1(k) - R \cdot F_o \cdot \hat{y}(k) + l_2 \tilde{y}(k) \quad (10.96)$$

and

$$\hat{x}_2(k+1) = F_o(\hat{x}_2(k) - \hat{y}(k)) + (x_{2\text{new}0}(k) + u(k)) + l_3 \tilde{y}(k) \quad (10.97)$$

where $l_2 \in R$ and $l_3 \in R$ are observer gains. Here, it is assumed that the initial value of $u(0)$ is bounded. Equation 10.94, Equation 10.96, and Equation 10.97 represent the dynamics of the observer to estimate the states of $x_1(k)$ and $x_2(k)$.

Define the state estimation errors as

$$\tilde{x}_i(k) = \hat{x}_i(k) - x_i(k) \quad i = 1, 2 \quad (10.98)$$

Combining (10.71) through (10.86), we obtain the dynamics in estimation error as

$$\begin{aligned}\tilde{x}_1(k+1) &= F_o \tilde{x}_1(k) + (l_2 - R \cdot F_o) \tilde{y}(k) - \Delta x_{1\text{new}}(k) - \Delta F(k)x_1(k) \\ &\quad + R\Delta F(k)y(k) - d_1(k)\end{aligned}\quad (10.99)$$

$$\begin{aligned}\tilde{x}_2(k+1) &= F_o \tilde{x}_2(k) + (l_3 - R \cdot F_o) \tilde{y}(k) - \Delta F(k)(x_2(k) - y(k)) \\ &\quad - \Delta x_{2\text{new}}(k) - d_2(k)\end{aligned}\quad (10.100)$$

and

$$\begin{aligned}\tilde{y}(k+1) &= \hat{w}_1^T(k)\phi_1(\hat{z}_1(k)) + l_1\tilde{y}(k) - w_1^T\phi_1(z_1(k)) - \varepsilon_1(k) \\ &= (\hat{w}_1(k) - w_1)^T\phi_1(\hat{z}_1(k)) + w_1^T(\phi_1(\hat{z}_1(k)) - \phi_1(z_1(k))) - \varepsilon_1(k) \\ &= \tilde{w}_1^T(k)\phi_1(\hat{z}_1(k)) + \tilde{w}_1^T(k)\phi_1(\tilde{z}_1(k)) - \varepsilon_1(k) \\ &= \zeta_1(k) + w_1^T\phi_1(\tilde{z}_1(k)) - \varepsilon_1(k)\end{aligned}\quad (10.101)$$

where

$$\tilde{w}_1(k-1) = \hat{w}_1(k) - w_1 \quad (10.102)$$

$$\zeta_1(k) = \tilde{w}_1^T(k)\phi_1(z_1(k)) \quad (10.103)$$

and for the purpose of simplicity, $(\phi_1(\hat{z}_1(k)) - \phi_1(z_1(k)))$ is written as $\phi_1(\tilde{z}_1(k))$.

10.4.3 ADAPTIVE OUTPUT FEEDBACK EGR CONTROLLER DESIGN

The control objective of maintaining the heat release close to its target value of y_d is achieved by minimizing the variations in the equivalence ratio. Given y_d and the engine dynamics along with EGR system (10.69) through (10.73), we could obtain the desired operating point of total mass of air and fuel in the cylinder, x_{1d} and x_{2d} , respectively. By driving both states $x_1(k)$ and $x_2(k)$ to approach their respective targets x_{1d} and x_{2d} , actual heat release $y(k)$ will be shown to approach y_d . By developing a separate adaptive critic NN controller (He and Jagannathan 2005) to maintain the actual EGR at a desired value, we can show that the inert gas will evolve into a stable value since Equation 10.71 is a stable system with $F(k)$ being less than 1 and the weights of the gases kept constant with minor variations. For this system, one can design a simple robust NN feedback linearization controller separately to maintain certain EGR level. This will allow the required amount of EGR to be fed into the intake system. The controller for the EGR system is not detailed here but one can design a controller similar to the

one in Chapter 3 where a NN feedback linearizing controller is given. With the estimated states $\hat{x}_1(k)$ and $\hat{x}_2(k)$, the controller design follows the backstepping technique. The details are given in the following sections.

10.4.3.1 Error Dynamics

Step 1: Virtual controller design

Define the error between actual and desired air as

$$e_1(k) = x_1(k) - x_{1d} \quad (10.104)$$

which can be rewritten as

$$\begin{aligned} e_1(k+1) &= x_1(k+1) - x_{1d} \\ &= F(k)[x_1(k) - R \cdot CE(k)x_2(k) + r_{O_2}(k) \\ &\quad + r_{N_2}(k)] - x_{1d} + x_{1\text{new}}(k) + d_1(k) \end{aligned} \quad (10.105)$$

For simplicity let us denote

$$f_1(k) = F(k)[x_1(k) + r_{O_2}(k) + r_{N_2}(k)] + x_{1\text{new}}(k) \quad (10.106)$$

and

$$g_1(k) = R \cdot F(k)CE(k) \quad (10.107)$$

Then the system error equation can be expressed as

$$e_1(k+1) = f_1(k) - g_1(k)x_2(k) + d_1(k) \quad (10.108)$$

By viewing $x_2(k)$ as a virtual control input, a desired feedback control signal can be designed as

$$x_{2d}(k) = \frac{f_1(k)}{g_1(k)} \quad (10.109)$$

The term $x_{2d}(k)$ can be approximated by the first NN as

$$x_{2d}(k) = w_2^T \phi_2(v_2^T x(k)) + \varepsilon_2(x(k)) = w_2^T \phi_2(x(k)) + \varepsilon_2(x(k)) \quad (10.110)$$

where the input in the state $x(k) = [x_1(k), x_2(k)]^T$, $w_2 \in R^{n_2}$, and $v_2 \in R^{2 \times n_1}$ denote the constant ideal output- and hidden-layer weights, n_2 is the number

of nodes in the hidden-layer, the hidden-layer activation function $\phi_2(v_2^T x(k))$ is simplified as $\phi_2(x(k))$, and $\varepsilon_2(x(k))$ is the approximation error. Since both $x_1(k)$ and $x_2(k)$ are unavailable, the estimated state $\hat{x}(k)$ is selected as the NN input.

Consequently, the virtual control input is taken as

$$x_{2d}(k) = \hat{w}_2^T(k) \phi_2(v_2^T \hat{x}(k)) = \hat{w}_2^T(k) \phi_2(\hat{x}(k)) \quad (10.111)$$

where $\hat{w}_2(k) \in R^{n_2}$ is the actual weight matrix for the first action NN. Define the weight estimation error by

$$\tilde{w}_2(k) = \hat{w}_2(k) - w_2 \quad (10.112)$$

Define the error between $x_2(k)$ and $\hat{x}_{2d}(k)$ as

$$e_2(k) = x_2(k) - \hat{x}_{2d}(k) \quad (10.113)$$

Equation 10.104 can be expressed using (10.113) for $x_2(k)$ as

$$e_1(k+1) = f_1(k) - g_1(k)(e_2(k) + \hat{x}_{2d}(k)) + d_1(k) \quad (10.114)$$

or equivalently,

$$\begin{aligned} e_1(k+1) &= f_1(k) - g_1(k)(e_2(k) + x_{2d}(k) - x_{2d}(k) + \hat{x}_{2d}(k)) + d_1(k) \\ &= f_1(k) - g_1(k)(e_2(k) + x_{2d}(k) - x_{2d}(k) + \hat{x}_{2d}(k)) + d_1(k) \\ &= -g_1(k)(e_2(k) - x_{2d}(k) + \hat{x}_{2d}(k)) + d_1(k) \\ &= -g_1(k)(e_2(k) + \hat{w}_2^T(k) \phi_2(\hat{x}(k)) - w_2^T \phi_2(x(k)) - \varepsilon_2(x(k))) + d_1(k) \end{aligned} \quad (10.115)$$

Similar to (10.101), (10.115) can be further expressed as

$$e_1(k+1) = -g_1(k)(e_2(k) - \xi_2(k) + w_2^T \phi_2(\tilde{x}(k)) - \varepsilon_2(x(k))) + d_1(k) \quad (10.116)$$

where,

$$\xi_2(k) = \tilde{w}_2^T(k) \phi_2(\hat{x}(k)) \quad (10.117)$$

and

$$w_2^T \phi_2(\tilde{x}(k)) = w_2^T (\phi_2(\hat{x}(k)) - \phi_2(x(k))) \quad (10.118)$$

Step 2: Design of the control input $u(k)$

Rewriting the error $e_2(k)$ from (10.109) as

$$\begin{aligned} e_2(k+1) &= x_2(k+1) - \tilde{x}_{2d}(k+1) = (1 - CE(k))F(k)x_2(k) + (MF(k) \\ &\quad + u(k)) - \hat{x}_{2d}(k+1) + d_2(k) \end{aligned} \quad (10.119)$$

For simplicity, let us denote

$$x_2(k+1) = F(k)(1 - CE(k))x_2(k) + x_{2\text{new}}(k) \quad (10.120)$$

Equation 10.119 can be written as

$$e_2(k+1) = f_2(k) + u(k) - \hat{x}_{2d}(k+1) + d_2(k) \quad (10.121)$$

where $\hat{x}_{2d}(k+1)$ is the future value of $\hat{x}_{2d}(k)$. Here, $\hat{x}_{2d}(k+1)$ is not available in the current time step. However, from (10.109) and (10.111), it can be clear that $\hat{x}_{2d}(k+1)$ is a smooth nonlinear function of the state $x(k)$ and virtual control input $\hat{x}_{2d}(k)$. Using the discussion detailed in Section 10.3, Chapter 4, and in other chapters, $\hat{x}_{2d}(k+1)$ can be approximated by using another NN. For other methods refer to Lewis et al. (2002).

Now select the desired control input by using a second NN for the controller design as

$$\begin{aligned} u_d(k) &= (-f_2(k) + \hat{x}_{2d}(k+1)) = w_3^T \phi_3(v_3^T z_3(k)) + \varepsilon_3(z_3(k)) \\ &= w_3^T \phi_3(z_3(k)) + \varepsilon_3(z_3(k)) \end{aligned} \quad (10.122)$$

where $w_3 \in R^{n_3}$ and $v_3 \in R^{3 \times n_3}$ denote the constant ideal output- and hidden-layer weights, n_3 is the hidden-layer nodes number, $\phi_3(v_3^T z_3(k))$ the hidden-layer activation function is denoted as $\phi_3(z_3(k))$, $\varepsilon_3(z_3(k))$ is the approximation error, $z_3(k) \in R^3$ is the NN input given by (10.123). Considering the fact that both $x_1(k)$ and $x_2(k)$ cannot be measured, $z_3(k)$ is substituted with $\hat{z}_3(k) \in R^3$ where

$$z_3(k) = [x(k), \hat{x}_{2d}(k)]^T \in R^3 \quad (10.123)$$

and

$$\hat{z}_3(k) = [\hat{x}(k), \hat{x}_{2d}(k)]^T \in R^3 \quad (10.124)$$

Define

$$\hat{e}_1(k) = \hat{x}_1(k) - x_{1d} \quad (10.125)$$

and

$$\hat{e}_2(k) = \hat{x}_2(k) - x_{2d} \quad (10.126)$$

The actual control input is now selected as

$$u(k) = \hat{w}_3^T(k)\phi_3(v_c^T\hat{z}_3(k)) + l_4\hat{e}_2(k) = \hat{w}_3^T(k)\phi_3(\hat{z}_3(k)) + l_4\hat{e}_2(k) \quad (10.127)$$

where $\hat{w}_3^T \in R^{n_3}$ is the actual output-layer weight vector, $l_4 \in R$ is the controller gain selected to stabilize the system. Similar to the derivation of (10.111), combining (10.121), (10.123) with (10.127) yields

$$e_2(k+1) = l_4\hat{e}_2(k) + \xi_3(k) + w_3^T\phi_3(\tilde{z}(k)) - \varepsilon_3(z_3(k)) + d_2(k) \quad (10.128)$$

where

$$\tilde{w}_3(k) = \hat{w}_3(k) - w_3 \quad (10.129)$$

$$\xi_3(k) = \tilde{w}_3^T(k)\phi_3(\hat{z}_3(k)) \quad (10.130)$$

and

$$w_3^T\phi_3(\tilde{z}(k)) = w_3^T(\phi_3(\hat{z}_3(k)) - \phi_3(z_3(k))) \quad (10.131)$$

Equation 10.116 and Equation 10.128 represent the closed-loop error dynamics. It is required to show that the estimation errors (10.95) and (10.98), the system errors (10.116) and (10.128) and the NN weight matrices $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded.

10.4.3.2 Weight Updates for Guaranteed Performance

Assumption 10.4.1 (Bounded Ideal Weights): Let w_1 , w_2 , and w_3 be the unknown output layer target weights for the NN observer and NN controller respectively and assume that they are bounded above so that

$$\|w_1\| \leq \|w_{1m}\| \quad \|w_2\| \leq \|w_{2m}\| \quad \text{and} \quad \|w_3\| \leq \|w_{3m}\| \quad (10.132)$$

where $w_{1m} \in R^+$, $w_{2m} \in R^+$, and $w_{3m} \in R^+$ represent the bounds on the unknown target weights where the Frobenius norm is used.

Fact 10.4.1: The activation functions are bounded above by known positive values so that

$$\|\phi_i(\cdot)\| \leq \phi_{im} \quad i = 1, 2, 3 \quad (10.133)$$

where ϕ_{im} , $i = 1, 2, 3$ are the upper bounds.

Assumption 10.4.2 (Bounded NN Approximation Error): The NN approximation errors $\varepsilon_1(z_1(k))$, $\varepsilon_2(x(k))$, and $\varepsilon_3(z_3(k))$ are bounded over the compact set by ε_{1m} , ε_{2m} , and ε_{3m} , respectively.

Theorem 10.4.1 (EGR Controller): Consider the system given in (10.69) through (10.71) and let the Assumptions 10.4.1 and 10.4.2 hold. Let the unknown disturbances be bounded by $|d_1(k)| \leq d_{1m}$ and $|d_2(k)| \leq d_{2m}$, respectively. Let the observer NN weight tuning be given by

$$\hat{w}_1(k+1) = \hat{w}_1(k) - \alpha_1 \phi_1(\hat{z}_1(k)) (\hat{w}_1^T(k) \phi_1(\hat{z}_1(k) + l_5 \tilde{y}(k))) \quad (10.134)$$

with the virtual control NN weight tuning be provided by

$$\hat{w}_2(k+1) = \hat{w}_2(k) - \alpha_2 \phi_2(\hat{x}(k)) (\hat{w}_2^T(k) \phi_2(\hat{x}(k) + l_6 \hat{e}_1(k))) \quad (10.135)$$

and the control NN weight tuning be provided by

$$\hat{w}_3(k+1) = \hat{w}_3(k) - \alpha_3 \phi_3(\hat{z}_3(k)) (\hat{w}_3^T(k) \phi_3(\hat{z}_3(k) + l_7 \hat{e}_2(k))) \quad (10.136)$$

where $\alpha_1 \in R$, $\alpha_2 \in R$, $\alpha_3 \in R$ and $l_5 \in R$, $l_6 \in R$, and $l_7 \in R$ are design parameters. Let the system observer be given by (10.94) to (10.96), virtual and actual control inputs be defined as (10.111) and (10.127), respectively. The estimation errors (10.99) through (10.93), the tracking errors (10.116) and (10.128), and the NN weight estimates $\hat{w}_1(k)$, $\hat{w}_2(k)$ and $\hat{w}_3(k)$ are UUB provided the design parameters are selected as:

$$(a) \quad 0 < \alpha_i \|\phi_i(k)\|^2 < 1 \quad i = 1, 2, 3 \quad (10.137)$$

$$(b) \quad l_3^2 < 1 - \frac{(l_1 - R \cdot F_0)^2}{6R^2 \cdot \Delta F_m^2} - \frac{(l_2 - F_0)^2}{6\Delta F_m^2} - 4l_5^2 \quad (10.138)$$

$$(c) \quad l_6^2 < \min \left(\frac{(1 - F_0^2)}{18R^2 \cdot \Delta F_m^2}, \frac{1}{18R^2} \right) \quad (10.139)$$

$$(d) \quad l_4^2 + 6l_7^2 < \min \left(\frac{(1 - F_0^2)}{6\Delta F_m^2}, \frac{1}{3} \right) \quad (10.140)$$

Proof: See lean engine controller proof in Appendix 10.A

Remark 9: For general nonlinear discrete-time systems, the design parameters can be selected using a priori bounds. Given specific values of R , F_0 , and ΔF_m , the design parameters can be derived as l_i , $i = 1$ to 7. For instance, given $R = 14.6$, $F_0 = 0.14$, and $\Delta F_m = 0.02$, we can select $l_1 = 1.99$, $l_2 = 0.13$, $l_3 = 0.4$, $l_4 = 0.14$, $l_5 = 0.25$, $l_6 = 0.016$, and $l_7 = 0.1667$ to satisfy (10.137) through (10.140).

10.4.4 NUMERICAL SIMULATION

The parameters are selected as the following: 20,000 cycles are considered with $R = 15.13$ (iso-octane), residual gas fraction $F = 0.09$, mass of nominal new air = 0.52485, mass of nominal new fuel = 0.02428, the standard deviation of mass of new fuel is 0.007, cylinder volume in moles = 0.021, molecular weight of fuel = 114, molecular weight of air = 28.84, $\phi_u = 0.665$, $\phi_l = 0.645$, maximum combustion efficiency = 1, and the gains of backstepping controller presented in Section 10.4.3 are selected at 0.1 and placed diagonally. An equivalence ratio of one was maintained with variation of 1%. Molecular weight of EGR = 30.4, hydrogen carbon ratio is 1.87. All initial values of air, fuel, and inert gases were chosen to be 0.

The selection of other parameters is given in the remark from the previous page. The activation functions used are the hyperbolic tangent sigmoid functions. The simulation was run for 1000 cycles of engine operation by varying the EGR value from 19 to 29%. Figure 10.27 shows the heat release variation per cycle via a return map without and with control whereas Figure 10.28 presents the backstepping lean engine NN controller performance in terms of the variations in heat release when the control action is turned on at 1001th cycle. It is observed that the engine exhibits minimal dispersion with high EGR levels even with perturbation on the residual gas fraction being unknown.

Figure 10.29 illustrates that the variations in combustion efficiency is minimized with control. The associated total air (new plus residual) and fuel plots with and without control for this EGR level is shown in Figure 10.30 and Figure 10.31, respectively. The error in estimation of the total air and fuel is depicted in Figure 10.32 and Figure 10.33. These indicate that the cyclic dispersion in heat release is minimized by the control and observer. As a result, the variations in residual fuel and air are minimized.

The dispersion without control is significant and can cause a misfire, whereas with control it can be seen that the dispersion is controlled within a tight bound. It is necessary to apply the coefficient of variation metric to determine the appropriate amount of EGR level needed for the engine in order to meet emission standards.

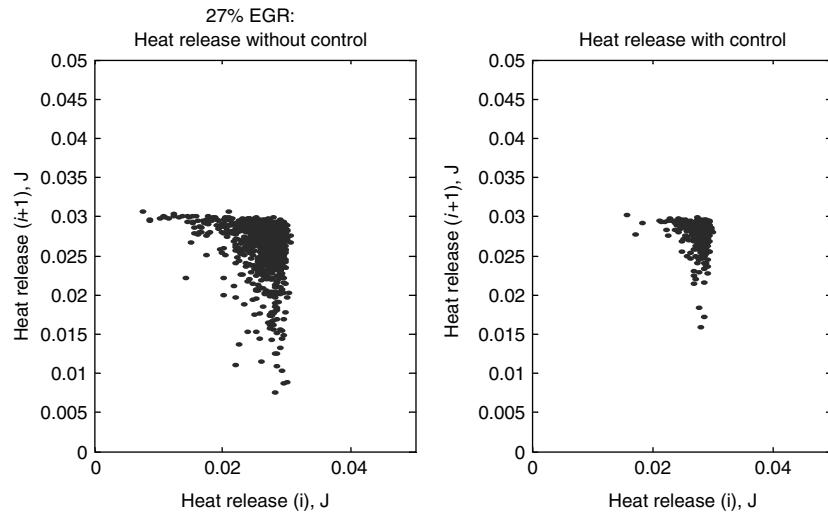


FIGURE 10.27 Heat release variations without and with control at 27% EGR.

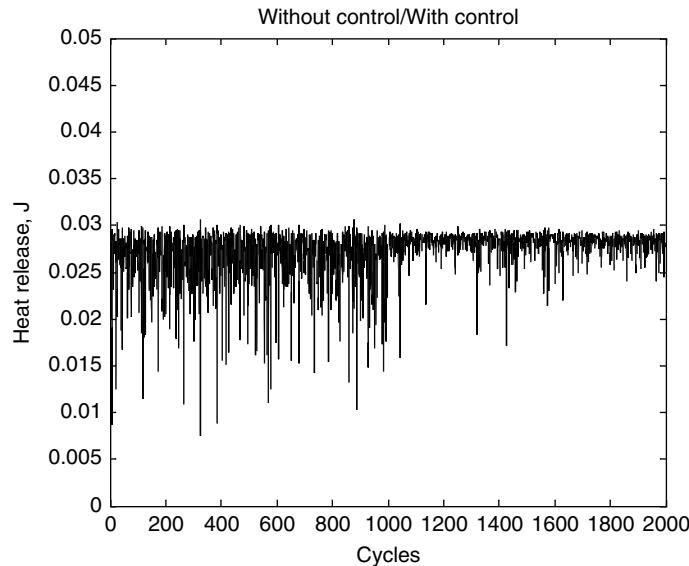


FIGURE 10.28 Heat release variations when control action is initiated at 1001th cycle with 27% EGR.

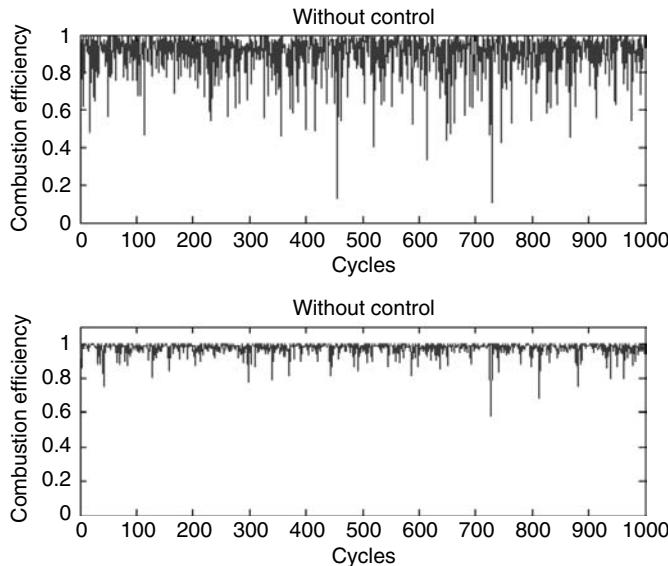


FIGURE 10.29 Combustion efficiency with and without control.

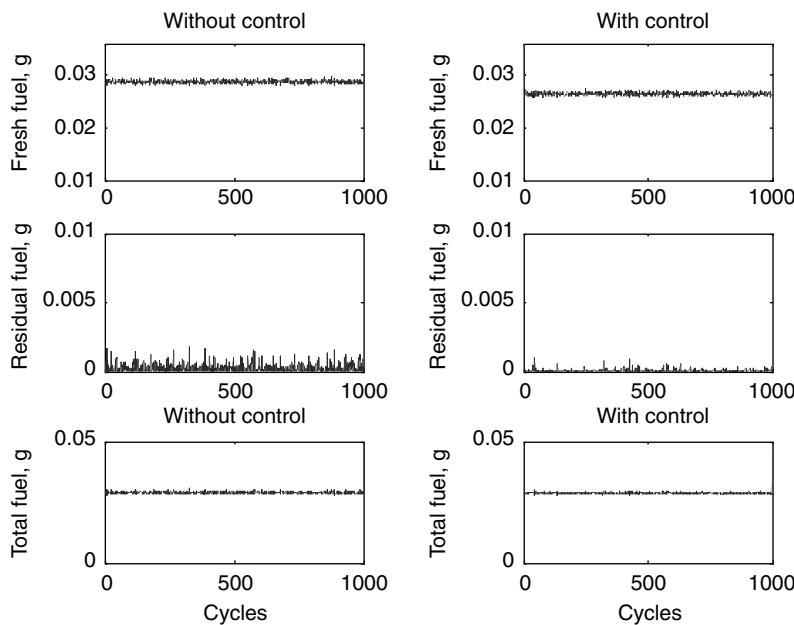


FIGURE 10.30 Total new and residual fuel without and with control action.

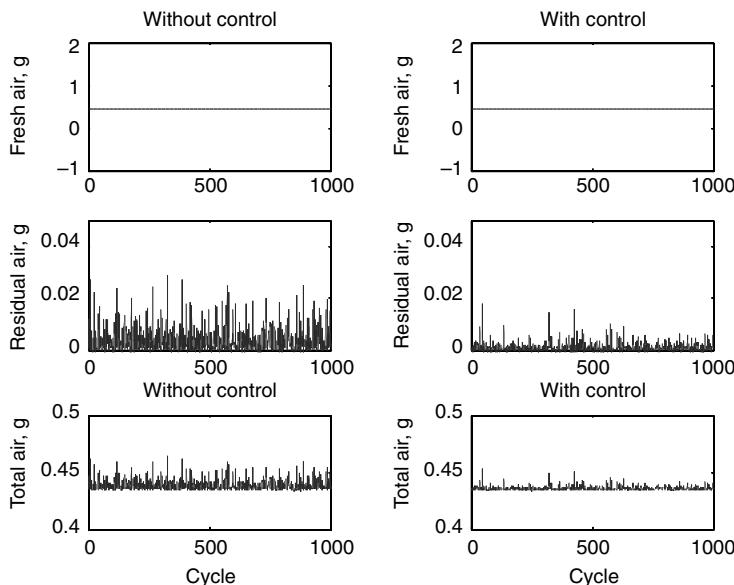


FIGURE 10.31 Total new and residual air without and with control action.

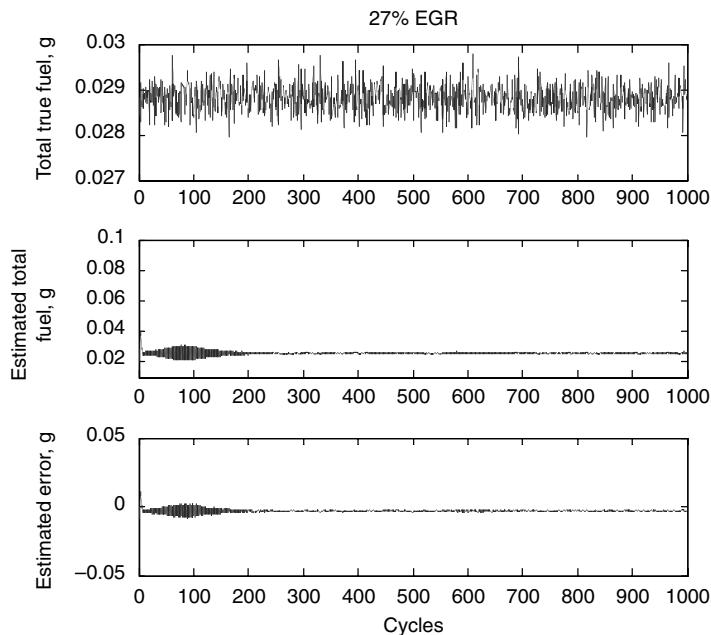


FIGURE 10.32 Estimated error in total fuel.

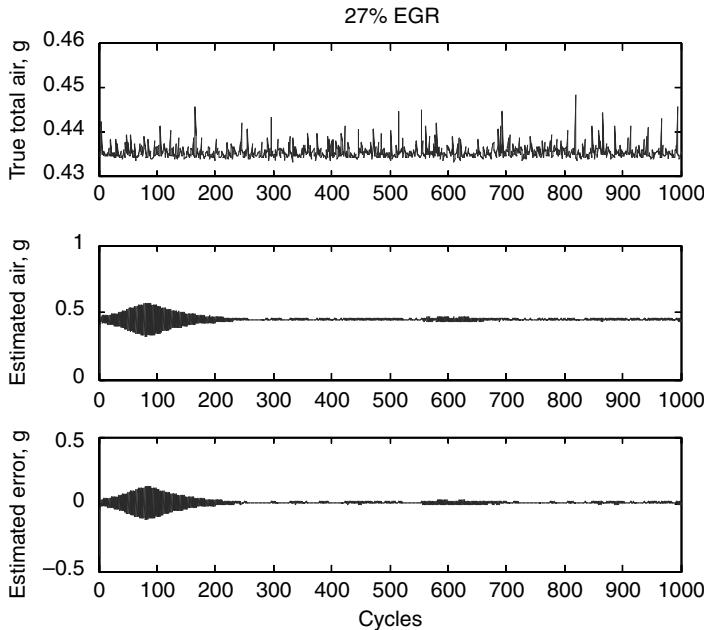


FIGURE 10.33 Estimated error in total air.

10.5 CONCLUSIONS

This chapter presents adaptive NN controllers for lean engine operation and with high EGR levels. The hardware and software issues of the controller development are discussed in detail and the steps employed for designing the controller are detailed. Lyapunov stability analysis is used to develop the adaptive NN controllers. Uniform ultimately boundedness of the closed-loop system is demonstrated. Experimental results of the lean engine controller demonstrate that the cyclic dispersion has been reduced significantly with control action compared to without control. This allows the controller to be operated at extreme lean conditions resulting in a significant reduction in NO_x and fuel intake compared to the stoichiometric conditions. However, the controller reduces the cyclic dispersion by making the fuel slightly richer. This has shifted the operating point slightly. Operating an engine by using nominal fuel indicates that the cyclic dispersion produced at this point is on par with the controlled scenario. This translated into minimal reduction in other emission products; an issue that warrants further investigation. Future work involves an appropriate selection of control parameter, for instance spark timing and feedback variable for minimizing cyclic dispersion.

REFERENCES

- Daily, J.W., Cycle-to-cycle variations: a chaotic process, *Combust. Sci. Technol.*, 57, 149–162, 1988.
- Davis, Jr., Daw, C.S., Feldkamp, L.A., Hoard, J.W., Yuan, F., and Conolly, T., Method of controlling cyclic variation engine combustion. U.S. Patent 5,921,221, 1999.
- Daw, C.S., Finney, C.E.A., Green, J.B., Kennel, M.B., and Thomas, J.F., A simple model for cyclic variations in a spark-ignition engine, SAE Technical Paper Series, 962086, 1996.
- Daw, C.S., Finney, C.E.A., Kennel, M.B., and Connolly, F.T., Observing and modeling nonlinear dynamics in an internal combustion engine, *Phys. Rev. E*, 57, 2811–2819, 1998.
- Dudek, K.P. and Sain, M.K., A control-oriented model for cylinder pressure in internal combustion engines, *IEEE Trans. Autom. Contr.*, 34, 386–397, 1989.
- He, P. and Jagannathan, S., Reinforcement-based neuro-output feedback control of discrete-time systems with input constraints, *IEEE Trans. Syst., Man Cybern. — Part B*, 35, 150–154, 2005.
- He, P. and Jagannathan, S., Neuroemission controller for reducing cyclic dispersion in lean combustion spark ignition engines, *Automatica*, 41, 1133–1142, July 2005.
- He, P., Vance, J., and Jagannathan, S., Heat release based neuro output feedback control of minimizing cyclic dispersion in spark ignition engines, *preprint*, 2005.
- Heywood, J.B., *Internal Combustion Engine Fundamentals*, McGraw-Hill, New York, 1988.
- Igelnik, B. and Pao, Y.H., Stochastic choice of basis functions in adaptive function approximation and the functional-link net, *IEEE Trans. Neural Netw.*, 6, 1320–1323, 1995.
- Inoue, T., Matsushita, S., Nakanishi, K., and Okano, H., Toyota lean combustion system — the third generation system. SAE Technical Paper Series, 930873, 1993.
- Itoyama, H., Iwano, H., Osamura, K., and Oota, K., Air-fuel ratio control system for internal combustion engine, U.S. Patent 0,100,454, 2002.
- Jagannathan, S., Control of a class of nonlinear systems using multilayered neural networks, *IEEE Trans. Neural Netw.*, 12, 1113–1120, 2001.
- Jagannathan, S. and Lewis, F.L., Discrete-time neural net controller for a class of nonlinear dynamical systems, *IEEE Trans. Autom. Contr.*, 41, 1693–1699, 1996.
- Jagannathan, S., He, P., Singh, A., and Drallmeier, J., Neural network-based output feedback control of a class of nonlinear discrete-time systems with application to engines with high EGR levels, *Proceedings of the Yale Workshop on Adaptive Control*, New Haven, 2005.
- Kantor, J.C., A dynamical instability of spark ignition engines, *Science*, 224, 1233–1235, 1984.
- Krstic, M., Kanellakopoulos, I., and Kokotovic, P.V., *Nonlinear and Adaptive Control Design*, John Wiley & Sons, Inc., New York, 1995.

- Lewis, F.L., Campos, J., and Selmic, R., *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society for Industrial and Applied Mathematics, Philadelphia, 2002.
- Lunsden, G., Eddleston, D., and Sykes, R., Comparing lean burn with EGR, SAE Paper No. 970505, 1997.
- Martin, J.K., Plee, S.L., and Remboski, D.J., Burn modes and prior-cycle effects on cyclic variations in lean-burn spark-ignition combustion, SAE Paper No. 880201, 1988.
- Ozidor, N., Dulger, M., and Sher, E., Cyclic variability in spark ignition engines: a literature survey, SAE Paper No. 940987, 1994.
- Pingan, He. and Jagannathan, S., Lean combustion stability of spark ignition engines using backstepping scheme, *Proc. IEEE Conf. Contr. Appl.*, 1, 167–172, 2003.
- Sutton, R.W. and Drallmeier, J.A., Development of nonlinear cyclic dispersion in spark ignition engines under the influence of high levels of EGR, *Proceedings of the Central States Section of the Combustion Institute*, pp. 175–180, 2000.
- Van Nieuwstadt, M.J., Kolmanovsky, I.V., Moraal, P.E., Stefanopoulos, A., and Jankovic, M., EGR-VDT control schemes: experimental comparison for a high-speed diesel engine, *IEEE Contr. Syst. Mag.*, 64–79, 2000.
- Vance, J., Design and implementation of neural network output controller for SI engines, M.S. Thesis, Department of ECE, University of Missouri-Rolla, Nov. 2005.
- Vance, J., He, P., Jagannathan, S., and Drallmeier, J., Design and implementation of an embedded lean engine controller for minimizing cyclic dispersion in heat release, *preprint*, 2005.
- Wagner, R.M., *The impact of fuel spray behavior on combustion stability in a spark ignition engine*, M.S. Thesis, University of Missouri-Rolla, 1995.
- Wagner, R.M., *Identification and characterization of complex dynamic structure in spark ignition engines*, Ph.D. Dissertation, University of Missouri-Rolla, Department of Mechanical Engineering, 1999.
- Wagner, R.M., Drallmeier, J.A., and Daw, C.S., Nonlinear cycle dynamics in lean spark ignition combustion, *Proceedings of 27th Symposium (International) of Combustion*, 1998a.
- Wagner, R.M., Drallmeier, J.A., and Daw, C.S., Origins of cyclic dispersion patterns in spark ignition engines, *Proceedings of the Central State Technical Meeting of the Combustion Institute*, 1998b.
- Yeh, P.C. and Kokotovic, P.V., Adaptive output feedback design for a class of nonlinear discrete-time systems, *IEEE Trans. Autom. Contr.*, 40, 1663–1668, 1995.

PROBLEMS

SECTION 10.7

10.7-1: (*Adaptive critic lean engine NN controller*). Design an adaptive critic NN controller for the system described by (10.1) through (10.3). Prove the

stability of the controller and demonstrate the performance of the controller by using coefficient of variation metric.

10.7-2: (*Adaptive critic NN EGR controller*). Design an adaptive critic NN controller for the system described by (10.69) through (10.77). Prove the stability of the controller and demonstrate the performance of the controller by using coefficient of variation metric.

APPENDIX 10.A

Proof of Theorem 10.2.1: Define the Lyapunov function

$$\begin{aligned} J(k) = & \sum_{i=1}^3 \frac{\gamma_i}{\alpha_i} \tilde{w}_i^T(k) \tilde{w}_i(k) + \frac{\gamma_4}{\alpha_4} \tilde{x}_1^2(k) + \frac{\gamma_5}{5} \tilde{x}_2^2(k) \\ & + \frac{\gamma_6}{3} \tilde{y}^2(k) + \frac{\gamma_7}{3} e_1^2(k) + \frac{\gamma_8}{4} e_2^2(k) \end{aligned} \quad (10.A.1)$$

where $0 < \gamma_i$, $i = 1, 5, 8$ are auxiliary constants; the NN weights estimation errors \tilde{w}_1 , \tilde{w}_2 , and \tilde{w}_3 are defined in (10.30), (10.40), and (10.57), respectively; the observation errors $\tilde{x}_1(k)$, $\tilde{x}_2(k)$, and $\tilde{y}(k)$ are defined in (10.26) and (10.23), respectively; the system errors $e_1(k)$ and $e_2(k)$ are defined in (10.32) and (10.41), respectively; and α_i , $i = 1, 2, 3$ are NN adaptation gains. The Lyapunov function (10.A.1) consisting of the system errors, observation errors, and the weights estimation errors obviates the need for CE-condition.

The first difference of the Lyapunov function is given by

$$\Delta J(k) = \sum_{i=1}^8 \Delta J_i(k) \quad (10.A.2)$$

The first item of $\Delta J_1(k)$ is obtained using (10.62) as

$$\begin{aligned} \Delta J_1(k) = & \frac{\gamma_1}{\alpha_1} [\tilde{w}_1^T(k+1) \tilde{w}_1(k+1) - \tilde{w}_1^T(k) \tilde{w}_1(k)] \\ \leq & -\gamma_1 (1 - \alpha_1 \|\phi_1(\cdot)\|^2) \times (\hat{w}_1^T(k) \phi_1(\cdot) + l_5 \tilde{y}(k))^2 \\ & - \gamma_1 \zeta_1^2(k) + 2\gamma_1 l_5^2 \tilde{y}^2(k) + 2\gamma_1 (w_1^T \phi_1(\cdot))^2 \end{aligned} \quad (10.A.3)$$

where $\zeta_1(k)$ is defined in (10.31).

Now taking the second term in the first difference (10.A.1) and substituting (10.63) into (10.A.2), we get

$$\begin{aligned}\Delta J_2(k) &= \frac{\gamma_2}{\alpha_2} [\tilde{w}_2^T(k+1)\tilde{w}_2(k+1) - \tilde{w}_2^T(k)\tilde{w}_2(k)] \\ &\leq -\gamma_2(1 - \alpha_2\|\phi_2(\cdot)\|^2)(\tilde{w}_2^T(k)\phi_2(\cdot) + l_6\tilde{x}_1(k) + l_6e_1(k))^2 \\ &\quad - \gamma_2\xi_2^2(k) + 3\gamma_2l_6^2\tilde{x}_1^2(k) + 3\gamma_2l_6^2e_1(k) + 3\gamma_2(w_2^T\phi_2(\cdot))^2\end{aligned}\tag{10.A.4}$$

Taking the third term in the first difference (10.A.1) and substituting (10.64) into (10.A.2), we get

$$\begin{aligned}\Delta J_3(k) &= \frac{\gamma_3}{\alpha_3} [\tilde{w}_3^T(k+1)\tilde{w}_3(k+1) - \tilde{w}_3^T(k)\tilde{w}_3(k)] \\ &\leq -\gamma_3(1 - \alpha_3\|\phi_3(\cdot)\|^2)(\tilde{w}_3^T(k)\phi_3(\cdot) + l_7\tilde{x}_2(k) + l_7e_2(k))^2 \\ &\quad - \gamma_3\xi_3^2(k) + 3\gamma_3l_7^2\tilde{x}_2^2(k) + 3\gamma_3l_7^2e_2^2(k) + 3\gamma_3(w_3^T\phi_3(\cdot))^2\end{aligned}\tag{10.A.5}$$

Similarly, we have

$$\begin{aligned}\Delta J_4(k) &= \gamma_4[F_0^2\tilde{x}_1^2(k) + (l_1 - R \cdot F_0)^2\tilde{y}^2(k) + \Delta F^2(k)e_1^2(k)] \\ &\quad + \gamma_4[(l'_1(k))^2e_2^2(k) + (l'_1(k))^2\xi_2^2(k) + d_{11}^2(k) - \tilde{x}_1^2(k)]\end{aligned}\tag{10.A.6}$$

where

$$l'_1(k) = R \cdot \Delta F(k) \cdot CE(k)\tag{10.A.7}$$

$$d_{11}(k) = R \cdot \Delta F(k) \cdot CE(k) \cdot w_2^T\phi_2(\cdot) - \Delta AF(k) - d_1(k)\tag{10.A.8}$$

and $\xi_2(k)$ is defined in (10.45).

$$\begin{aligned}\Delta J_5(k) &= \gamma_5[F_0^2\tilde{x}_2^2(k) + (l_2 - F_0)^2\tilde{y}^2(k) + d_{21}^2(k) - \tilde{x}_2^2(k)] \\ &\quad + \gamma_5[((1 - CE(k))\Delta F(k))^2(e_2^2(k) + \xi_2^2(k))]\end{aligned}\tag{10.A.9}$$

where

$$d_{21} = -d_2(k) - \Delta F(k)(1 - CE(k)) \cdot w_2^T\phi_2(k)\tag{10.A.10}$$

$$\Delta J_6(k) \leq \gamma_6(\xi_1^2(k) + l_3^2\tilde{y}^2(k) + d_3^2(k) - \tilde{y}^2(k))\tag{10.A.11}$$

$$\Delta J_7(k) \leq \gamma_7(g_1^2(k)e_2^2(k) + g_1^2(k)\zeta_2^2(k) + d_{12}^2(k) - e_1^2(k)) \quad (10.A.12)$$

$$\Delta J_8(k) \leq \gamma_8(l_4^2e_2^2(k) + l_4^2\tilde{x}_2^2(k) + \zeta_3^2(k) + d_{22}^2(k) - e_2^2(k)) \quad (10.A.13)$$

Combining (10.A.3) through (10.A.13) to get the first difference of the Lyapunov function and simplifying it, we get

$$\begin{aligned} \Delta J(k) \leq & -\gamma_1(1 - \alpha_1\|\phi_1(\cdot)\|^2)(\hat{w}_1^T(k)\phi_1(\cdot) + l_5\tilde{y}(k))^2 \\ & -\gamma_2(1 - \alpha_2\|\phi_2(\cdot)\|^2)(\hat{w}_2^T(k)\phi_2(\cdot) + l_6\tilde{x}_1(k) + l_6e_1(k))^2 \\ & -\gamma_3(1 - \alpha_3\|\phi_3(\cdot)\|^2)(\hat{w}_3^T(k)\phi_3(\cdot) + l_7\tilde{x}_2(k) + l_7e_2(k))^2 \\ & -(\gamma_1 - \gamma_6)\zeta_1^2(k) - (\gamma_2 - \gamma_4)(l'_1(k))^2 - \gamma_5((1 - CE(k))\Delta F(k))^2 \\ & -\gamma_7g_1^2(k)\zeta_2^2(k) - (\gamma_3 - \gamma_8)\zeta_3^2(k) - ((1 - F_0^2)\gamma_4 - 3\gamma_2l_6^2)\tilde{x}_1^2(k) \\ & -((1 - F_0^2)\gamma_5 - 3\gamma_3l_7^2 - \gamma_8l_4^2)\tilde{x}_2^2(k) - ((1 - l_3^2)\gamma_6 \\ & -(l_1 - R \cdot F_0)^2\gamma_4 - (l_2 - F_0)^2\gamma_5 - 2\gamma_1l_5^2)\tilde{y}^2(k) - (\gamma_7 - 3\gamma_2l_6^2 \\ & -\gamma_4\Delta F^2(k))e_1^2(k) + D_M^2 - ((1 - l_4^2)\gamma_8 - 3\gamma_3l_7^2 - \gamma_4(l'_1(k))^2 \\ & -\gamma_5((1 - CE(k))\Delta F(k))^2 - \gamma_7g_1^2(k))e_2^2(k) \end{aligned} \quad (10.A.14)$$

where

$$\begin{aligned} D_M^2 = & 2\gamma_1 w_{1m}^2 \phi_{1m}^2 + 3\gamma_2 w_{2m}^2 \phi_{2m}^2 + 3\gamma_3 w_{3m}^2 \phi_{3m}^2 + \gamma_4 d_{11m}^2 \\ & + \gamma_5 d_{21m}^2 + \gamma_6 d_{3m}^2 + \gamma_7 d_{12m}^2 + \gamma_8 d_{22m}^2 \end{aligned} \quad (10.A.15)$$

Choose

$\gamma_1 = 2$, $\gamma_2 = 1$, $\gamma_3 = 2$, $\gamma_4 = 1/6R^2\Delta F_m^2$, $\gamma_5 = 1/6\Delta F_m^2$, $\gamma_6 = 1$, $\gamma_7 = 1/3R^2$, and $\gamma_8 = 1$, then, (10.A.14) is simplified as

$$\begin{aligned} \Delta J(k) \leq & -2(1 - \alpha_1\|\phi_1(\cdot)\|^2)(\hat{w}_1^T(k)\phi_1(\cdot) + l_5\tilde{y}(k))^2 \\ & -(1 - \alpha_2\|\phi_2(\cdot)\|^2)(\hat{w}_2^T(k)\phi_2(\cdot) + l_6\tilde{x}_1(k) + l_6e_1(k))^2 \\ & -2(1 - \alpha_3\|\phi_3(\cdot)\|^2)(\hat{w}_3^T(k)\phi_3(\cdot) + l_7\tilde{x}_2(k) + l_7e_2(k))^2 \\ & -\zeta_1^2(k) - \frac{1}{3}\zeta_2^2(k) - \zeta_3^2(k) - \left(\frac{(1 - F_0^2)}{6R^2\Delta F_m^2} - 3l_6^2\right)\tilde{x}_1^2(k) \\ & -\left(\frac{(1 - F_0^2)}{6\Delta F_m^2} - 6l_7^2 - l_4^2\right)\tilde{x}_2^2(k) \end{aligned}$$

$$\begin{aligned}
& - \left((1 - l_3^2) - \frac{(l_1 - R \cdot F_0)^2}{6R^2 \Delta F_m^2} - \frac{(l_2 - F_0)^2}{6\Delta F_m^2} - 4l_5^2 \right) \tilde{y}^2(k) \\
& - \left(\frac{1}{6R^2} - 3l_6^2 \right) e_1^2(k) + D_M^2 - \left((1 - l_4^2) - 6l_7^2 - \frac{2}{3} \right) e_2^2(k)
\end{aligned} \tag{10.A.16}$$

This implies $\Delta J(k) < 0$ as long as (10.66) to (10.68) hold and

$$|\zeta_1(k)| > D_M \tag{10.A.17}$$

or

$$|\zeta_2(k)| > \sqrt{3}D_M \tag{10.A.18}$$

or

$$|\zeta_3(k)| > D_M \tag{10.A.19}$$

or

$$|\tilde{x}_1(k)| > \frac{D_M}{\sqrt{(1 - F_0^2)/(6R^2 \Delta F_m^2) - 3l_6^2}} \tag{10.A.20}$$

or

$$|\tilde{x}_2(k)| > \frac{D_M}{\sqrt{(1 - F_0^2)/(6\Delta F_m^2) - 6l_7^2 - l_4^2}} \tag{10.A.21}$$

or

$$|\tilde{y}(k)| > \frac{D_M}{\sqrt{(1 - l_3^2) - (l_1 - R \cdot F_0)^2/(6R^2 \Delta F_m^2) - (l_2 - F_0)^2/(6\Delta F_m^2) - 4l_5^2}} \tag{10.A.22}$$

or

$$|e_1(k)| > \frac{D_M}{\sqrt{\frac{1}{6R^2} - 3l_6^2}} \tag{10.A.23}$$

or

$$|e_2(k)| > \frac{D_M}{\sqrt{(\frac{1}{3} - l_4^2) - 6l_7^2}} \tag{10.A.24}$$

According to a standard Lyapunov extension theorem (Lewis et al. 1993), this demonstrates that the system tracking error and the weight estimation errors are UUB. The boundedness of $\|\zeta_1(k)\|$, $\|\zeta_2(k)\|$, and $\|\zeta_3(k)\|$ implies that $\|\tilde{w}_1(k)\|$, $\|\tilde{w}_2(k)\|$, and $\|\tilde{w}_3(k)\|$ are bounded and this further implies that the weight estimates $\hat{w}_1(k)$, $\hat{w}_2(k)$, and $\hat{w}_3(k)$ are bounded. Therefore all the signals in the closed-loop system are bounded.

APPENDIX 10.B

```
*****
```

Author: Jonathan Vance
 File: engcon.c
 Date: 2005

Main program monitors digital I/O ports of PC770 board for pressure data. Pressure data are received according to a handshaking protocol where the PC requests data and the microcontroller sends the data and a valid data signal. Upon receipt of a start signal the received pressure data is integrated with cylinder volume taken into account to find heat release.

Heat release is the input to the neural network controller that calculates the fuel adjustment for the next cycle.

The change in fuel mass for the next engine cycle is input into an equation that returns a fuel pulse width according to fuel injector flow specific to the CFR engine.

The fuel pulse width is processed again to find a 16-bit timer value to load into the microcontroller.

Raw pressure measurements come from the A/D as 8-bit values. These values are input to an equation that converts them to corresponding voltages. The calculated voltages are input to another equation where the output is pressure in kilopascals.

```
voltage = f(8-bit_raw_pressure_data); V
pressure = g(voltage); kPa
```

On power-up, hardware reset, and software reset the digital I/O ports of the PC770 are pulled high and configured as inputs.

```
*****
```

```
/
```

```
#define n 35 //controller nodes
#define n0 35 //observer nodes

#define WEIGHT_INIT 0.1
```



```
int main (int argc, char** argv)
{

    NeuralNetwork *NN;
    EZIOCINIT init;
    FILE *Controller_output;
    FILE *Controller_param;
    FILE *Pressure_data;
    time_t now;
    int INPUT,OUTPUT,START,READY,RQST,CONTROL;
    int a,i,idata,iFF,ii;
    int flag;
    int press_index;
    int pressure[MAX_PRESS_INDEX]; /*raw unsigned 8-bit pressure
from A/D*/
    float pressure2[MAX_PRESS_INDEX]; /*pressure in kilopascals*/
    float voltage[MAX_PRESS_INDEX]; /*raw pressure from a/d converted to
voltages*/
    int press_shift;
    float press_shift2;

    extern float Qk;
    extern float u;
    extern float u_scale;
    extern float PHI;
    extern float MF;
    extern float AF;
    extern float F;
    extern float tm;
    extern float mwf;
    extern float mwa;
    extern float CEmax;
    extern float l1, l2, l3, l4, l5, l6, l7, a0, a1, a2;

    unsigned char out_index;
    unsigned char uedata, ucdatalast;
    unsigned char start_status, rqst_status, ready_status,
calc_status, control_status;
    unsigned char new_fuel_LSByte, new_fuel_MSByte,
new_fuel_info;
    unsigned char useNN;
    float new_fuel_pw;
    float Qkgood,Qktemp;
    float setpoint;
    float dt=0.00008334;
    float dt_scale;
    float Temp;
    float BarPress; /*ambient pressure in kilopascals (kPa)*/
    float RelHum;
    char junk[20];
    float fa,fb;
    int measure_press=0;
```

```
int use_ambient=0;
float avg_fuel=0;
float avg_fuel_temp=0;
int avg_i=0;
int pambtemp;
float vambtemp2;

const float gamma=1.4;
const float conpi=3.141592653;
const float bore=0.001*82.5;
const float stroke=0.001*114.3;
const float rod_length=0.001*254.0;
const float compression_ratio=9.0;
const int soc=346;
const int eoc=490;

float displacement, clearance_volume, volume_max, crank_radius,
rod_crank_ratio, angle;

float const1;
float const2;
float dvdt[1440];
float vcyl[1440];

float dh,dpdt,hr0;

/*CALCULATE ENGINE PARAMETERS FOR HEAT RELEASE
MEASUREMENT***** */
displacement=0.25*conpi*stroke*bore*bore;
clearance_volume=displacement/(compression_ratio-1.0);
volume_max=compression_ratio*clearance_volume;
crank_radius=0.5*stroke;
rod_crank_ratio=rod_length/crank_radius;

for (i=0;i<1440;i=i+1)
{
    angle=0.5*i*conpi/180.0;
    vcyl[i]=1.0+0.5*(compression_ratio-1.0)*(rod_crank_ratio+1.0-cos
(angle)-sqrt(rod_crank_ratio*rod_crank_ratio-(sin(angle)*sin(angle)) ));
    vcyl[i]=vcyl[i]*clearance_volume;
}

for (i=1;i<1439;i=i+1)
{
    dvdt[i]=(vcyl[i+1]+vcyl[i-1])/(2.0*(conpi/180.0));
}

dh=0.5*(conpi/180.0);

const1=(gamma/(gamma-1.0));
```

```

const2=(1.0/(gamma-1.0));

/*INITIALIZE PROGRAM FILES ****
****

now = time((time_t *)NULL);
flag=1;
printf("\033[2J");

Controller_param = fopen("engcon.cfg", "r"); /*get controller
parameters*/ if (Controller_param==NULL)
    {perror("Unable to open file.");}
fscanf(Controller_param,"%s %f",&junk,&MF);
fscanf(Controller_param,"%s %f",&junk,&AF);
fscanf(Controller_param,"%s %f",&junk,&F);
fscanf(Controller_param,"%s %f",&junk,&dt_scale);
fscanf(Controller_param,"%s %f",&junk,&Phi_Upper);
fscanf(Controller_param,"%s %f",&junk,&Phi_Lower);
fscanf(Controller_param,"%s %f",&junk,&R);
fscanf(Controller_param,"%s %f",&junk,&Temp);
fscanf(Controller_param,"%s %f",&junk,&BarPress);
fscanf(Controller_param,"%s %f",&junk,&RelHum);
fscanf(Controller_param,"%s %f",&junk,&tm);
fscanf(Controller_param,"%s %f",&junk,&mwf);
fscanf(Controller_param,"%s %f",&junk,&mwa);
fscanf(Controller_param,"%s %f",&junk,&CEmax);
fscanf(Controller_param,"%s %f",&junk,&l1);
fscanf(Controller_param,"%s %f",&junk,&l2);
fscanf(Controller_param,"%s %f",&junk,&l3);
fscanf(Controller_param,"%s %f",&junk,&l4);
fscanf(Controller_param,"%s %f",&junk,&l5);
fscanf(Controller_param,"%s %f",&junk,&l6);
fscanf(Controller_param,"%s %f",&junk,&l7);
fscanf(Controller_param,"%s %f",&junk,&a0);
fscanf(Controller_param,"%s %f",&junk,&a1);
fscanf(Controller_param,"%s %f",&junk,&a2);

printf("\033[0;30;47mENGINE      CONTROLLER      WITH      LEAN      COMBUSTION
ALGORITHM\033[0;37;40m\n\n");
printf("\t%s\n\n", ctime(&now));

dt=dt*dt_scale;

printf("Measure pressure(0:no;1:yes)?: ");
scanf("%d",&measure_press);
printf("Use ambient pressure information(0:no;1:yes)?: ");
scanf("%d",&use_ambient);
printf("Enter open-loop fuel pulse-width (%.2f ms<pw<% .2f
ms):
",MIN_FUEL_PW,MAX_FUEL_PW);
scanf("%f",&setpoint);
out_index=3;

```

```

new_fuel_info=0x00;
if (!(setpoint>MIN_FUEL_PW && setpoint<MAX_FUEL_PW))
{
    setpoint=13.0;
}
new_fuel_MSByte=timer_load_calc(setpoint,1);
new_fuel_LSByte=timer_load_calc(setpoint,0);

avg_fuel=setpoint;
PHI=R*(MF/AF);

/*SETUP FILE TO STORE U(K),Q(K),etc - appending new data at
eof***** */

if (!(Controller_output=fopen("CTRL.dat", "a")))
{
    perror("Error opening CTRL.dat file to store u(k) and Q(k).");
    return 1;
}
fprintf(Controller_output, "\n\nENGINE_CONTROLLER_DATA:_%s\n",ctime
(&now));
fprintf(Controller_output, "\tOpen-loop_setpoint:%f;Qk_dt_scale: %f\n",
setpoint,dt_scale);
fprintf(Controller_output, "\tnnodes=%d;n0nodes=%d\n",n,n0);
fprintf(Controller_output, "\tAF=%f; MF=%f\n",AF,MF);
fprintf(Controller_output, "\tR=%f; F=%f \n",R,F);
fprintf(Controller_output, "\ttm=%f;           mwf=%f;      mwa=%f;
CEmax=%f\n",tm,mwf,mwa,CEmax);
fprintf(Controller_output, "\tPHI=%f;          Phi_Lower=%f;
Phi_Upper=%f\n",PHI,Phi_Lower,Phi_Upper);
fprintf(Controller_output, " \tTemp=%f;          Barom.Pressure=%f;
Rel.Humidity=%f\n",Temp,BarPress,RelHum);
fprintf(Controller_output, "\n");
fprintf(Controller_output, "k u(k) Q(k) Fuel(ms) ah(k-1) mh(k-1) Qh(k-1)
Pdata \n");

/*SETUP FILE TO STORE PRESSURE
MEASUREMENTS***** */

if (measure_press==1)
{
    if (!(Pressure_data=fopen("PRESS.dat", "w")))
    {
        perror("Error opening PRESS.dat");
        return 1;
    }
}

printf("\nInitializing NN...\n");
NN=(NeuralNetwork *)malloc(sizeof(NeuralNetwork));
if (NN==NULL)
{

```

```
printf("No memory for NN %d.\n",1);
perror("NN memory error");
return 1;
}
Controller_initialize(NN);
printf("NN initialized.\n");
/*printf("NN->w1[2]=%f\n",NN->w1[2]);*/
// INITIALIZE I/O PORT DESCRIPTORS /////////////////////////////////
/////
printf("Initializing I/O ports...\n");

INPUT = open("/dev/ezio",O_RDWR);
if(INPUT < 0)
{
    perror("Error opening file for INPUT.");
    return 1;
}
OUTPUT = open("/dev/ezio",O_RDWR);
if(OUTPUT < 0)
{
    perror("Error opening file for OUTPUT.");
    return 1;
}
START = open("/dev/ezio",O_RDWR);
if(START < 0)
{
    perror("Error opening file for START.");
    return 1;
}
READY = open("/dev/ezio",O_RDWR);
if(READY < 0)
{
    perror("Error opening file for READY.");
    return 1;
}
RQST = open("/dev/ezio",O_RDWR);
if(RQST < 0)
{
    perror("Error opening file for RQST.");
    return 1;
}
CONTROL = open("/dev/ezio",O_RDWR);
if(CONTROL < 0)
{
    perror("Error opening file for CONTROL.");
    return 1;
}

//INPUT ON PORT A /////////////////////////////////
/////
init.bit0 = 0;
```

```
init.width = 8; //pins 19,21,23,25,24,22,20,18 - LSB to MSB respectively
if(ioctl(INPUT,EZIIOC_INIT,&init))
{
printf("While trying to set init for port A for INPUT.\n");
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(INPUT,EZIIOC_DDR,0x00); //set I/O to input

//OUTPUT ON PORT B /////////////////////////////////
/////
init.bit0 = 8;
init.width = 8; //pins 10,8,4,6,1,3,5,7 - LSB to MSB respectively
if(ioctl(OUTPUT,EZIIOC_INIT,&init))
{
printf("While trying to set init for port B for OUTPUT.\n");
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(OUTPUT,EZIIOC_DDR,0xFF); //set I/O to output

//RQST ON PORT C bit 0 ///////////////////////////////
/////
init.bit0 = 16;
init.width = 1; //pin 13
if(ioctl(RQST,EZIIOC_INIT,&init))
{
printf("While trying to set init for port C bit 0 for RQST.\n");
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(RQST,EZIIOC_DDR,0xE3); //set I/O to output

//READY ON PORT C bit 1 //////////////////////////////
/////
init.bit0 = 17;
init.width = 1; //pin 16
if(ioctl(READY,EZIIOC_INIT,&init))
{
printf("While trying to set init for port C bit 1 for READY.\n");
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(READY,EZIIOC_DDR,0xE3); //set I/O to output

//START ON PORT C bit 2 and 3 //////////////////////////////
/////
init.bit0 = 18;
init.width = 2; //pins 15 and 17 (bits 2 and 3 respectively)
if(ioctl(START,EZIIOC_INIT,&init))
{
printf("While trying to set init for port C bit 2:3 for START.\n");
```

```
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(START,EZIOC_DDR,0xE3); //set I/O to input

//CONTROL ON PORT C bit 4 /////////////////////////////////
/////
init.bit0 = 20;
init.width = 1; //pin 14 (PC bit 4 respectively)
if(ioctl(CONTROL,EZIOC_INIT,&init))
{
printf("While trying to set init for port C bit 4 for CONTROL.\n");
perror("IOCTL error initializing interface for digital I/O port.");
return 1;
}
ioctl(CONTROL,EZIOC_DDR,0xE3); //set I/O to input
printf("I/O ports initialized.\n");
///////////////////////////////
////

indk=1;
press_index=0;
start_status=0;
control_status=0;
calc_status=0;
rqst_status=0;
press_shift=0;
press_shift2=0;
vambtemp2=0;
write(RQST,&rqst_status,1);
ready_status=0;
write(READY,&ready_status,1);
printf("Running... \n\n");

*****+
start_status: 0: start,xdata 0,0
           1: start,xdata 0,1
           2: start,xdata 1,0
           3: start,xdata 1,1
***** /

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

while(1)
{
read(START,&start_status,1);
if (calc_status==0)
{
if (start_status==0 && rqst_status==0)
{
//printf("Requesting pressure info...\n");
}
```

```
rqst_status=1;
write(RQST,&rqst_status,1);
}
else if (start_status==1 && rqst_status==1) //requested data is ready
{
read(INPUT,&ucdata,1);
if (press_index==0)
{
pambtemp=((int) ucdata);
vambtemp2=vc0+vc1*((float) pambtemp);

pressure[press_index]=pambtemp;
voltage[press_index]=vambtemp2;
pressure2[press_index]=(vambtemp2*1000.0/1.25);

if (use_ambient==1)
{
press_shift=pambtemp;
press_shift2=BarPress-(vambtemp2*1000.0/1.25);
}
else
{
/*pressure array already initialized*/
pressure[press_index]=((int) ucdata);
voltage[press_index]=vc0+vc1*((float) ucdata);
pressure2[press_index]=(voltage[press_index]*1000.0/1.25) +
press_shift2;
}

if (press_index<MAX_PRESS_INDEX-1)
{
press_index=press_index+1;
}
rqst_status=0;
write(RQST,&rqst_status,1);
}
else if ((start_status==2 || start_status==3))
/* perform integrate, calc NN,
create new fuel pulse width, output new fuel pulse width,
assert ready_status */
{
rqst_status=0;
write(RQST,&rqst_status,1);
read(CONTROL,&control_status,1);
Qk=0.0;
for(i=2;i<press_index;i=i+1)
{
dpdt=(pressure2[i]-pressure2[i-1])/dh;

hr0=(const1*pressure2[i]*dvdt[i+2*soc])+(const2*v cyl[i+2*soc]*dpdt);
```

```

Qk=Qk+hr0;
}

Qk=Qk*dt_scale;

if (control_status)
{
    Controller(&flag, NN);

    new_fuel_pw = ((605.13312144*(u_scale+MF))-0.38967856);

    if (new_fuel_pw > MAX_FUEL_PW){new_fuel_pw=MAX_FUEL_PW;}
    else if (new_fuel_pw < MIN_FUEL_PW){new_fuel_pw=MIN_FUEL_PW; }

    if (avg_i<100)
    {
        avg_i=avg_i+1;
        avg_fuel_temp=avg_fuel_temp+new_fuel_pw;
    }
    else
    {
        avg_i=0;
        avg_fuel=avg_fuel_temp/100.0;
        avg_fuel_temp=0;
    }

    new_fuel_MSByte=timer_load_calc(new_fuel_pw,1);
    new_fuel_LSByte=timer_load_calc(new_fuel_pw,0);
}
else
{
    u=0;
    new_fuel_pw=setpoint;
    new_fuel_MSByte=timer_load_calc(setpoint,1);
    new_fuel_LSByte=timer_load_calc(setpoint,0);
}

new_fuel_info=0x00;
out_index=3;
calc_status=1;

}

}

else if (calc_status==1)
{
    if ((start_status==2 || start_status==3))
    {
        if (start_status==2 && ready_status==1)
        {
            ready_status=0;
            write(READY,&ready_status,1);
        }
    }
}

```

```
}

else if (start_status==2 && ready_status==0){;}
else if (start_status==3 && ready_status==1){;}
else if (start_status==3 && ready_status==0)
{

    if(new_fuel_MSByte==235){return 0;}
    if(new_fuel_MSByte==0){return 0;}


    if (out_index==3)
    {
        //printf("\033[0;30;40m%d-",new_fuel_info);
        write(OUTPUT,&new_fuel_info,1);
        //printf("%d-",new_fuel_info);
    }
    else if (out_index==2)
    {
        //printf("%d-",new_fuel_MSByte);
        write(OUTPUT,&new_fuel_MSByte,1);
        //printf("%d-",new_fuel_MSByte);
    }
    else if (out_index==1)
    {
        //printf("%d-",new_fuel_LSByte);
        write(OUTPUT,&new_fuel_LSByte,1);
        //printf("%d\033[0;37;40m",new_fuel_LSByte);
    }
    else if (out_index==0){;}
    ready_status=1;
    write(READY,&ready_status,1);
    while(start_status==3){read(START,&start_status,1);}
    out_index=out_index-1;
}
}
else if (start_status==0)
{
    ready_status=0;
    write(READY,&ready_status,1);
    calc_status=0;

    //PRINT MESSAGES
    printf("\n\033[0;32;40mCycle %d\033[0;37;40m\n",indk);
    printf("Qk=\033[0;33;40m%f\033[0;37;40m",Qk);
    printf(" from %d pressure measurements:\n",press_index);
    printf("\tpressure shift=%f\n",press_shift2);
    printf("\tpressure[1,50,100,175,250]=[%.1f,%.1f,%.1f,%.1f]\n",
    pressure2[1],pressure2[5
    0],pressure2[100],pressure2[175],pressure2[250]),

    printf("New fuel pulse width: \033[0;36;40m%.4f\033[0;37;40m ms\n",
    new_fuel_pw);
    printf("Average control fuel: \033[0;36;40m%.4f\033[0;37;40m ms\n",
    
```

```

    avg_fuel);

//SAVE u(k) and Q(k) INFORMATION IN FILE FOR ANALYSIS
fprintf(Controller_output,"%d %f %f %f %f %f %d\n",indk,u,Qk,new_
fuel_pw, NN->xh[0],NN->xh[1],NN->xh[2],press_index);

//fprintf(Pressure_data,"Pressure for cycle %d\n",indk);
if (measure_press==1 && use_ambient==0)
{
    for (ii=0; ii<press_index; ii=ii+1) //to press_index
    {
        fprintf(Pressure_data,"%d %d \n",ii,pressure[ii]);
    }
}
else if (measure_press==1 && use_ambient==1)
{
    for (ii=1; ii<press_index; ii=ii+1) //to press_index
    {
        if (ii==1)
        {
            fprintf(Pressure_data,"%d %d %d\n",ii,pressure[1],pressure[0]);
        }
        else
        {
            fprintf(Pressure_data,"%d %d\n",ii,pressure[ii]);
        }
    }
    indk=indk+1;
    press_index=0;
    press_shift=0;
    press_shift2=0;
    vambtemp2=0;
}
}

return 0;
}

/*ooooooooooooooooooooooooooooo*/

void Controller_initialize(NeuralNetwork *NN)
{
    FILE *Controller_initvals;
    time_t now;
    int i,k;
    float input0[5];
    float input1[6];
    float input2[4];
    float sum;
    float phi_Group;
}

```

```
float phi_Mean;
extern float u;
extern float Qk;
extern float u_scale;
extern float PHI;
extern float MF;
extern float AF;
extern float F;
extern float R;
extern float Phi_Upper;
extern float Phi_Lower;
extern float tm;
extern float mwf;
extern float mwa;
extern float CEmax;
extern float l1, l2, l3, l4, l5, l6, l7, a0, a1, a2;
FILE *Controller_weights;
char junk[20];

now = time((time_t *)NULL);

if (!(Controller_initvals=fopen("NN.dat", "a")))
{
    perror("Error opening NN.dat file.");
}
fprintf(Controller_initvals, "\nNN_INITIAL_WEIGHTS:_%s\n", ctime(&now));

/**Initialize the controller gain and control input u*****
 ****
u=0;    //0;    initial fuel change(nominal)
u_scale=u;
NN->xh[0]=AF;    //14; estimated total air (using nominal value) (target)
NN->xh[1]=MF;    //0.7; estimated total fuel (nominal) (target)
NN->xh[2]=0.03;  //0.5; estimated heat release (target)
NN->X2d=MF;      //0; desired fuel
Qk=0.03;         //0.5; heat release

PHI=R*(MF/AF);

/*Calculate the desired values of total air, fuel and heat release****/
NN->x_desired[0]=tm/(PHI/(R*mwf)+1.0/mwa);
NN->x_desired[1]=tm/(R/(PHI*mwa)+1.0/mwf);
phi_Mean=(Phi_Upper+Phi_Lower)/2.0;
phi_Group=(PHI-phi_Mean)/(Phi_Upper-Phi_Lower);
/*NN->x_desired[0]=AF;
NN->x_desired[1]=MF;*/

NN->x_desired[2]=NN->x_desired[1]*CEmax/(1.0+pow(100.0,-phi_Group));

//printf("x_dsr[0]=%f;  x_dsr[1]=%f;  x_dsr[2]=%f;\n",NN->x_desired[0],
NN->x_desired[1],NN->x_desired[2]);
```

```

*****Initialize the weights of neural
network*****
for(i=0;i<n0;i++)
{
    for(k=0;k<5;k++)
    {
        NN->v0[i][k]=((float)n)* ((float)rand()/(float)RAND_MAX);
        fprintf(Controller_initvals,"NN->v0[%d][%d]=%f\n",i,k,NN->v0[i][k]);
    }
}
for(i=0;i<n;i++)
{
    for(k=0;k<6;k++)
    {
        NN->v1[i][k]=0.2+((float)rand()/(float)RAND_MAX);
        fprintf(Controller_initvals,"NN->v1[%d][%d]=%f\n",i,k,NN->v1[i][k]);
    }
}
for(i=0;i<n;i++)
{
    for(k=0;k<4;k++)
    {
        NN->v2[i][k]=((float)rand())/((float)RAND_MAX);
        fprintf(Controller_initvals,"NN->v2[%d][%d]=%f\n",i,k,NN->v2[i][k]);
    }
}

/*READ INITIAL WEIGHTS FROM NNw.dat*/
if (!(Controller_weights=fopen("NNw.cfg", "r")))
{
    perror("Error opening NNw.cfg file.");
}
for(i=0;i<n0+1;i++)
{
    fscanf(Controller_weights,"%s %f",&junk,&NN->w0[i]);
}
for(i=0;i<n+1;i++)
{
    fscanf(Controller_weights,"%s %f",&junk,&NN->w1[i]);
}
for(i=0;i<n+1;i++)
{
    fscanf(Controller_weights,"%s %f",&junk,&NN->w2[i]);
}
fclose(Controller_weights);

/*for(i=0;i<n0+1;i++)
{
    NN->w0[i]=0;
}

```

```
NN->w0_old[i]=0;
}
for(i=0;i<n+1;i++)
{
    NN->w1[i]=0;
    NN->w2[i]=0;
} */

/*Input      vector      of      neural
network***** */
input0[0]=NN->xh[0];
input0[1]=NN->xh[1];
input0[2]=NN->xh[2];
input0[3]=u;
input0[4]=1.0;

input1[0]=NN->xh[0];
input1[1]=NN->xh[1];
input1[2]=NN->x_desired[0];
input1[3]=AF;
input1[4]=R;
input1[5]=1.0;

input2[0]=NN->xh[0];
input2[1]=NN->xh[1];
input2[2]=NN->x_desired[1];
input2[3]=1.0;

/*Initialize the output of hidden layer***** */
*****
for(i=0;i<n0;i++)
{
    sum=0;
    for(k=0;k<5;k++)
    {
        sum=sum+NN->v0[i][k]*input0[k];
    }
    NN->psi0[i]=2.0/(1.0+exp(-2.0*sum))-1.0;
}
NN->psi0[i]=1.0;

for(i=0;i<n;i++)
{
    sum=0;
    for(k=0;k<6;k++)
    {
        sum=sum+NN->v1[i][k]*input1[k];
    }
    NN->psi1[i]=2.0/(1.0+exp(-2.0*sum))-1.0;
}
NN->psi1[i]=1.0;
```

```

for(i=0;i<n;i++)
{
    sum=0;
    for(k=0;k<4;k++)
    {
        sum=sum+NN->v2[i][k]*input2[k];
    }
    NN->psi2[i]=2.0/(1.0+exp(-2.0*sum))-1.0;
}
NN->psi2[i]=1.0;

/*Initialize the output of Neural
network***** */
sum=0;
for(i=0;i<n0+1;i++)
{
    sum=sum+NN->w0[i]*NN->psi0[i];
}
NN->NN0_output=sum;

sum=0;
for(i=0;i<n+1;i++)
{
    sum=sum+NN->w1[i]*NN->psi1[i];
}
NN->NN1_output=sum;

sum=0;
for(i=0;i<n+1;i++)
{
    sum=sum+NN->w2[i]*NN->psi2[i];
}
NN->NN2_output=sum;

***** */

//printf("in INIT: NN2_output: %f\n",NN->NN2_output);
//printf("Controller_init: u=%f\n",u);

fclose(Controller_initvals);

}

void Controller(int *flag, NeuralNetwork *NN)
{
    float input0[5];
    float input1[6];
    float input2[4];
    float sum;
    int i,k;
    extern float u;
}

```

```

extern float Qk;
extern float u_scale;
extern float PHI;
extern float MF;
extern float AF;
extern float F;
extern float R;
extern float Phi_Upper;
extern float Phi_Lower;
extern float tm;
extern float mwf;
extern float mwa;
extern float CEmax;
extern float 11, 12, 13, 14, 15, 16, 17, a0, a1, a2;
FILE *Controller_weights;

*flag=1;

*****Error
system*****
*****/
NN->eh[0] = NN->xh[0] - NN->x_desired[0];
NN->eh[1] = NN->xh[1] - NN->X2d;
NN->eh[2] = NN->xh[2] - Qk;

*****Calculate the control input
u*****
u = NN->NN2_output + 14*(NN->eh[1]);
//printf("controller u: %f\n",u);
u_scale=u;

*****Updating weight matrix of neural
network*****
for(i=0;i<n0+1;i++)
{
    NN->w0_old[i] = NN->w0[i];
    NN->w0[i] = NN->w0[i] - a0 * NN->psi0[i] * (NN->NN0_output + 15 *
NN->eh[2]);//15=0.5
}
for(i=0;i<n+1;i++)
{
    NN->w1[i] = NN->w1[i] - a1 * NN->psi1[i] * (NN->NN1_output + 16 *
NN->eh[0]);//16=0.1
    NN->w2[i] = NN->w2[i] - a2 * NN->psi2[i] * (NN->NN2_output + 17 *
NN->eh[1]);//17=0.07
}

*****Update the Observer neural network
(NN0)*****
input0[0] = NN->xh[0];
input0[1] = NN->xh[1];

```

```

input0[2] = NN->xh[2];
input0[3] = u;
input0[4] = 1.0;
for(i=0;i<n0;i++)
{
    sum = 0;
    for(k=0;k<5;k++)
    {
        sum = sum + NN->v0[i][k] * input0[k];
    }
    NN->psi0[i] = 2.0/(1.0 + exp(-2.0 * sum)) - 1.0;
    /*printf("NN->psi0[%d]=%f\n",i,NN->psi0[i]);*/
}
NN->psi0[n0] = 1.0;

sum=0;
for(i=0;i<n0+1;i++)
{
    sum = sum + NN->w0_old[i] * NN->psi0[i];
}
NN->NN0_output = sum;

NN->xh[0] = F * NN->xh[0] - R * F * NN->xh[2] + AF + 11 * NN->eh[2];
NN->xh[1] = F * NN->xh[1] - F * NN->xh[2] + (MF + u) + 12 * NN->eh[2];

//printf("NN->xh[2] before: %f ",NN->xh[2]);

NN->xh[2] = NN->NN0_output + 13 * NN->eh[2];
//NN->xh[2] = NN->NN0_output + 13 * NN->eh[2] - 0.99 * NN->eh[2];

//printf("NN->xh[2] after: %f \n",NN->xh[2]);
//printf("NN->eh[2]:%f*13:%f NN->NN0_output:%f\n",NN->eh[2],13*NN->eh[2],
NN->NN0_output);

*****Update      output      of      hidden
layer*****
input1[0] = NN->xh[0];
input1[1] = NN->xh[1];
input1[2] = NN->x_desired[0];
input1[3] = AF;
input1[4] = R;
input1[5] = 1.0;

for(i=0;i<n;i++)
{
    sum = 0;
    for(k=0;k<6;k++)
    {
        sum = sum + NN->v1[i][k] * input1[k];
    }
    NN->psi1[i] = 2.0/(1.0 + exp(-2.0 * sum)) - 1.0;
}

```

```
NN->psi1[n] = 1.0;

sum=0;
for(i=0;i<n+1;i++)
{
    sum = sum + NN->w1[i] * NN->psi1[i];
}
NN->NN1_output = sum;
NN->X2d = NN->NN1_output;

/*********************Update the NN2 neural
network*****/
input2[0] = NN->xh[0];
input2[1] = NN->xh[1];
input2[2] = NN->x2d;
input2[3] = 1.0;

for(i=0;i<n;i++)
{
    sum = 0;
    for(k=0;k<4;k++)
        sum = sum + NN->v2[i][k] * input2[k];
    NN->psi2[i] = 2.0/(1.0 + exp(-2.0 * sum)) - 1.0;
}
NN->psi2[n] = 1.0;

sum = 0;
for(i=0;i<n+1;i++)
{
    sum = sum + NN->w2[i] * NN->psi2[i];
}
NN->NN2_output = sum;

/*****************/
*****/
//printf("NN->xh[0]=%f    NN->xh[1]=%f    NN->xh[2]=%f\n
u=%f",NN->xh[0],NN->xh[1],NN->xh[2],u);
//printf("NN->v0[2][2]=%f\n",NN->v0[2][2]);

/*LOOP TO CATCH THE WEIGHTS AT x CYCLES*/
if (indk==60000)
{
    if (!(Controller_weights=fopen("NNw.dat", "w")))
    {
        perror("Error opening NNw.dat file.");
    }
    //fprintf(Controller_weights,"NN INITIAL OUTPUT LAYER WEIGHTS
AT CYCLE %d\n",indk);
    for(i=0;i<n0+1;i++)
    {
```

```

        fprintf(Controller_weights, "NN->w0[%d]= %.14f\n", i, NN->w0[i]);
    }
    for(i=0;i<n+1;i++)
    {
        fprintf(Controller_weights, "NN->w1[%d]= %.14f\n", i, NN->w1[i]);
    }
    for(i=0;i<n+1;i++)
    {
        fprintf(Controller_weights, "NN->w2[%d]= %.14f\n", i, NN->w2[i]);
    }
    fclose(Controller_weights);
}

}

/*@@@@@@@@@@@@@@@@@@@CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC*/



float integrate(int array[], int array_length, float dk)
{   //perform integration by summing trapezoids
    int i;
    float Fint;
    float fa;
    float fb;

    for(i=1; i<array_length; i=i+1)
    {
        fa = (float) array[i-1];
        fb = (float) array[i];
        Fint = Fint + (fa+0.5*(fb-fa));
    }
    Fint=Fint/dk;
    return Fint;
}

unsigned char timer_load_calc(float setpoint, unsigned char byte)
{
    float tmr_val;
    int i;
    unsigned char THbyte=0;
    unsigned char TLbyte=0;
    tmr_val = rintf( (2*-1666.6667*setpoint) + 65535 );
    for (i=15;i>=0;i=i-1)
    {
        if (tmr_val-((float) pow2(i))>=0)
        {
            tmr_val=tmr_val-((float) pow2(i));
            if (i>7) {THbyte=THbyte+(pow2(i-8));}
            else {TLbyte=TLbyte+(pow2(i));}
        }
    }
    if (byte) {return THbyte;}
}
```

```
    else {return TLbyte;}
```

```
}
```

```
int pow2(int x)
```

```
{
```

```
    int temp=1;
```

```
    int i;
```

```
    if(x==0){return 1;}
```

```
    else
```

```
{
```

```
        for(i=0;i<x;i=i+1)
```

```
        {
```

```
            temp=2*temp;
```

```
        }
```

```
        return temp;
```

```
    }
```

```
}
```

```
float integrate2(float array[], int array_length, float dk)
```

```
{
```

```
    int i;
```

```
    float F=0.0;
```

```
    float fa;
```

```
    float fb;
```

```
    for(i=1; i<array_length; i=i+1)
```

```
    {
```

```
        fa = (float) array[i-1];
```

```
        fb = (float) array[i];
```

```
        F = F + (fa+(0.5*(fb-fa)))*dk ;
```

```
    }
```

```
    return F;
```

```
}
```

```
/* ezioc.h */
```

```
//#include <linux/ioctl.h>
```

```
#include <sys/ioctl.h>
```

```
struct ezioc_init_struct
```

```
{
```

```
    unsigned int bit0;
```

```
    unsigned int width;
```

```
};
```

```
typedef struct ezioc_init_struct EZIOCINIT;
```

```
#define EZIOC_IOCTL_BASE 'e'
```

```
#define EZIOC_INIT _IOW(EZIOC_IOCTL_BASE, 0, EZIOCINIT)
```

```
#define EZIOC_DDR _IOW(EZIOC_IOCTL_BASE, 1, unsigned char)
```

```
//Engine Config file
```

```

MF 0.01773
AF 0.3831
F 0.09
dt_scale 1.0
Phi_Upper 0.675
Phi_Lower 0.655
R 15.13
Temp(F) 80.0
Baro.Pressure(kPa) 101.59
Rel.Humidity(%) 52.0
tm 0.021
mwf 114.0
mwa 28.84
CEmax 1.0
11 1.99
12 0.13
13 -0.9
14 0.12
15 0.5
16 0.1
17 0.07
a0 0.005
a1 0.03
a2 0.03

```

Notes:

Qk should be ~ 0.03 on average and <0.04 to match Daw model simulations.

```

//NN Config file
NN->w0[0]= 0.00018710833683
NN->w0[1]= 0.00018710833683
NN->w0[2]= 0.00018710833683
NN->w0[3]= 0.00018710833683
NN->w0[4]= 0.00018715634360
NN->w0[5]= 0.00018710833683
NN->w0[6]= 0.00018710833683
NN->w0[7]= 0.00018710833683
NN->w0[8]= 0.00018710833683
NN->w0[9]= 0.00018710833683
NN->w0[10]= 0.00018710833683
NN->w0[11]= 0.00018704912509
NN->w0[12]= 0.00018710833683
NN->w0[13]= 0.00018710833683
NN->w0[14]= 0.00018710833683
NN->w0[15]= 0.00018710833683
NN->w0[16]= 0.00018710833683
NN->w0[17]= 0.00018710833683
NN->w0[18]= 0.00018710833683
NN->w0[19]= 0.00018710833683
NN->w0[20]= 0.00018710833683

```

```
NN->w0[21]= 0.00018710833683
NN->w0[22]= 0.00018710833683
NN->w0[23]= 0.00018710833683
NN->w0[24]= 0.00018710833683
NN->w0[25]= 0.00018710833683
NN->w0[26]= 0.00018710839504
NN->w0[27]= 0.00018710833683
NN->w0[28]= 0.00018710629956
NN->w0[29]= 0.00018710833683
NN->w0[30]= 0.00018710833683
NN->w0[31]= 0.00018710833683
NN->w0[32]= 0.00018710833683
NN->w0[33]= 0.00018666288815
NN->w0[34]= 0.00018710833683
NN->w0[35]= 0.00018710833683
NN->w1[0]= 0.00006805425073
NN->w1[1]= 0.00006802870485
NN->w1[2]= 0.00006805425073
NN->w1[3]= 0.00006805425073
NN->w1[4]= 0.00006805425073
NN->w1[5]= 0.00006802904682
NN->w1[6]= 0.00006805366866
NN->w1[7]= 0.00006805425073
NN->w1[8]= 0.00006803462020
NN->w1[9]= 0.00006805425073
NN->w1[10]= 0.00006805422163
NN->w1[11]= 0.00006805104204
NN->w1[12]= 0.00006805425073
NN->w1[13]= 0.00006805425073
NN->w1[14]= 0.00006805412704
NN->w1[15]= 0.00006802455027
NN->w1[16]= 0.00006805425073
NN->w1[17]= 0.00006785405276
NN->w1[18]= 0.00006805425073
NN->w1[19]= 0.00006805425073
NN->w1[20]= 0.00006805425073
NN->w1[21]= 0.00006805425073
NN->w1[22]= 0.00006805425073
NN->w1[23]= 0.00006804876466
NN->w1[24]= 0.00006805305020
NN->w1[25]= 0.00006805425073
NN->w1[26]= 0.00006805425073
NN->w1[27]= 0.00006781557749
NN->w1[28]= 0.00006799328548
NN->w1[29]= 0.00006805425073
NN->w1[30]= 0.00006805403973
NN->w1[31]= 0.00006805425073
NN->w1[32]= 0.00006804351870
NN->w1[33]= 0.00006804947043
NN->w1[34]= 0.00006805425073
NN->w1[35]= 0.00006805425073
NN->w2[0]= -0.00001783465996
```

```
NN->w2[1]= -0.00005640572635
NN->w2[2]= -0.00003650068538
NN->w2[3]= -0.00002339928142
NN->w2[4]= -0.00004964816981
NN->w2[5]= -0.00003809098416
NN->w2[6]= -0.00004511027146
NN->w2[7]= -0.00001357483870
NN->w2[8]= -0.00003752015982
NN->w2[9]= -0.00003086912329
NN->w2[10]= -0.00005522031279
NN->w2[11]= -0.00004232534411
NN->w2[12]= -0.00003085109347
NN->w2[13]= -0.00005764279194
NN->w2[14]= -0.00003580156408
NN->w2[15]= -0.00004449248081
NN->w2[16]= -0.00003231232404
NN->w2[17]= -0.00002435381066
NN->w2[18]= -0.00005355494795
NN->w2[19]= -0.00005536790559
NN->w2[20]= -0.00004131783862
NN->w2[21]= -0.00003770003241
NN->w2[22]= -0.00002361704901
NN->w2[23]= -0.00001183129734
NN->w2[24]= -0.00001038685787
NN->w2[25]= -0.00004556606655
NN->w2[26]= -0.00005648361184
NN->w2[27]= -0.00006447029591
NN->w2[28]= -0.00005542119106
NN->w2[29]= -0.00004130167144
NN->w2[30]= -0.00005753606456
NN->w2[31]= -0.00005057143790
NN->w2[32]= -0.00006591995771
NN->w2[33]= -0.00005503460125
NN->w2[34]= -0.00006509398372
NN->w2[35]= -0.00008052986959
```

Index

Note: Page numbers in *italics* refer to illustrations.

A

- Abdallah, C.T., 20
Abu-Khalaf, M., 474
Actuator nonlinearities, uncertain
nonlinear discrete-time systems
with, NN control, 265–341
background, 266–274
backlash, 272–273
deadzone, 269–272
friction, 266–269
nonlinear system with unknown
backlash, adaptive NN control
of, 309–319
saturation, 273–274
Adaptive critic NN architecture, 70
Adaptive critic NN controller
performance, 281, 383, 388
structure with input constraints, 290
Adaptive NN control design using state
measurements, 374–392
adaptive critic NN controller design,
381–392
adaptive NN backstepping controller
design, 375–378
critic NN design, 382–383
for general nonlinear systems in
nonstrict feedback form,
387–388
for SI engines, 388–392
tracking error-based adaptive NN
controller, 374–381
weight updates, 378–381
weight-tuning algorithms, 383–392
Adaptive NN controller design with
saturation nonlinearity, 287–296
adaptive NN controller structure with
saturation, 288
auxiliary system design, 287–288
closed-loop system stability analysis,
288–296

- Adaptive NN output feedback
controller, 357
performance, 405
structure, 401
Adjoint (backpropagation) neural
network, 55
Angelli, D., 309–310
Annaswamy, A.M., 274, 276
Armstrong-Helouvry, B., 267
Åström, K.J., 83, 103–104, 197
Asymptotic stability, of dynamical
systems, 84

B

- Backlash compensation using dynamic
inversion, 312–319
structure, 314–315, 318–319
Backlash dynamical system, 272–273
Backlash nonlinearity, 272–274
Backpropagation tuning, 47–67
derivation, 51–63
using sigmoid activation functions, 53
Balakrishnan, S.N., 275, 297
Barron, A.R., 32
Beard, R.W., 474, 486
Becker, K.H., 20
Boundedness
of dynamical systems, 84–86
extensions, 99–102
Brunovsky canonical form, 75–77
Brynes, C.I., 310, 474, 477

C

- Campos, J., 298
Canudas de Wit, C., 268

Certainty equivalence (CE) principle, 103
 CFR engine, 515, 519
 Chen, F.-C., 142, 344, 372
 Chen, H., 103
 Chen, Z., 475
 Closed-loop system stability analysis, 288–296
 CMAC NN (cerebellar model articulation controller networks), 13–15
 Commuri, S., 147
 Credit assignment problem, 48
 Cybenko, G., 31–32

D

Davis, Jr., 524
 Daw, C.S., 537
 Deadzone nonlinearity, 270–271, 269–272, 300–301
 compensation, 301–303
 saturation nonlinearities, 303
 Delta rule, 222
 Dendrites, 2
 Discrete-time adaptive control, 75–137, 384
 control design, 373
 mathematical background, 79–83
 nonlinear stability analysis and controls design, 88–102
 structure, 207
 two-layer NN controller using tracking error notion, 379
 using multilayer NN, 234, 238
 using one-layer NN, 150, 161, 224
 using three-layer NN, 193, 212
 with improved weight tuning, 188–191
 Discrete-time dynamical NN, 20
 phase-plane plot of, 22–24
 Discrete-time Hopfield network in block diagram form, 17
 Discrete-time model reference adaptive control, 447–471
 m th-order MIMO system dynamics, 448–450
 NN controller design, 451–460
 NN controller structure and error system dynamics, 451–454
 projection algorithm, 460–468
 weight updates for, 454–460
 Discrete-Time NN
 output feedback controller, 401

system identification using, 423–445
 Dörfler, M., 20
 Drallmeier, J.A., 524, 547, 549
 Dynamical systems, 15–24, 75–79
 asymptotic stability, 84
 boundedness, 84–86
 linear systems, 77–79
 Lyapunov stability, 84
 passivity, 86–87
 properties, 83–88
 stability, 83–86, 106–110

E

EGR engine controller design and implementation, 547–563
 adaptive output feedback EGR controller design, 553–559
 bifurcation diagram, 547
 engine dynamics, 549–551
 error dynamics, 554–557
 NN observer design, 551–553
 numerical simulation, 559–563
 weight updates, 557–559
 Embedded hardware implementation, and NN output feedback controller design, 511–566
 EGR engine controller design and implementation, 547–563
 embedded hardware-PC real-time digital control system, 512–514
 hardware description, 512–514
 lean engine controller design and implementation, 523–526
 SI engine test bed, 514–522
 software description, 514
 Engine-PC interface, 517
 Epoch vs. batch updating, in NN weight training, 42–47
 Equivalence ratio error, 391

F

Feedback interconnection, 88
 Feedback linearization, 197–200
 controller design, 199–200, 199
 error dynamics, 198
 input-output feedback linearization controllers, 197–198
 NN feedback linearization, 200–233

Feldkamp, L.A., 275

Finlayson, B.A., 494

Friction, 266–269

dynamic friction models, 268–269

static friction models, 267–268, 267

Functional link NN, 32

G

Galan, G., 269

Gaussian radial basis function networks,
see RBFs

Gauss–Newton algorithm, 55

Generalized recurrent NN, 19–24

GHJB-based control, 493, 507

Goodwin, C.G., 64, 83

Gradient descent tuning, 39–42

improvements, 63–67

Grundelius, M., 309–310

Guo, L., 103

H

Haykin, S., 1, 35, 64

He, P., 200, 277, 298, 344, 372,
393–394, 524

Hebb, D.O., 67

Hebbian tuning, 67–69

HJB (Hamilton–Jacobi–Bellman)

formulation, NN control in
discrete-time using, 473–510

NN least-squares approach, 486–490

numerical examples,
490–510

optimal control and generalized HJB

equation in discrete-time,
475–486

Hopfield network, 15–19, 15–19

Horne, B.G., 1, 35

Hornik, K., 31

Hush, D.R., 1, 35

I

Igelnik, B., 34, 283, 302, 308, 346, 529

Inoue, T., 524

J

Jagannathan, S., 104, 112, 142, 160–161,

177, 185, 195, 200, 223–224, 256,

269, 277, 286, 298, 310, 317, 344,

372–373, 381, 386, 393–394, 448,

451, 461, 475, 524

K

Kanellakopoulos, I., 103

Karason, S.P., 274, 276

Khalil, H.K., 142, 344, 372

Kleinman, D., 474

Kokotovic, P.V., 270, 274, 298, 310,
344, 373

Kosko, B., 1

Kumar, P.R., 103

Kung, S.Y., 1, 35, 64

L

Lean engine controller design and

implementation, 523–526

adaptive NN backstepping design,
531–534

adaptive NN output feedback controller
design, 530–537

engine dynamics, 526–528

experimental results, 539–546

NN controller C implementation
simulation, 537–539

NN observer design, 528–530

runtimes with number of neurons, 522

weight updates, 535–537

Least-squares NN output error, 60

vs. epoch, 49

Levenberg–Marquardt algorithm,
55–59, 64

Levine, D.S., 1

Lewis, F.L., 1–2, 20, 52, 82, 96, 104, 142,
147, 160–161, 177, 185, 195,

200–201, 223, 286, 298, 303, 310,

317, 373, 380–381, 448, 451, 454,
461, 474

Li, W., 197

Lin, W., 310, 474, 477

Lin, X., 275, 297

Linear discrete-time system, 491–494
 Linear-in-the parameter NN, 12–15
 LIP (linear in the unknown parameter),
 139–142
 Lippmann, R.P., 1
 Luenberger, D.G., 75, 265
 Lyapunov stability/analysis, 103–104
 for autonomous systems, 88–92
 controller design using, 92–96
 and controls design for linear systems,
 94–95
 of dynamical systems, 84
 extensions, 99–102
 for linear systems, 95–96
 of LTI feedback controllers, 96
 for nonautonomous systems, 97–99
 stability, 84, 89–92
 Lyshevski, S.E., 474

M

Mathematical background, in discrete-time adaptive control, 79–83
 continuity and function norms, 82–83
 quadratic forms and definiteness, 81–82
 singular value decomposition, 80–81
 vector and matrix norms, 79–82
 Miller, W.T., 474
 Miller, W.T. III., 69, 298
 MIMO (multi-input and multi-output system), 105
 dynamics of nonlinear MIMO system,
 107–109
 MIMO discrete-time systems, output
 feedback control, 343–370
 MIMO systems identifier dynamics,
 426–429
 mth order MIMO discrete-time
 nonlinear system dynamics,
 109–110, 143–145
 mth-order MIMO system dynamics,
 448–450
 Model reference adaptive controller for
 nonlinear discrete-time systems,
 461–468
 Momentum gradient algorithm, 64–65
 adaptive learning rate, 65–66
 safe learning rate, 66–67
 MRAC (model reference adaptive control),
 103, 447–448
 of nonlinear systems, 454–460

Multilayer NN, for feedback linearization,
 233–354
 weight updates not requiring PE,
 236–254
 weight updates requiring PE, 234–236
 Multilayer NN controller design, 33,
 167–191
 error dynamics and NN controller
 structure, 170–171
 identifier models, 426–427, 432, 440
 multilayer NN weight updates, 172–179
 PE condition relaxation, multilayer NN
 weight-tuning modification for,
 185–191
 projection algorithm, 179–184
 structure, 171, 180, 182–184, 187
 Multilayer NN training, 47–67
 background, 49–50
 Munos, R., 474
 Murray, J.J., 275

N

Narendra, K.S., 2, 15, 54, 142, 447
 Negative viscous friction, 268
 Neuron
 anatomy, 2
 mathematical model, 3, 3–8
 NN (neural networks)
 approximation property of n-layer NN,
 32–33
 background, 1–70
 multilayer perception, 8–12
 NN learning and control architectures,
 69–71
 NN topologies and recall, 2–24
 NN weight selection and training, 35–69
 number of hidden-layer neurons, 34
 properties, 24–35
 NN closed-loop system
 using an n-layer NN, 191
 using a one-layer NN, 255
 NN control
 controller performance, 292
 controller structure, 453
 controller with improved weight-tuning
 and projection algorithm, 253
 controller with unknown input
 deadzones and magnitude
 constraints, 301

- controller without saturation
nonlinearity, 283–287
- controller, heat release with, 393
- with deadzone compensator, 309
- with discrete-time tuning, 142–197
- in discrete-time using
Hamilton–Jacobi–Bellman
formulation, 473–510
- design, 146–147
- feedback linearization, 197–200
- of nonlinear systems and feedback
linearization, 139–264
- of nonstrict feedback nonlinear systems,
371–422
- of uncertain nonlinear discrete-time
systems with actuator
nonlinearities, 265–341
- vs. standard adaptive control, 140–142
- NN decision boundaries, 48
- NN feedback linearization, 200–233
- controller design for, 204–211
- controller design, 210–211
- error system dynamics, 206–209
- multilayer NN for, 233–354
- NN approximation of unknown
functions, 204–206
- one-layer NN for, 211–233
- system dynamics and tracking problem,
201–204
- well-defined control problem, 209–210
- NN identification of discrete-time
nonlinear systems, 442–443
- NN identifier design, 429–439
- structure, 430–432
- multilayer NN weight updates, 432–439
- NN learning and control architectures,
69–71
- unsupervised and reinforcement
learning, 69–70
- NN output feedback controller design, and
embedded hardware
implementation, 511–566
- NN output layer weights, 293
- NN passivity, 191–197, 254–261
- of closed-loop system, 195–196
- of multilayer NN, 196–197
- multilayer NN controllers passivity
properties, 256–261
- one-layer NN controllers passivity
properties, 256
- one-layer NN, passivity properties,
192–195
- properties, 439–443
- tracking error system passivity
properties, 191–192, 255
- NN properties, 24–35
- association, 28–31
- classification, 25–28
- classification and association, 25–31
- function approximation, 31–35
- NN weight selection and training, 35–69
- direct computation vs. training, 35
- learning and operational phases, 36
- learning schemes classification, 35–36
- off-line vs. online learning, 36
- Nonlinear discrete-time systems, in
nonstrict feedback nonlinear
systems, 371–373
- backstepping design, 373–374
- Nonlinear dynamical systems, 425–426
- and feedback linearization, NN control
of, 139–264
- Nonlinear stability analysis and controls
design, 88–102
- Nonlinear system with unknown backlash,
adaptive NN control of, 309–319
- backlash compensation using dynamic
inversion, 312–319
- controller design using filtered tracking
error without backlash
nonlinearity, 311–312
- description, 310–311
- Nonstrict feedback nonlinear systems,
NN control of, 371–422
- adaptive NN control design using state
measurements, 374–392
- nonlinear discrete-time systems in,
371–373
- output feed back NN controller design,
392–406

O

- One-layer NN, 5
- decision region, 26
- output surface, 7
- One-layer NN controller design, 145–167
- action NN, 281–282
- controller structure, 147
- critic NN, 280–281
- NN and error system dynamics
structure, 147–148

One-layer NN controller design
(continued)
 NN weight updates for guaranteed tracking performance, 148–155
 no disturbances and no NN reconstruction errors, 156–160
 PE condition relaxation, parameter tuning modification for, 160–167
 projection algorithm, 155–156
 strategic utility function, 279–280
 One-layer NN training, gradient descent, 38–47
 epoch vs. batch updating, 42–47
 gradient descent tuning, 39–42
 matrix formulation, 41–42
 One-layer NN, for feedback linearization, 211–233
 projection algorithm, 222–223
 weight updates not requiring PE, 223–233
 weight updates requiring PE, 211–222
 Output error plots vs. weights for a neuron, 30
 Output feedback control, of MIMO discrete-time systems, 343–370
 auxiliary controller design, 348–349
 controller design with magnitude constraints, 349–350
 design, 345–350
 NN controller design, 347–350
 nonlinear discrete-time systems class, 345
 observer design, 346–347
 weight updates for guaranteed performance, 350–361
 Output feedback NN controller design, 392–406
 adaptive NN controller design, 396–400
 of discrete-time nonlinear system, 404–406
 NN observer design, 394–396
 virtual controller design, 397–398
 weight updates for, 400–406

interconnections, 87–88
 PC770 embedded computer, 513
 Peretto, P., 1, 35, 64
 PID (proportional, integral, and derivative) control algorithms, 139
 Prokhorov, D.V., 275

R

Random vector functional link networks, 34
 RBFs (radial basis functions), 12–13, 14
 Recker, P., 298
 Reinforcement and unsupervised learning methods, 69–70
 comparison, 70–71
 Reinforcement learning, 298–299
 Reinforcement learning NN Control without magnitude constraints, 285, 291
 Reinforcement learning NN controller design, 304–309
 critic NN design, 305–306
 error dynamics, 304–305
 for nonlinear systems with deadzones, 307
 Reinforcement NN learning control, with saturation, 274–297
 filtered tracking error, control design based on, 277–279
 nonlinear system description, 276–277
 one-layer NN controller design, 279–283
 RLNN (reinforcement learning-based neural network), 299
 Robust implicit STR, 102–128
 adaptive control formulation, 105–106
 background, 104–110
 no disturbances and no STR reconstruction errors, 117–119
 parameter-tuning modification for PE condition relaxation, 119–123
 projection algorithm, 116–117

P

Pao, Y.H., 34, 283, 302, 308, 346, 529
 Parisini, T., 474
 Park, C., 474
 Parthasarathy, K., 54, 142, 447
 Passivity, of dynamical systems, 86–87

S

Sadegh, N., 145
 Sandberg, I.W., 31
 Sanner, R.M., 35, 211
 Saturation, 273–274

adaptive NN controller design with saturation nonlinearity, 287–296
NN controller without, 283–287
reinforcement NN learning control with, 274–297
Selmic, R.R., 298, 303
Shaft encoder, 515
Shervais, S., 275
Si, J., 275, 297
SI engine test bed, 514–522
controller timing specifications, 520–521
engine–PC interface hardware operation, 516–517
PC operation, 518–520
software implementation, 521–522
Sin, K.S., 64, 83
Slotine, J.-J.E., 35, 197, 211
Sofage, D.A., 69, 298
Standard adaptive control vs. NN control, 140–142
STR (self tuning regulator)
design, 111–116
passivity properties, 123–127
STR parameter updates, 112–116
structure and error system dynamics, 111–112
Stribeck effect, 268
Supervised learning, 298–299
Sutton, R.W., 524, 547, 549
Syrmos, V.L., 96
System identification, using discrete-time neural networks, 423–445
MIMO systems identifier dynamics, 426–429
NN identifier design, 429–439
nonlinear dynamical systems, 425–426

T

Tao, G., 270, 274, 298, 310
Three-Layer NN
identifier, 432–439
three-layer NN passivity using tuning algorithms, 440–442
Tian, M., 298
Tracking error
and reinforcement learning-based controls design, comparison, 296–297

tracking error-based adaptive NN controller, 374–381
Tsiotras, P., 474
Two-layer neural network, 8
output surface, 11
and its samples for training, 56
Two-Link Planar RR Robot Arm System, 498–510, 499

U

Uncertain nonlinear discrete-time systems with actuator nonlinearities, NN control of, 265–341
Uncertain nonlinear system, with unknown deadzone and saturation nonlinearities, 297–309
deadzone compensation with magnitude constraints in, 300–303
deadzone nonlinearity, 300–301
description and error dynamics, 300
reinforcement learning NN controller design, 304–309
Unsupervised and reinforcement learning methods, 69–70
comparison, 70–71
Unsupervised learning, 298–299
UUB (uniformly ultimately bounded) point, 84
illustration, 85
by Lyapunov analysis, 99–102

V

Vance, J., 512, 526
von Bertalanffy, L., 265

W

Wang, Y.T., 275, 297
Weight computation, 36–38
Weight updates, in output feedback control of MIMO discrete-time systems, 350–361
critic NN design, 351–353
strategic utility function, 351
weight updating rule for observer NN, 350–351

Weight updates, in output feedback control
of MIMO discrete-time systems
(continued)
weight updating rule for the action NN,
353–361
Weiner, N., 54
Werbos, P.J., 142
White, D.A., 69, 298
Whitehead, A.N., 265
Widrow, B., 38, 41, 64
Widrow–Hoff rule, 222
Wittenmark, B., 83, 104

Y

Yeh, P.C., 310, 344
Yesildirek, A., 201

Z

Zhang, T., 144
Zoppoli, R., 474