Studies in Computational Intelligence 112

Edgar N. Sanchez
Alma Y. Alanis
Alexander G. Loukianov

Discrete-Time High Order Neural Control

Trained with Kalman Filtering

Springer

Studies in Computational Intelligence, Volume 112

Editor-in-chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 91. Horst Bunke, Kandel Abraham and Last Mark (Eds.)
Applied Pattern Recognition, 2008
ISBN 978-3-540-76830-2

Vol. 92. Ang Yang, Yin Shan and Lam Thu Bui (Eds.)
Success in Evolutionary Computation, 2008
ISBN 978-3-540-76285-0

Vol. 93. Manolis Wallace, Marios Angelides and Phivos Mylonas (Eds.)
Advances in Semantic Media Adaptation and Personalization, 2008
ISBN 978-3-540-76359-8

Vol. 94. Arpad Kelemen, Ajith Abraham and Yuehui Chen (Eds.)
Computational Intelligence in Bioinformatics, 2008
ISBN 978-3-540-76802-9

Vol. 95. Radu Dogaru
Systematic Design for Emergence in Cellular Nonlinear Networks, 2008
ISBN 978-3-540-76800-5

Vol. 96. Aboul-Ella Hassanien, Ajith Abraham and Janusz Kacprzyk (Eds.)
Computational Intelligence in Multimedia Processing: Recent Advances, 2008
ISBN 978-3-540-76826-5

Vol. 97. Gloria Phillips-Wren, Nikhil Ichalkaranje and Lakhmi C. Jain (Eds.)
Intelligent Decision Making: An AI-Based Approach, 2008
ISBN 978-3-540-76829-9

Vol. 98. Ashish Ghosh, Satchidananda Dehuri and Susmita Ghosh (Eds.)
Multi-Objective Evolutionary Algorithms for Knowledge Discovery from Databases, 2008
ISBN 978-3-540-77466-2

Vol. 99. George Meghabghab and Abraham Kandel
Search Engines, Link Analysis, and User's Web Behavior, 2008
ISBN 978-3-540-77468-6

Vol. 100. Anthony Brabazon and Michael O'Neill (Eds.)
Natural Computing in Computational Finance, 2008
ISBN 978-3-540-77476-1

Vol. 101. Michael Granitzer, Mathias Lux and Marc Spaniol (Eds.)
Multimedia Semantics - The Role of Metadata, 2008
ISBN 978-3-540-77472-3

Vol. 102. Carlos Cotta, Simeon Reich, Robert Schaefer and Antoni Ligeza (Eds.)
Knowledge-Driven Computing, 2008
ISBN 978-3-540-77474-7

Vol. 103. Devendra K. Chaturvedi
Soft Computing Techniques and its Applications in Electrical Engineering, 2008
ISBN 978-3-540-77480-8

Vol. 104. Maria Virvou and Lakhmi C. Jain (Eds.)
Intelligent Interactive Systems in Knowledge-Based Environment, 2008
ISBN 978-3-540-77470-9

Vol. 105. Wolfgang Guenthner
Enhancing Cognitive Assistance Systems with Inertial Measurement Units, 2008
ISBN 978-3-540-76996-5

Vol. 106. Jacqueline Jarvis, Dennis Jarvis, Ralph Rönnquist and Lakhmi C. Jain (Eds.)
Holonic Execution: A BDI Approach, 2008
ISBN 978-3-540-77478-5

Vol. 107. Margarita Sordo, Sachin Vaidya and Lakhmi C. Jain (Eds.)
Advanced Computational Intelligence Paradigms in Healthcare - 3, 2008
ISBN 978-3-540-77661-1

Vol. 108. Vito Trianni
Evolutionary Swarm Robotics, 2008
ISBN 978-3-540-77611-6

Vol. 109. Panagiotis Chountas, Ilias Petrounias and Janusz Kacprzyk (Eds.)
Intelligent Techniques and Tools for Novel System Architectures, 2008
ISBN 978-3-540-77621-5

Vol. 110. Makoto Yokoo, Takayuki Ito, Minjie Zhang, Juhnyoung Lee and Tokuro Matsuo (Eds.)
Electronic Commerce, 2008
ISBN 978-3-540-77808-0

Vol. 111. David Elmakias (Ed.)
New Computational Methods in Power System Reliability, 2008
ISBN 978-3-540-77810-3

Vol. 112. Edgar N. Sanchez, Alma Y. Alanís and Alexander G. Loukianov
Discrete-Time High Order Neural Control: Trained with Kalman Filtering, 2008
ISBN 978-3-540-78288-9

Edgar N. Sanchez
Alma Y. Alanís
Alexander G. Loukianov

Discrete-Time High Order Neural Control

Trained with Kalman Filtering

With 90 Figures and 4 Tables

Springer

Dr. Edgar N. Sanchez
CINVESTAV
Unidad Guadalajara
Apartado Postal 31-438
Plaza la Luna, Guadalajara
Jalisco C.P. 45091
México
sanchez@gdl.cinvestav.mx

Dr. Alexander G. Loukianov
CINVESTAV
Unidad Guadalajara
Apartado Postal 31-438
Plaza la Luna, Guadalajara
Jalisco C.P. 45091
México
louk@gdl.cinvestav.mx

Dr. Alma Y. Alanís
Departamento deciencias
computacionales
CUCEI
Universidad de Guadalajara
Blvd. Marcelino Garcia Barragan
#1421, esq. Calzada Olimpica
Modulo "O" Planta baja
Guadalajara, Jalisco, Mexico
C.P. 44430
alma.alanis@cucei.udg.mx

ISBN 978-3-540-78288-9 e-ISBN 978-3-540-78289-6

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2008922059

Cover design: Deblik, Berlin, Germany

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

The first author dedicates this book to his wife Maria de Lourdes Mejia and his children Zulia Mayari Sanchez, Ana Maria Sanchez, and Edgar Camilo Sanchez

The second author dedicates this book to her husband Gilberto Alvarez, her parents Alfonso Alanis and Yolanda Garcia, and her brother Alfonso Alanis

The third author dedicates this book to his wife Ludmila Loukianova and his son Nicolay Loukianov

Preface

Neural networks have become a well-established methodology as exemplified by their applications to identification and control of general nonlinear and complex systems; the use of high order neural networks for modeling and learning has recently increased.

Using neural networks, control algorithms can be developed to be robust to uncertainties and modeling errors. The most used NN structures are Feedforward networks and Recurrent networks. The latter type offers a better suited tool to model and control of nonlinear systems.

There exist different training algorithms for neural networks, which, however, normally encounter some technical problems such as local minima, slow learning, and high sensitivity to initial conditions, among others. As a viable alternative, new training algorithms, for example, those based on Kalman filtering, have been proposed.

There already exists publications about trajectory tracking using neural networks; however, most of those works were developed for continuous-time systems. On the other hand, while extensive literature is available for linear discrete-time control system, nonlinear discrete-time control design techniques have not been discussed to the same degree. Besides, discrete-time neural networks are better fitted for real-time implementations.

This book presents a solution for the trajectory tracking problem of unknown nonlinear systems based on four schemes. For the first one, a direct design method is consider, the well known backstepping one, under the assumption of the complete access to the state; the second one considers an indirect method, solved with the block control and the sliding mode techniques, under the same assumption. For the third scheme, the backstepping technique is reconsidered, including a neural observer; and finally the block control and the sliding mode techniques are used again, with a neural observer. All the proposed schemes are developed in discrete-time and include the respective stability analyses, using the Lyapunov approach, for each one of the proposed schemes.

To this end the real-time implementation for the schemes proposed in this book are presented, validating the theoretical results, using a three phase induction motor benchmark. The control of an induction motor is challenging, since its dynamics is described by multivariable, coupled, and highly nonlinear system; besides, it is one of the most used actuators for industrial applications due to its reliability, ruggedness, and relatively low cost.

The material presented in this book basically reports research results obtained during the previous three years at the Automatic Control Laboratory of CINVESTAV-IPN, Guadalajara campus. We thank CONACYT-Mexico for the financial support during this research through Projects 46069Y and 57801Y.

We are particulary grateful to a number of people for carefully reviewing draft material for this book, and for discussions which in one way or another have influenced parts of the text: Ofelia Begovich, Juan M. Ramirez, Eduardo Bayro, Marco Perez, Jagannathan Sarangapani, Guanrong Chen and Frank Lewis.

We also thank Juan Quiñones, Jorge Rivera, Adrian Navarro, Ernesto Amezcua, Cristobal Sanroman, Jose Jimenez, Victor Preciado, and Hector Becerra for their collaboration in order to implement the real time experiments.

Guadalajara, Jalisco, Mexico,
December, 2007

Edgar N. Sanchez
Alma Y. Alanis
Alexander G. Loukianov

Contents

1

Introduction

1.1 Preliminaries

The ultimate goal of control engineering is to implement an automatic system that could operate with increasing independence from human actions in an unstructured and uncertain environment. Such a system may be named autonomous or intelligent. It would need only to be presented with a goal and would achieve its objective by learning through continuous interaction with its environment through feedback about its behavior [13].

One class of models that has the capability to implement this learning is the artificial neural networks. Indeed, the neural morphology of the nervous system is quite complex to analyze. Nevertheless, simplified analogies have been developed, which could be used for engineering applications. Based on these simplified understandings, artificial neural networks are built [6].

An artificial neural network is a massively parallel distributed processor, inspired from biological neural networks, which can store experimental knowledge and makes it available for use. An artificial neural network consists of a finite number of neurons (structural element), which are interconnected to each other. It has some similarities with the brain, such as knowledge is acquired through a learning process and interneuron connectivity named as synaptic weights are used to store this knowledge, among others [13].

The research on neural networks, since its reborn in the early 1980s of the twentieth century, promotes a great interest principally due to the capability of static neural networks to approximate arbitrarily well any continuous function. Besides in recent years the use of recurrent neural networks has been increased; their process is described by differential equations for continuous time or by differential equations for discrete time [6].

Using neural networks, control algorithms can be developed to be robust to uncertainties and modeling errors. The most used neural network structures are *Feedforward* networks and *Recurrent* ones [1, 14]. The last type offers a better suited tool to model and control nonlinear systems [11].

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 1–4 (2008)
www.springerlink.com

There exist different training algorithms for neural networks, which, however, normally encounter some technical problems such as local minima, slow learning, and high sensitivity to initial conditions, among others. As a viable alternative, new training algorithms, for example, those based on Kalman filtering, have been proposed [5,6,15]. Because of the fact that training a neural network typically results in a nonlinear problem, the Extended Kalman filter (EKF) is a common tool to use, instead of a linear Kalman filter [6].

There already exists publications about trajectory tracking using neural networks [3, 7–12, 14]; in most of them, the design methodology is based on the Lyapunov approach. However, most of those works were developed for continuous-time systems. On the other hand, while extensive literature is available for linear discrete-time control system, nonlinear discrete-time control design techniques have not been discussed to the same degree. For nonlinear discrete-time systems, the control problem is more complex due to the couplings among subsystems, inputs, and outputs [2,4,8]. Besides, discrete-time neural networks are better fitted for real-time implementations.

1.2 Motivation

Taking into account the facts exposed above, it is obvious the necessity to design control algorithms for multiple input multiple output (MIMO) discrete-time nonlinear systems based on neural networks. These algorithms should be robust to external disturbances as well as parametric variations.

On the other hand, in most nonlinear control designs, it is usually assumed that all the system state are measurable. In practice, however, only part of this state is measured directly. For this reason, nonlinear state estimation remains an important topic for study on the nonlinear systems theory. For continuous time recurrent neural observers have also been proposed, which do not require a precise plant model. Nevertheless, the discrete-time case has not been dealt with the same intensity; thus is a field replete of opportunities for research and applications.

Therefore, the major motivation for this book is to develop alternative methodologies that allow the design of robust controllers for discrete-time nonlinear systems with unknown dynamics.

1.3 Objectives

The main objectives of this work are stated as follows:

- To synthesize a scheme for output trajectory tracking based on a high order neural network (HONN) structure trained with an EKF, to approximate a control law designed by the backstepping technique, for a class of MIMO discrete-time nonlinear systems.

- To synthesize a scheme for output trajectory tracking based on a recurrent high order neural network (RHONN) trained with an EKF, to identify using a class of MIMO discrete-time nonlinear systems and based on the neural model design a control law by the block control and sliding modes techniques.
- To synthesize a neural observer for a class of MIMO discrete-time nonlinear systems, using a training algorithm based on an EKF.
- To synthesize a scheme for output trajectory tracking based on a HONN structure trained with an EKF, to approximate a control law designed by the backstepping technique using neural observer for a class of MIMO discrete-time nonlinear systems.
- To synthesize a scheme for output trajectory tracking based on a neural observer trained with an EKF, to estimate using a class of MIMO discrete-time nonlinear systems and based on the neural model design a control law by the block control and sliding modes techniques, for a class of MIMO discrete-time nonlinear systems.
- To establish the stability analyses, using the Lyapunov approach, for each one of the proposed schemes.
- To implement the real time experiments for each one of the proposed schemes.

1.4 Book Structure

This book presents a solution for the trajectory tracking problem of unknown nonlinear systems based on four schemes. For the first one, a direct design method is considered: the well known backstepping one, under the assumption of the complete access to the state; the second one considers an indirect method, solved with the block control and the sliding mode techniques, under the same assumption. For the third scheme, the backstepping technique is reconsidered, including a neural observer; and finally the block control and the sliding mode techniques are used again too, with a neural observer. All the proposed schemes are developed in discrete-time.

This book is organized as follows.

In Chap. 2, some mathematical preliminaries are introduced, including stability definitions, the extended Kalman filter, and some foundations of neural control.

Then Chap. 3 presents a high order neural network (HONN) to solve the tracking problem for a class of MIMO discrete-time nonlinear systems, using the backstepping technique. The training of the neural network is performed online using an extended Kalman filter.

After that in Chap. 4 a recurrent high order neural network is used to identify a class of discrete-time nonlinear systems and the identified model is used to design a block control from controller. The training of the neural networks is performed online using an extended Kalman filter.

In Chap. 5, an RHONN is used to design a Luenberger-like observer for a class of MIMO discrete-time nonlinear systems. The RHONO proposed is trained with an EKF-based algorithm. The training of the RHONO is performed online in a parallel configuration.

In Chap. 6, two solutions for the discrete-time output trajectory tracking problem are proposed by means of the backstepping and block control techniques, both of them based on an RHONO. Based on the nonlinear observer as designed on the previous chapter, the two controllers are synthesized, respectively.

Chapter 7 includes experimental results, on real time, for the neural identifier, the neural observer, and the four control schemes developed in the previous chapters; all of them are applied to a three phase induction motor.

Finally some relevant conclusions and future work are stated.

1.5 Notation

Through this book, we use the following notations:

$k \in 0 \cup \mathbb{Z}^+$	Sampling step
$\lvert\bullet\rvert$	Absolute value
$\lVert\bullet\rVert$	Euclidian norm for vectors and any adequate norm for matrices
$S(\bullet)$	Sigmoid function
$x \in \Re^n$	Plant state
$\widehat{x} \in \Re^n$	Neural network state
$w_i \in \Re^L$	ith neural network estimated weight vector
$w_i^* \in \Re^L$	ith neural network ideal weight vector
$L_i \in \Re$	Number of high order connections
$u \in \Re^m$	Control action
$u^* \in \Re^m$	Ideal control action
$\varrho \in \Re^m$	Neural network external input
$z_i \in \Re^{L_i}$	High order terms
$K \in \Re^{L_i \times m}$	Kalman gain matrix
$P \in \Re^{L_i \times L_i}$	Associated prediction error covariance matrix
$Q \in \Re^{L_i \times L_i}$	Associated state noise covariance matrix
$R \in \Re^{m \times m}$	Associated measurement noise covariance matrix
$g_i \in \Re$	ith neural observer gain
$r \in \Re$	Number of blocks
$n_i \in \Re$	Dimension of the ith block
$S_D \in \Re^{n_r}$	Sliding manifold
$\mathbf{k}_i \in \Re$	Control gain of the ith block
$\mathbf{z}_i \in \Re^{n_i}$	State transformation of the ith block
$e \in \Re^p$	Output error
$\widetilde{x} \in \Re^n$	State observer error
$\widetilde{w}_i \in \Re^{L_i}$	Weights estimation error

2

Mathematical Preliminaries

In this chapter, important mathematical preliminaries, required in future chapters, are presented.

2.1 Stability Definitions

Consider an MIMO nonlinear system:

$$x(k+1) = F(x(k), u(k)) \tag{2.1}$$

$$y(k) = h(x(k)), \tag{2.2}$$

where $x \in \Re^n$, $u \in \Re^m$, and $F \in \Re^n \times \Re^m \to \Re^n$ is nonlinear function.

Definition 2.1. *The system* (2.1) *is said to be forced or to have input. In contrast, the system described by an equation without explicit presence of an input u, that is,*

$$x(k+1) = F(x(k))$$

is said to be unforced. It can be obtained after selecting the input u as a feedback function of the state

$$u(k) = \xi(x(k)). \tag{2.3}$$

Such substitution eliminates u:

$$x(k+1) = F(x(k), \xi(x(k))), \tag{2.4}$$

and yields an unforced system (2.4) [9].

Definition 2.2. *The solution of* (2.1)–(2.3) *is semiglobally uniformly ultimately bounded (SGUUB), if for any Ω, a compact subset of $\Re^n$ and all $x(k_0) \in \Omega$, there exists an $\varepsilon > 0$ and a number $N(\varepsilon, x(k_0))$ such that $\|x(k)\| < \varepsilon$ for all $k \geq k_0 + N$.*

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 5–10 (2008)
www.springerlink.com

In other words, the solution of (2.1) is said to be SGUUB if, for any a priori given (arbitrarily large) bounded set Ω and any a priori given (arbitrarily small) set Ω_0, which contains $(0,0)$ as an interior point, there exists a control (2.3) such that every trajectory of the closed loop system starting from Ω enters the set $\Omega_0 = \{x(k) | \|x(k)\| < \varepsilon\}$ in a finite time and remains in it thereafter.

Theorem 2.1. *Let $V(x(k))$ be a Lyapunov function for a discrete-time system (2.1), which satisfies the following properties:*

$$\begin{aligned} \gamma_1(\|x(k)\|) \leq V(x(k)) \leq \gamma_2(\|x(k)\|), \\ V(x(k+1)) - V(x(k)) = \Delta V(x(k)) \\ \leq -\gamma_3(\|x(k)\|) + \gamma_3(\zeta), \end{aligned}$$

where ζ is a positive constant, $\gamma_1(\bullet)$ and $\gamma_2(\bullet)$ are strictly increasing functions, and $\gamma_3(\bullet)$ is a continuous, nondecreasing function. Thus if

$$\Delta V(x) < 0 \quad for \quad \|x(k)\| > \zeta,$$

then $x(k)$ is uniformly ultimately bounded, i.e., there is a time instant k_{T}, such that $\|x(k)\| < \zeta, \forall k < k_{\mathrm{T}}$.

Definition 2.3. *A subset $S \in \Re^n$ is bounded if there exists $r > 0$ such that $\|x\| \leq r$ for all $x \in S$ [9].*

Theorem 2.2 (Separation Principle). [12]. *The asymptotic stabilization problem of the system (2.1)–(2.2), via estimated state feedback*

$$\begin{aligned} u(k) &= \xi(\widehat{x}(k)), \\ \widehat{x}(k+1) &= F(\widehat{x}(k), u(k), y(k)) \end{aligned} \tag{2.5}$$

is solvable if and only if the system (2.1)–(2.2) is asymptotically stabilizable and exponentially detectable.

Corollary 2.1. [12]. *There is an exponential observer for a Lyapunov stable discrete-time nonlinear system (2.1)–(2.2) with $u = 0$ if and only if the linear approximation*

$$\begin{aligned} x(k+1) &= A(k)x(k) + Bu(k), \\ y(k) &= Cx(k), \end{aligned} \tag{2.6}$$

$$A = \left.\frac{\partial F}{\partial x}\right|_{x=0}, \quad B = \left.\frac{\partial F}{\partial u}\right|_{x=0}, \quad C = \left.\frac{\partial h}{\partial x}\right|_{x=0}$$

of the system (2.1)–(2.2) is detectable.

2.2 Discrete-Time High Order Neural Networks

The use of multilayer neural networks is well known for pattern recognition and for modeling of static systems. The NN is trained to learn an input–output map. Theoretical works have proven that, even with just one hidden layer, a NN can uniformly approximate any continuous function over a compact domain, provided that the NN has a sufficient number of synaptic connections.

For control tasks, extensions of the first-order Hopfield model called recurrent high order neural networks (RHONN), which present more interactions among the neurons, are proposed in [13, 16]. Additionally, the RHONN model is very flexible and allows to incorporate to the neural model a priori information about the system structure.

Consider the following discrete-time RHONN:

$$\widehat{x}_i(k+1) = w_i^\top z_i(\widehat{x}(k), \upsilon(k)), \quad i = 1, \cdots, n, \tag{2.7}$$

where $\widehat{x}_i$ $(i = 1, 2, \cdots, n)$ is the state of the ith neuron, L_i is the respective number of high order connections, $\{I_1, I_2, \cdots, I_{L_i}\}$ is a collection of nonordered subsets of $\{1, 2, \cdots, n+m\}$, n is the state dimension, m is the number of external inputs, w_i $(i = 1, 2, \cdots, n)$ is the respective online adapted weight vector, and $z_i(\widehat{x}(k), \varrho(k))$ is given by

$$z_i(x(k), \varrho(k)) = \begin{bmatrix} z_{i_1} \\ z_{i_2} \\ \vdots \\ z_{i_{L_i}} \end{bmatrix} = \begin{bmatrix} \prod_{j \in I_1} \xi_{i_j}^{d_{ij}(1)} \\ \prod_{j \in I_2} \xi_{i_j}^{d_{ij}(2)} \\ \vdots \\ \prod_{j \in I_{L_i}} \xi_{i_j}^{d_{ij}(L_i)} \end{bmatrix}, \tag{2.8}$$

with $d_{j_i}(k)$ being a nonnegative integers, and ξ_i defined as follows:

$$\xi_i = \begin{bmatrix} \xi_{i_1} \\ \vdots \\ \xi_{i_1} \\ \xi_{i_{n+1}} \\ \vdots \\ \xi_{i_{n+m}} \end{bmatrix} = \begin{bmatrix} S(x_1) \\ \vdots \\ S(x_n) \\ \varrho_1 \\ \vdots \\ \varrho_m \end{bmatrix}. \tag{2.9}$$

In (2.9), $\varrho = [\varrho_1, \varrho_2, \ldots, \varrho_m]^\top$ is the input vector to the neural network, and $S(\bullet)$ is defined by

$$S(\varsigma) = \frac{1}{1 + \exp(-\beta\varsigma)}, \quad \beta > 0, \tag{2.10}$$

where ς is any real value variable.

Consider the problem to approximate the general discrete-time nonlinear system (2.1), by the following discrete-time RHONN series–parallel representation [16]:

$$x_i\,(k+1) = w_i^{*\top} z_i(x(k), \varrho(k)) + \epsilon_{z_i}, \quad i = 1, \cdots, n, \tag{2.11}$$

where x_i is the ith plant state, ϵ_{z_i} is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [16]. Assume that there exists ideal weight vector w_i^* such that $\|\epsilon_{z_i}\|$ can be minimized on a compact set $\Omega_{z_i} \subset \Re^{L_i}$. The ideal weight vector w_i^* is an artificial quantity required for analytical purpose [16]. In general, it is assumed that this vector exists and is constant but unknown. Let us define its estimate as w_i and the estimation error as

$$\widetilde{w}_i(k) = w_i^* - w_i(k)\,. \tag{2.12}$$

The estimate w_i is used for stability analysis, which will be discussed later. Since w_i^* is constant, then $\widetilde{w}_i(k+1) - \widetilde{w}_i(k) = w_i(k) - w_i(k+1), \forall k \in 0 \cup \mathbb{Z}^+$.

From (2.7) three possible models can be derived:

- Parallel model

$$\widehat{x}_i(k+1) = w_i^\top z_i(\widehat{x}(k), \varrho(k)), \quad i = 1, \cdots, n \tag{2.13}$$

- Series–Parallel model

$$\widehat{x}_i(k+1) = w_i^\top z_i(x(k), \varrho(k)), \quad i = 1, \cdots, n \tag{2.14}$$

- Feedforward model (HONN)

$$\widehat{x}_i(k) = w_i^\top z_i(\varrho(k)), \quad i = 1, \cdots, n, \tag{2.15}$$

  where $\widehat{x}$ is the NN state vector, x is the plant state vector, and ϱ is the input vector to NN.

2.3 The EKF Training Algorithm

The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time learning [21]. However, it is a first order gradient descent method and hence its learning speed could be very slow [10]. Recently, extended Kalman filter (EKF) based algorithms have been introduced to train neural networks [1,3]. With the EKF based algorithm, the learning convergence is improved [10]. The EKF training of neural networks, both feedforward and recurrent ones, has proven to be reliable and practical for many applications over the past 10 years [3].

It is known that Kalman filtering (KF) estimates the state of a linear system with additive state and output white noises [7,20]. For KF-based neural

network training, the network weights become the states to be estimated. In this case, the error between the neural network output and the measured plant output can be considered as additive white noise. Because of the fact that the neural network mapping is nonlinear, an EKF-type is required (see [18] and references therein).

The training goal is to find the optimal weight values, which minimize the prediction error. The EKF-based training algorithm is described by [7]

$$\begin{aligned} K_i(k) &= P_i(k)H_i(k)\left[R_i(k) + H_i^\top(k)P_i(k)H_i(k)\right]^{-1}, \\ w_i(k+1) &= w_i(k) + \eta_i K_i(k)\left[y(k) - \widehat{y}(k)\right], \\ P_i(k+1) &= P_i(k) - K_i(k)\, H_i^\top(k)P_i(k) + Q_i(k), \end{aligned} \tag{2.16}$$

where $P_i \in \Re^{L_i \times L_i}$ is the prediction error associated covariance matrix, $w_i \in \Re^{L_i}$ is the weight (state) vector, L_i is the total number of neural network weights, $y \in \Re^m$ is the measured output vector, $\widehat{y} \in \Re^m$ is the network output, η_i is a design parameter, $K_i \in \Re^{L_i \times m}$ is the Kalman gain matrix, $Q_i \in \Re^{L_i \times L_i}$ is the state noise associated covariance matrix, $R_i \in \Re^{m \times m}$ is the measurement noise associated covariance matrix, $H_i \in \Re^{L_i \times m}$ is a matrix for which each entry $(H_{_{ij}})$ is the derivative of one of the neural network output, $(\widehat{y})$, with respect to one neural network weight, $(w_{_{ij}})$, as follows:

$$H_{_{ij}}(k) = \left[\frac{\partial \widehat{y}(k)}{\partial w_{_{ij}}(k)}\right]_{w_i(k)=\widehat{w}_i(k+1)}, \quad i = 1, \ldots, n \text{ and } \; j = 1, \ldots, L_i. \tag{2.17}$$

Usually P_i, Q_i, and R_i are initialized as diagonal matrices, with entries $P_i(0)$, $Q_i(0)$, and $R_i(0)$, respectively. It is important to note that $H_i(k)$, $K_i(k)$, and $P_i(k)$ for the EKF are bounded [20]. Therefore, there exist constants $\overline{H_i} > 0$, $\overline{K_i} > 0$, and $\overline{P_i} > 0$ such that

$$\begin{aligned} \|H_i(k)\| &\leq \overline{H_i}, \\ \|K_i(k)\| &\leq \overline{K_i}, \\ \|P_i(k)\| &\leq \overline{P_i}. \end{aligned} \tag{2.18}$$

Comment 2.1. The measurement and process noises are typically characterized as zero-mean, white noises with covariances given by $\delta_{k,j}R_i(k)$ and $\delta_{k,j}Q_i(k)$, respectively, with $\delta_{k,j}$ a Kronecker delta function (cero for $k \neq l$ and 1 for $k = l$) [8]. To simplify the notation in this book, the covariances will be represented by their respective associated matrices, $R_i(k)$ and $Q_i(k)$ for the noises and $P_i(k)$ for the prediction error.

2.4 Neural Control

To control a system is to force it to behave in a desired way. How to express this "desired behavior" depends primarily on the task to be solved, but the dynamics of the system, the actuators, the measurement equipment, the available

computational power, etc. influence the formulation of the desired behavior as well. Although the desired behavior obviously is very dependent on the application, the need to express it in mathematical terms suited for practical design of control systems seriously limits the means of expression. At the higher level, it is customary to distinguish two basic types of problems [14]:

Regulation problem. The fundamental desired behavior is to keep the output of the system at a constant level regardless of the disturbances acting on the system.

Tracking problem. The fundamental desired behavior is to force the system output to track a reference trajectory closely.

Neural networks (NN) have become a well-established methodology as exemplified by their applications to identification and control of general nonlinear and complex systems [6]; the use of high order neural networks for modeling and learning has recently increased [19]. Specifically, the problem of designing robust neural controllers for nonlinear systems with uncertainties and disturbances, which guarantees stability and trajectory tracking, has received an increasing attention lately.

Using neural networks, control algorithms can be developed to be robust to uncertainties and modeling errors. The most used NN structures are *Feedforward* networks and *Recurrent* ones [19]. The last type offers a better suited tool to model and control nonlinear systems [15].

The neural control problem can be approached in two different ways:

Direct control system design. "Direct" means that the controller is a neural network. A neural network controller is often advantageous when the real-time platform available prohibits complicated solutions. The implementation is simple while the design and tuning are difficult. With a few exceptions this class of designs is model-based in the sense that a model of the system is required in order to design the controller.

Indirect control system design. This class of designs is always model-based. The idea is to use a neural network to model the system to be controlled; this model is then employed in a more "conventional" controller design. The model is typically trained in advanced, but the controller is designed online. As it will appear, the indirect design is very flexible; thus it is the most appropriate for most of the common control problems.

The increasing use of NN to modeling and control is in great part due to the following features that makes them particularly attractive [4]:

- NN are universal approximators. It has been proven that any continuous nonlinear function can be approximated arbitrarily well over a compact set by a multilayer neural network, which consist of one or more hidden layers [2].
- Learning and adaptation. The intelligence of neural networks comes from their generalization ability with respect to unknown data. Online adaptation of the weights is possible.

3

Discrete-Time Adaptive Neural Backstepping

This chapter deals with adaptive tracking for a class of MIMO discrete-time nonlinear systems in presence of bounded disturbances. In this chapter, a high order neural network structure is used to approximate a control law designed by the backstepping technique, applied to a block strict feedback form (BSFF). It also presents the respective stability analysis, on the basis of the Lyapunov approach, for the whole scheme including the extended Kalman filter (EKF)-based NN learning algorithm. Applicability of this scheme is illustrated via simulation for a discrete-time nonlinear model of an electric induction motor.

In recent adaptive and robust control literature, numerous approaches have been proposed for the design of nonlinear control systems. Among these, adaptive backstepping constitutes a major design methodology [6, 9]. The idea behind backstepping design is that some appropriate functions of state variables are selected recursively as virtual control inputs for lower dimension subsystems of the overall system [12]. Each backstepping stage results in a new virtual control designs from the preceding design stages. When the procedure ends, a feedback design for the true control input results, which achieves the original design objective. The backstepping technique provides a systematic framework for the design of tracking and regulation strategies, suitable for a large class of state feedback linearizable nonlinear systems [1, 9–11].

3.1 Neural Backstepping Controller Design

The model of many practical nonlinear systems can be expressed in (or transformed into) a special state-space form named block strict feedback form (BSFF) [9] as follows:

$$\begin{aligned} x^i(k+1) &= f^i(\overline{x}^i(k)) + g^i(\overline{x}^i(k))x^{i+1}(k) + d^i(k), \qquad i = 1, 2, \ldots, r-1, \\ x^r(k+1) &= f^r(x(k)) + g^r(x(k))\,u(k) + d^r(k), \\ y(k) &= x^1(k), \end{aligned} \tag{3.1}$$

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 11–28 (2008)
www.springerlink.com

where $x(k) = \left[x^{1\top}(k), \ldots, x^{r\top}(k)\right]^{\top}$ are the state variables, and $\overline{x}^i(k) = \left[x^{1\top}, x^{2\top}, \ldots, x^{i\top}\right]^{\top}$, $x^i \in \Re^{n_i}$, $r \geq 2$, r is the number of blocks, $u(k) \in \Re^m$ is the system input, $y(k) \in \Re^m$ is the system output; for simplicity of notation through the remaining of this chapter $d^i(k) = d^i(x(k), k) \in \Re^{n_i}$ is the bounded unknown disturbance vector, then there exists a constant $\overline{d}_i$ such that $\|d_i(k)\| \leq \overline{d}_i$, for $0 < k < \infty$, $f^i(\bullet)$ and $g^i(\bullet)$ are unknown smooth nonlinear functions.

If we consider the original system (3.1) as a one-step ahead predictor, then we can transform it into an equivalent maximum r-step ahead one, which can predict the future states $x^1(k+r)$, $x^2(k+r-1), \ldots, x^r(k+1)$; the causality contradiction is avoided when the controller is constructed based on the maximum r-step ahead prediction by backstepping [3, 4]:

$$\begin{aligned}
x^1(k+r) &= \overline{f}^1\left(\bar{x}^1(k)\right) + \overline{g}^1\left(\overline{x}^1(k)\right) x^2(k+r-1) + d^1(k+r),\\
&\vdots\\
x^{r-1}(k+2) &= \overline{f}^{r-1}\left(\overline{x}^{r-1}(k)\right) + \overline{g}^{r-1}\left(\overline{x}^{r-1}(k)\right) x^r(k+1) + d^{r-1}(k+2),\\
x^r(k+1) &= \overline{f}^r(x(k)) + \overline{g}^r(x(k))\, u(k) + d^r(k),\\
y(k) &= x^1(k),
\end{aligned} \tag{3.2}$$

where $\overline{f}^i(\bullet)$ and $\overline{g}^i(\bullet)$ are unknown functions of $f^i\left(\overline{x}^i(k)\right)$ and $g^i(\overline{x}^i(k))$, respectively. For convenience of analysis, let us define $(i = 1, \ldots, r-1)$

$$\begin{aligned}
\overline{f}^i(k) &\triangleq \overline{f}^i\left(\overline{x}_i(k)\right),\\
\overline{g}^i(k) &\triangleq \overline{g}^i\left(\overline{x}_i(k)\right),\\
\overline{f}^r(k) &\triangleq \overline{f}^r\left(X(k)\right),\\
\overline{g}^r(k) &\triangleq \overline{g}^r\left(X(k)\right).
\end{aligned}$$

Then, system (3.2) can be written as (for details please see *Appendix A*)

$$\begin{aligned}
x^1(k+r) &= \overline{f}^1(k) + \overline{g}^1(k) x^2(k+r-1) + d^1(k+r),\\
&\vdots\\
x^{r-1}(k+2) &= \overline{f}^{r-1}(k) + \overline{g}^{r-1}(k) x^r(k+1) + d^{r-1}(k+2),\\
x^r(k+1) &= \overline{f}^r(k) + \overline{g}^r(k) u(k) + d^r(k),\\
y(k) &= x_1(k).
\end{aligned} \tag{3.3}$$

The objective is to design a control $u(k)$ to force the system output $y(k)$ to track a desired trajectory $y_{\mathrm{d}}(k)$. Once (3.3) is defined, we apply the well known backstepping technique [9]. For system (3.2), we can define the desired virtual controls $(\alpha^{j*}(k),\ j = 1, \ldots, r-1)$ and the ideal practical control $(u^*(k))$ as follows:

$$\begin{aligned}
\alpha^{1*}(k) &\triangleq x^2(k) = \varphi^1\big(\overline{x}^1(k), y_{\mathrm{d}}\,(k+r)\big),\\
\alpha^{2*}(k) &\triangleq x^3(k) = \varphi^2\big(\overline{x}^2(k), \alpha^{1*}(k)\big),\\
&\vdots\\
\alpha^{r-1*}(k) &\triangleq x^r(k) = \varphi^{r-1}\left(\overline{x}^{r-1}(k), \alpha^{r-2*}(k)\right),\\
u^*(k) &= \varphi^r\big(x(k), \alpha^{r-1*}(k)\big),\\
y(k) &= x^1(k),
\end{aligned} \tag{3.4}$$

where $\varphi^j(j = 1, \ldots, r)$ are nonlinear smooth functions. It is obvious that the desired virtual controls $\alpha^{i*}(k)$ and the ideal control $u^*(k)$ will drive the output $y(k)$ to track the desired signal $y_{\mathrm{d}}(k)$ only if the exact system model is known and there are no unknown disturbances. However, in practical applications, these two conditions cannot be satisfied. In the following, neural networks will be used to approximate the desired virtual controls, as well as the desired practical controls, when the conditions established above are not satisfied. As in [4], we construct the virtual and practical controls via embedded backstepping without the causality contradiction [3]. Let us approximate the virtual controls and practical control by the following HONN ($i = 1, \ldots, r-1$):

$$\begin{aligned}
\alpha^i(k) &= {w^i}^\top z^i(\varrho^i(k)), \qquad i = 1, \ldots, r-1,\\
u(k) &= {w^r}^\top z^r(\varrho^r(k)),
\end{aligned} \tag{3.5}$$

with

$$\begin{aligned}
\varrho^1(k) &= \left[x^1(k), y_{\mathrm{d}}(k+r)\right]^\top,\\
\varrho^i(k) &= \left[\overline{x}^i(k), \alpha^{i-1}(k)\right]^\top, \qquad i = 2, \ldots, r-1,\\
\varrho^r(k) &= \left[x(k), \alpha^{r-1}(k)\right]^\top,
\end{aligned}$$

where $w^j \in \Re^{L_j}$ are the estimates of ideal constant weights $w^{j*}(j = 1, \ldots, r)$ and $z^j \in \Re^{L_j \times n_j}$. Define the estimation error as

$$\widetilde{w}^j(k) = w^{j*} - w^j(k). \tag{3.6}$$

Using the ideal constant weights and from (2.11) it follows that there exists a HONN, which approximate the virtual controls and practical control with a minimal error, defined as

$$\begin{aligned}
\alpha^i(k) &= {w^{i*}}^\top z^i(\varrho^i(k)),\\
u(k) &= {w^{r*}}^\top z^r(\varrho^r(k)) + \epsilon_{z^i}, \qquad i = 1, \ldots, r-1.
\end{aligned} \tag{3.7}$$

Then the corresponding weights updating laws are defined by

$$w^j(k+1) = w^j(k) + \eta^j K^j(k) e^j(k), \tag{3.8}$$

with

$$\begin{aligned} K^j(k) &= P^j(k) H^j(k) M^{j^{-1}}(k), \\ M^j(k) &= R^j(k) + H^{j^\top}(k) P^j(k) H^j(k), \\ P^j(k+1) &= P^j(k) - K^j(k) H^{j^\top}(k) P^j(k) + Q^j(k), \\ H^j(k) &= \left[\frac{\partial \widehat{v}^j(k)}{\partial w^j(k)} \right], \end{aligned} \tag{3.9}$$

and

$$e^j(k) = v^j(k) - \widehat{v}^j(k), \tag{3.10}$$

where $v^i(k) \in \Re^{n_i}$ is the desired signal and $\widehat{v}^i(k) \in \Re^{n_i}$ is the HONN function approximation defined, respectively, as follows

$$\begin{aligned} v^1(k) &= y_{\mathrm{d}}(k), \\ v^2(k) &= x^2(k), \\ &\vdots \\ v^r(k) &= x^r(k) \end{aligned} \tag{3.11}$$

and

$$\begin{aligned} \widehat{v}^1(k) &= y(k), \\ \widehat{v}^2(k) &= \alpha^1(k), \\ &\vdots \\ \widehat{v}^r(k) &= \alpha^{r-1}(k), \end{aligned} \tag{3.12}$$

$e^j(k)$ denotes the error at each step, as

$$\begin{aligned} e^1(k) &= y_{\mathrm{d}}(k) - y(k), \\ e^2(k) &= x^2(k) - \alpha^1(k), \\ &\vdots \\ e^r(k) &= x^r(k) - \alpha^{r-1}(k). \end{aligned} \tag{3.13}$$

The proposed control scheme is shown in Fig. 3.1. Besides, it is worth to include the following comments:

Comment 3.1. The NN approximation error vector ϵ_z is bounded. This is a well known neural network property [2].

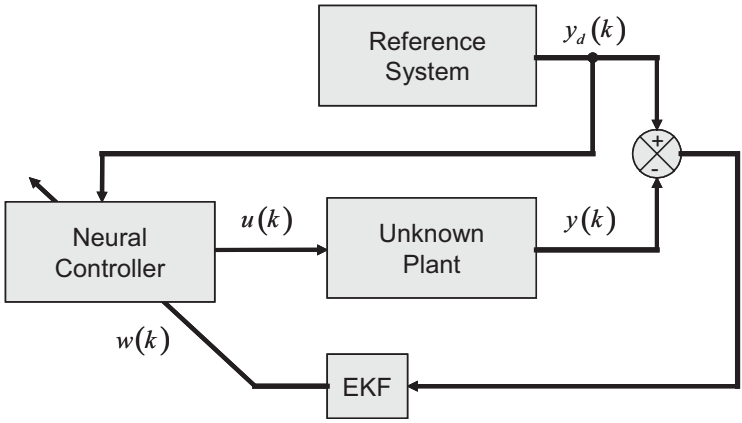

Fig. 3.1. Neural backstepping control scheme

Comment 3.2. The gain matrix of the EKF ($K(k)$) is bounded by a constant $\overline{K} > 0$, that is, $\|K(k)\| \leq \overline{K}$.

Before proceeding to demonstrate the main result of this chapter, we need to establish the following two lemmas.

Lemma 3.1. *The dynamics of the tracking error* (3.10) *can be formulated as*

$$e^j(k+1) = e^j(k) + \Delta e^j(k), \qquad (1 \leq j \leq r)\,, \tag{3.14}$$

with $\Delta e^j(k) \leq -\gamma^j e^j(k)$ *and* $\gamma^j = \max \left\| H^{j^\top}(k)\eta^j K^j(k) \right\|$.

Proof. Using (3.10) and considering that $\upsilon(k)$ do not depend on the HONN parameters, we obtain

$$\frac{\partial e^i(k)}{\partial w^i(k)} = -\frac{\partial \widehat{\upsilon}(k)}{\partial w^i(k)}. \tag{3.15}$$

Let us approximate (3.15) by

$$\Delta e^i(k) = \left[\frac{\partial e^i(k)}{\partial w^i(k)}\right]^\top \Delta w^i(k). \tag{3.16}$$

Substituting (3.9) and (3.15) in (3.16) yields

$$\Delta e^i(k) = -H^{i^\top}(k)\eta^i K^i(k) e^i(k). \tag{3.17}$$

Define

$$\gamma^i = \max \left\| H^{i^\top}(k)\eta^i K^i(k) \right\|$$

then we have

$$\Delta e^i(k) \leq -\gamma^i e^i(k). \tag{3.18}$$

□

Considering (3.1)–(3.13), we establish the main result of this chapter in the following theorem.

Theorem 3.1. *For the system* (3.1)*, the HONN* (3.5) *trained with the EKF-based algorithm* (3.9) *to approximate the control law* (3.4) *ensures that the tracking error* (3.13) *is semiglobally uniformly ultimately bounded (SGUUB); moreover, the HONN weights remain bounded.*

Proof. For the first block of system (3.1), with the virtual control $\alpha^{1*}(k)$ approximated by the HONN $\left(\alpha^1(k) = w^{1^\top} z^1\left(\varrho^1(k)\right)\right)$ and $e^1(k)$ defined as in (3.13), consider the Lyapunov function candidate

$$V^1(k) = e^{1^\top}(k)e^1(k) + \widetilde{w}^{1^\top}(k)\widetilde{w}^1(k), \tag{3.19}$$

whose first difference is

$$\begin{aligned}\Delta V^1(k) &= V^1(k+1) - V^1(k),\\ &= e^{1^\top}(k+1)e^1(k+1) + \widetilde{w}^{1^\top}(k+1)\widetilde{w}^1(k+1)\\ &\quad - e^{1^\top}(k)e^1(k) - \widetilde{w}^{1^\top}(k)\widetilde{w}^1(k).\end{aligned} \tag{3.20}$$

From (3.6) and (3.8), then

$$\widetilde{w}^1(k+1) = \widetilde{w}^1(k) - \eta^1 K^1(k)e^1(k). \tag{3.21}$$

Let us define

$$\begin{aligned}&\left[\widetilde{w}^1(k) - \eta^1 K^1(k)e^1(k)\right]^\top \left[\widetilde{w}^1(k) - \eta^1 K^1(k)e^1(k)\right]\\ &= \widetilde{w}^{1^\top}(k)\widetilde{w}^1(k) - 2\widetilde{w}^{1^\top}(k)\eta^1 K^1(k)e^1(k)\\ &\quad + \left(\eta^1 K^1(k)e^1\right)^\top \eta^1 K^1(k)e^1(k).\end{aligned} \tag{3.22}$$

From (3.13), then

$$\begin{aligned}e^1(k+1) &= e^1(k) + \Delta e^1(k),\\ e^{1^\top}(k+1)e^1(k+1) &= e^{1^\top}(k)e^1(k) + e^{1^\top}(k)\Delta e^1(k)\\ &\quad + \Delta e^{1^\top}(k)e^1(k) + \Delta e^{1^\top}(k)\Delta e^1(k),\\ e^{1^\top}(k+1)e^1(k+1) - e^{1^\top}(k)e^1(k) &= e^{1^\top}(k)\Delta e^1(k) + \Delta e^{1^\top}(k)e^1(k)\\ &\quad + \Delta e^{1^\top}(k)\Delta e^1(k),\end{aligned}$$

where $\Delta e^1(k)$ is the error difference. Substituting (3.21) and (3.22) in (3.20) results in

$$\begin{aligned}\Delta V^1(k) &= e^{1^\top}(k)\Delta e^1(k) + \Delta e^{1^\top}(k)e^1(k) + \Delta e^{1^\top}(k)\Delta e^1(k)\\ &\quad - 2\widetilde{w}^{1^\top}(k)\eta^1 K^1(k)e^1(k)\\ &\quad + (\eta^1 K^1(k)e^1(k))^\top \eta^1 K^1(k)e^1(k).\end{aligned} \tag{3.23}$$

From Lemma 3.1, substituting (3.18), we obtain

$$\begin{aligned}\Delta V^1(k) \leq & -2\gamma^1 e^{1^\top}(k)e^1(k) + \gamma^{1^2} e^{1^\top}(k)e^1(k) - 2\widetilde{w}^{1^\top}(k)\eta^1 K^1(k)e^1(k) \\ & + (\eta^1 K^1(k)e^1(k))^\top \eta^1 K^1(k)e^1(k), \\ \leq & -2\gamma^1 \left\|e^1(k)\right\|^2 + \gamma^{1^2}\left\|e^1(k)\right\|^2 - 2\|\eta^1 K^1(k)\|\left\|\widetilde{w}^1(k)\right\|\left\|e^1(k)\right\| \\ & + \|\eta^1 K^1(k)\|^2\left\|e^1(k)\right\|^2, \\ \leq & -2\gamma^1 \left\|e^1(k)\right\|^2 + \gamma^{1^2}\left\|e^1(k)\right\|^2 \\ & - 2\|\eta^1 K^1(k)\|\left\|w^* - w^1_{\max}\right\|\left\|e^1(k)\right\| \\ & + \|\eta^1 K^1(k)\|^2\left\|e^1(k)\right\|^2, \end{aligned} \tag{3.24}$$

with $\gamma^1 = \max\left\|H^{1^\top}(k)\eta^1 K^1(k)\right\| > 1$. There is $\eta^1 > 0$ such that $\vartheta^1 > 0$ with

$$\vartheta^1 = 2\gamma^1 - \gamma^{1^2} - \|\eta^1 K^1(k)\|^2,$$

then

$$\Delta V^1(k) \leq 0, \quad \text{once} \quad \left\|e^1(k)\right\| > \kappa^1, \tag{3.25}$$

with κ^1 defined as

$$\kappa^1 > \frac{\left(\gamma^{1^2} + \|\eta^1 K^1(k)\|^2\right)\left\|e^1_{\max}\right\|}{2\|\eta^1 K^1(k)\|}.$$

Therefore, the solution of (3.14) and (3.21) is stable, which leads to the SGUUB of $e^1(k)$ and $\widetilde{w}^1(k)$.

For the following ith ($i = 2, \ldots, r-1$) equation of the system (3.1), with the virtual control $\alpha^{i*}(k)$ approximated by the HONN $\alpha^i(k) = w^{i^\top} z^i(\varrho^i(k))$ and $e^i(k)$ defined in (3.13), consider the Lyapunov function candidate

$$V^i(k) = e^{i^\top}(k)e^i(k) + \widetilde{w}^{i^\top}(k)\widetilde{w}^i(k), \tag{3.26}$$

whose first difference is

$$\begin{aligned}\Delta V^i(k) &= V^i(k+1) - V^i(k), \\ &= e^{i^\top}(k+1)e^i(k+1) + \widetilde{w}^{i^\top}(k+1)\widetilde{w}^i(k+1) \\ &\quad - e^{i^\top}(k)e^i(k) - \widetilde{w}^{i^\top}(k)\widetilde{w}^i(k).\end{aligned} \tag{3.27}$$

From (3.6) and (3.8), then

$$\widetilde{w}^i(k+1) = \widetilde{w}^i(k) - \eta^i K^i(k)e^i(k). \tag{3.28}$$

Let us define

$$\begin{aligned}
&\left[\widetilde{w}^i(k) - \eta^i K^i(k) e^i(k)\right]^\top \left[\widetilde{w}^i(k) - \eta^i K^i(k) e^i(k)\right] \\
&= \widetilde{w}^{i^\top}(k)\widetilde{w}^i(k) - 2\widetilde{w}^{i^\top}(k)\eta^i K^i(k) e^i(k) \\
&\quad + \left(\eta^i K^i(k) e^i(k)\right)^\top \eta^i K^i(k) e^i(k).
\end{aligned} \tag{3.29}$$

From (3.13), then

$$\begin{aligned}
e^i(k+1) &= e^i(k) + \Delta e^i(k), \\
e^{i^\top}(k+1)e^i(k+1) &= e^{i^\top}(k)e^i(k) + e^{i^\top}(k)\Delta e^i(k) \\
&\quad + \Delta e^{i^\top}(k)e^i(k) \\
&\quad + \Delta e^{i^\top}(k)\Delta e^i(k), \\
e^{i^\top}(k+1)e^i(k+1) - e^{i^\top}(k)e^i(k) &= e^{i^\top}(k)\Delta e^i(k) + \Delta e^{i^\top}(k)e^i(k) \\
&\quad + \Delta e^{i^\top}(k)\Delta e^i(k),
\end{aligned}$$

where $\Delta e^i(k)$ is the error difference. Substituting (3.28) and (3.29) in (3.27) results in

$$\begin{aligned}
\Delta V^i(k) = {} & e^{i^\top}(k)\Delta e^i(k) + \Delta e^{i^\top}(k)e^i(k) + \Delta e^{i^\top}(k)\Delta e^i(k) \\
& - 2\widetilde{w}^{i^\top}(k)\eta^i K^i(k) e^i(k) \\
& + \left(\eta^i K^i(k) e^i(k)\right)^\top \eta^i K^i(k) e^i(k).
\end{aligned} \tag{3.30}$$

From Lemma 3.1, substituting (3.18), we obtain

$$\begin{aligned}
\Delta V^i(k) &\le -2\gamma^i e^{i^\top}(k)e^i(k) + \gamma^{i^2} e^{i^\top}(k)e^i(k) - 2\widetilde{w}^{i^\top}(k)\eta^i K^i(k) e^i(k) \\
&\quad + \left(\eta^i K^i(k) e^i(k)\right)^\top \eta^i K^i(k) e^i(k), \\
&\le -2\gamma^i \left\|e^i(k)\right\|^2 + \gamma^{i^2}\left\|e^i(k)\right\|^2 - 2\left\|\eta^i K^i(k)\right\| \left\|\widetilde{w}^i(k)\right\| \left\|e^i(k)\right\| \\
&\quad + \left\|\eta^i K^i(k)\right\|^2 \left\|e^i(k)\right\|^2, \\
&\le -2\gamma^i \left\|e^i(k)\right\|^2 + \gamma^{i^2}\left\|e^i(k)\right\|^2 \\
&\quad - 2\left\|\eta^i K^i(k)\right\| \left\|w^i_{\max} - w^*\right\| \left\|e^i(k)\right\| \\
&\quad + \left\|\eta^i K^i(k)\right\|^2 \left\|e^i(k)\right\|^2.
\end{aligned} \tag{3.31}$$

with $\gamma^i = \max\left\|H^{i^\top}(k)\eta^i K^i(k)\right\|$. There is $\eta^i > 0$ such that $\vartheta^i > 0$ with

$$\vartheta^i = 2\gamma^i - \gamma^{i^2} - \left\|\eta^i K^i(k)\right\|^2,$$

then

$$\Delta V^i(k) \le 0, \quad \text{once} \quad \left\|e^i(k)\right\| > \kappa^i, \tag{3.32}$$

with κ^i defined as

$$\kappa^i > \frac{\left(\gamma^{i^2} + \left\|\eta^i K^i(k)\right\|^2\right)\left\|e^i_{\max}\right\|}{2\left\|\eta^i K^i(k)\right\|}.$$

Therefore, the solution of (3.14) and (3.28) is stable, which leads to the SGUUB of $e^i(k)$ and $\widetilde{w}^i(k)$. □

3.2 Applications

In this section, we apply the above developed scheme (Fig. 3.1) to control a three-phase induction motor, which is one of the most used actuators for industrial applications due to its reliability, ruggedness, and relatively low cost. The control of an induction motor is challenging, since its dynamics is described by multivariable, coupled, and highly nonlinear system [13, 15]. Early works on control of induction motors was focused on the field oriented control (FOC) [7], exact input–output linearization, adaptive input–output linearization, and direct torque control (DTC) ([7] and references therein). However, most of those works were developed stabilized controllers for continuous-time model of the motor. In [13] a discrete-time model is proposed, as well as a control algorithm, assuming that the parameters and load torque of the motor model are known. Moreover, all these controllers are designed based on the physical model of the motor and results in sensitive control with respect to plant parameters variations. To this end, we consider the control problem assuming that some of the plant parameters as well as external disturbances (load torque) are unknown.

3.2.1 Motor Model

The six-order discrete-time induction motor model in the stator fixed reference frame (α, β), under the assumptions of equal mutual inductances and linear magnetic circuit, is given by [13]

$$\begin{aligned}
\omega(k+1) &= \omega(k) + \frac{\mu}{\alpha}(1-\alpha) \times M\left(i^\beta(k)\psi^\alpha(k) - i^\alpha(k)\psi^\beta(k)\right) \\
&\quad - \left(\frac{T}{J}\right) T_{\mathrm{L}}(k), \\
\psi^\alpha(k+1) &= \cos\left(n_{\mathrm{p}}\theta(k+1)\right)\rho_1(k) - \sin\left(n_{\mathrm{p}}\theta\,(k+1)\right)\rho_2(k), \\
\psi^\beta(k+1) &= \sin\left(n_{\mathrm{p}}\theta(k+1)\right)\rho_1(k) + \cos\left(n_{\mathrm{p}}\theta(k+1)\right)\rho_2(k), \\
i^\alpha(k+1) &= \varphi^\alpha(k) + \frac{T}{\sigma}u^\alpha(k) + d_1(k), \\
i^\beta(k+1) &= \varphi^\beta(k) + \frac{T}{\sigma}u^\beta(k) + d_2(k),
\end{aligned}$$

$$\theta(k+1) = \theta(k) + \omega(k)T - \frac{T_{\mathrm{L}}(k)}{J}T^2 + \frac{\mu}{\alpha}\left[T - \frac{(1-a)}{\alpha}\right] M\big(i^{\beta}(k)\psi^{\alpha}(k) - i^{\alpha}(k)\psi^{\beta}(k)\big), \quad (3.33)$$

with

$$\rho_1(k) = a\left(\cos\left(\phi(k)\right)\psi^{\alpha}(k) + \sin\left(\phi(k)\right)\psi^{\beta}(k)\right) + b\left(\cos\left(\phi(k)\right)i^{\alpha}(k) + \sin\left(\phi(k)\right)i^{\beta}(k)\right),$$

$$\rho_2(k) = a\left(\cos\left(\phi(k)\right)\psi^{\alpha}(k) - \sin\left(\phi(k)\right)\psi^{\beta}(k)\right) + b\left(\cos\left(\phi(k)\right)i^{\alpha}(k) - \sin\left(\phi(k)\right)i^{\beta}(k)\right),$$

$$\begin{aligned}\varphi^{\alpha}(k) &= i^{\alpha}(k) + \alpha\beta T\psi^{\alpha}(k) + n_{\mathrm{p}}\beta T\omega(k)\psi^{\alpha}(k) - \gamma T i^{\alpha}(k),\\ \varphi^{\beta}(k) &= i^{\beta}(k) + \alpha\beta T\psi^{\beta}(k) + n_{\mathrm{p}}\beta T\omega(k)\psi^{\beta}(k) - \gamma T i^{\beta}(k),\\ \phi(k) &= n_{\mathrm{p}}\theta(k),\end{aligned}$$

with $b = (1-a)M$, $\alpha = \frac{R_{\mathrm{r}}}{L_{\mathrm{r}}}$, $\gamma = \frac{M^2R_{\mathrm{r}}}{\sigma L_{\mathrm{r}}^2} + \frac{R_{\mathrm{s}}}{\sigma}$, $\sigma = L_{\mathrm{s}} - \frac{M^2}{L_{\mathrm{r}}}$, $\beta = \frac{M}{\sigma L_{\mathrm{r}}}$, $a = \mathrm{e}^{-\alpha T}$, $\mu = \frac{Mn_{\mathrm{p}}}{JL_{\mathrm{r}}}$, where L_{s}, L_{r}, and M are the stator, rotor, and mutual inductance, respectively; R_{s} and R_{r} are the stator and rotor resistances, respectively; n_{p} is the number of pole pairs; i^{α} and i^{β} represents the currents in the α and β phases, respectively; ψ^{α} and ψ^{β} represents the fluxes in the α and β phases, respectively; and θ is the rotor angular displacement.

3.2.2 Block-Strict-Feedback-Form (BSFF) for an Induction Motor

Let us define the following states:

$$\begin{aligned}x^1(k) &= \begin{bmatrix}\omega(k)\\ \Psi(k)\end{bmatrix}; \quad x^2(k) = \begin{bmatrix}i^{\alpha}(k)\\ i^{\beta}(k)\end{bmatrix},\\ u(k) &= \begin{bmatrix}u^{\alpha}(k)\\ u^{\beta}(k)\end{bmatrix}; \quad y_{\mathrm{d}}(k) = \begin{bmatrix}\omega_{\mathrm{d}}(k)\\ \Psi_{\mathrm{d}}(k)\end{bmatrix},\\ y(k) &= x^1(k),\end{aligned} \quad (3.34)$$

where $\Psi(k) = \psi^{\alpha^2}(k) + \psi^{\beta^2}(k)$ is the rotor flux magnitude, $\omega_{\mathrm{d}}(k)$ and $\Psi_{\mathrm{d}}(k)$ are the reference signals. The objective of control is to drive the output $y(k)$ to track the reference $y_{\mathrm{d}}(k)$. Using (3.34) the system (3.33) can be represented in the BSFF consisting of two blocks

$$\begin{aligned}x^1(k+1) &= f^1\big(x^1(k)\big) + g^1\big(x^1(k)\big)\,x^2(k) + d^1(k),\\ x^2(k+1) &= f^2\big(\overline{x}^2(k)\big) + g^2\big(\overline{x}^2(k)\big)\,u(k),\end{aligned}$$

where $f^1\big(x^1(k)\big)$, $g^1(x_1(k))$, $f_2(\overline{x}_2(k))$, and $g_2(\overline{x}_2(k))$ are assumed to be unknown and $d_1(k)$ is the unknown bounded disturbances; in this case this disturbance is the load torque. Now we use the HONN to approximate the desired virtual controls and the ideal practical controls described as

$$\begin{aligned}\alpha^{1*}(k) \triangleq x^2(k) &= \varphi^1\big(x^1(k), y_{\mathrm{d}}(k+2)\big),\\ u^*(k) &= \varphi^2\big(x^1(k), x^2(k), \alpha^{1*}(k)\big),\\ y(k) &= x^1(k).\end{aligned}$$

The HONN proposed for this application is as follows:

$$\begin{aligned}\alpha^1(k) &= {w^1}^{\top} z^1\left(\varrho^1(k)\right),\\ u(k) &= {w^2}^{\top} z^2\left(\varrho^2(k)\right),\end{aligned}$$

with

$$\begin{aligned}\varrho^1(k) &= \left[x^1(k), y_d\,(k+2)\right]^{\top},\\ \varrho^2(k) &= \left[x^1(k), x^2(k), \alpha^1(k)\right]^{\top}.\end{aligned}$$

The weights are updated using the EKF as follows:

$$\begin{aligned}w^i(k+1) &= w^i(k) + \eta^i K^i(k) e^i(k) \qquad (i = 1, 2),\\ K^i(k) &= P^i(k) H^i(k)\left[R^i(k) + {H^i}^{\top}(k) P^i(k) H^i(k)\right]^{-1},\\ P^i(k+1) &= P^i(k) - K^i(k) {H^i}^{\top}(k) P^i(k) + Q^i(k),\end{aligned}$$

with

$$\begin{aligned}e^1(k) &= y_{\mathrm{d}}(k) - y(k),\\ e^2(k) &= x^2(k) - \alpha^1(k)\,.\end{aligned}$$

The training is performed online using a series–parallel configuration. All the NN states are initialized in a random way. The associated covariances matrices are initialized as diagonals, and the nonzero elements are $P_1\,(0) = P_2\,(0) = 10000$; $Q_1\,(0) = Q_2\,(0) = 5000$, and $R_1\,(0) = R_2\,(0) = 10000$, respectively. The simulation is performing under the presence of the disturbances $d^1(k)$ as shown in Fig. 3.3 and parametric variations (Fig. 3.4).

3.2.3 Reduced Order Nonlinear Observer

The last control algorithm works with the full state measurement assumption [13]. However, the rotor fluxes measurement is a difficult task. Here, a reduced order nonlinear observer is designed for fluxes with rotor speed and current measurements only. The flux dynamics in (3.33) can be written as

$$\Psi(k+1) = aG(k)\Psi(k) + (1-a)MG(k)\mathbf{I}(k),$$

with

$$G(k) = \begin{bmatrix} \cos(n_{\mathrm{p}} T\omega(k)) & -\sin(n_{\mathrm{p}} T\omega(k)) \\ \sin(n_{\mathrm{p}} T\omega(k)) & \cos(n_{\mathrm{p}} T\omega(k)) \end{bmatrix},$$
$$\mathbf{I}(k) = \begin{bmatrix} i^{\alpha}(k) \\ i^{\beta}(k) \end{bmatrix}. \tag{3.35}$$

The proposed observer for the system (3.33) assumes the speed and current available for measurements:

$$\widehat{\Psi}(k+1) = aG(k)\widehat{\Psi}(k) + (1-a)MG(k)\mathbf{I}(k).$$

Let us define

$$e^{\Psi}(k) = \Psi(k) - \widehat{\Psi}(k).$$

Then

$$e^{\Psi}(k+1) = aG(k)e^{\Psi}(k).$$

A Lyapunov candidate function to proof stability of $e^{\Psi}(k)$ is

$$V(k) = e^{\Psi^{\top}}(k)e^{\Psi}(k), \tag{3.36}$$

with

$$\begin{aligned} \Delta V(k) &= V(k+1) - V(k) = e^{\Psi^{\top}}(k-1)\, e^{\Psi}(k+1) - e^{\Psi^{\top}}(k)e^{\Psi}(k), \\ &= e^{\Psi^{\top}}(k)\big(a^2 G^{\top}(k)G(k) - I\big)\, e^{\Psi}(k), \end{aligned}$$

where

$$a^2 G^{\top}(k)G(k) - I < 0. \tag{3.37}$$

By (3.35), $G^{\top}(k)G(k) = I$ then the condition (3.37) is reduced to

$$\begin{bmatrix} a^2 & 0 \\ 0 & a^2 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} < 0,$$

where $a < 1$, $a = \mathrm{e}^{-\alpha T}$. This condition is satisfied due to the fact that T and α are always positive. So the increment of the Lyapunov function (3.36) is always negative, implying that the tracking error tends asymptotically to zero. Now we use $\widehat{\psi}^{\alpha}$ and $\widehat{\psi}^{\beta}$ to implement the control algorithm developed above.

3.2.4 Simulation Results

The simulation is performed using the system (3.33) with the parameters given in Table 3.1.

The tracking results are presented in Figs. 3.2 and 3.3. There the tracking performance can be verified for the two plant outputs. Figure 3.4 displays the load torque applied as an external disturbance. Figure 3.5 portrays a parametric variation introduced in the rotor resistance (R_{r}) as an increment. Figure 3.6 shows the weight evolution. Figures 3.7 and 3.8 display the control law signals. Figures 3.9 and 3.10 portray the fluxes and their estimates.

Table 3.1. Induction motor parameters

Parameter	Value	Description
R_{s}	$14\,\Omega$	Stator resistance
L_{s}	400 mH	Stator inductance
M	377 mH	Mutual inductance
R_{r}	$10.1\,\Omega$	Rotor resistance
L_{r}	412.8 mH	Rotor inductance
n_{p}	2	Number of pole pairs
J	$0.01\,\mathrm{Kg\,m^2}$	Moment of inertia
ω_{n}	$168.5\,\mathrm{rad\,s^{-1}}$	Nominal speed
$T_{L_{\mathrm{n}}}$	1.1 N m	Nominal load
T	0.0001 s	Sampling period

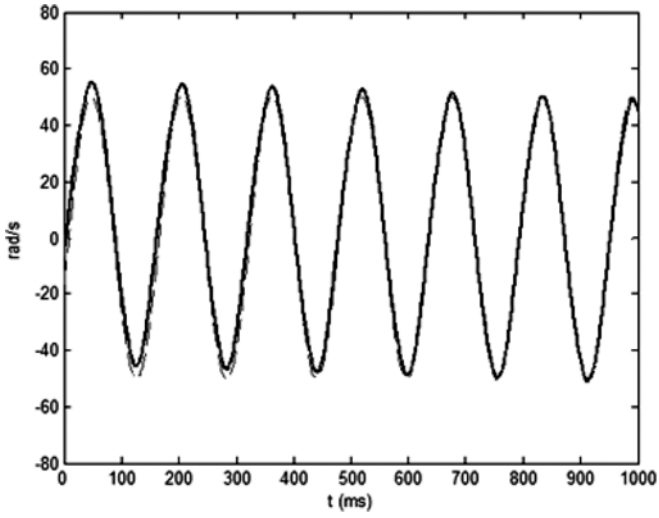

Fig. 3.2. Tracking performance $\omega(k)$ (*solid line*) and $\omega_{\mathrm{d}}(k)$ (*dashed line*)

Comment 3.3. The purpose of this chapter is to improve the tracking performance for a class of MIMO discrete-time nonlinear systems, by means of the use of the EKF as the neural network learning algorithm; this approach is validated by the simulation results presented above.

Comment 3.4. In this chapter, the causality contradiction is avoided due to the fact that the controller is constructed based on the maximum r-step ahead predictor by the backstepping technique.

Comment 3.5. In literature there are few results that present both external disturbances (load torque) and parametric changes (resistance variations) as in this book.

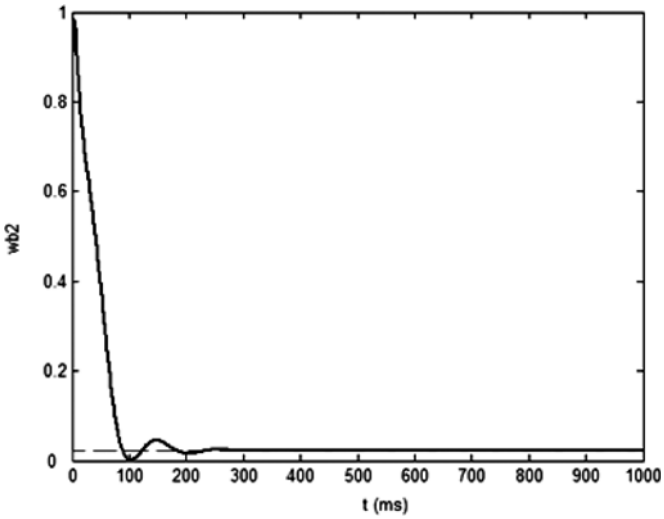

Fig. 3.3. Tracking performance $\Psi(k)$ (*solid line*) and $\Psi_{\mathrm{d}}(k)$ (*dashed line*)

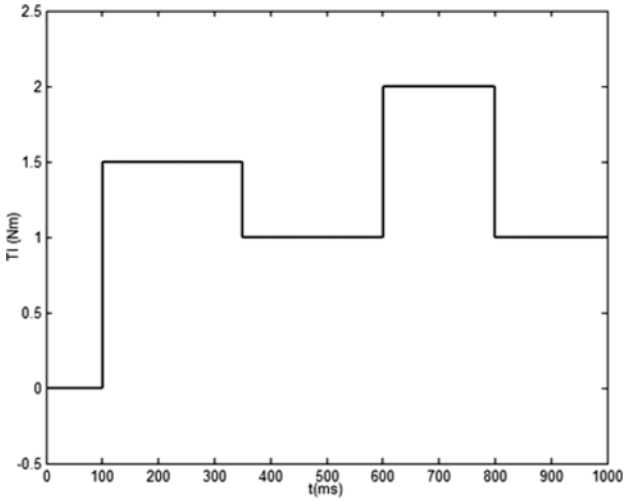

Fig. 3.4. Load torque $T_{\mathrm{L}}(k)$

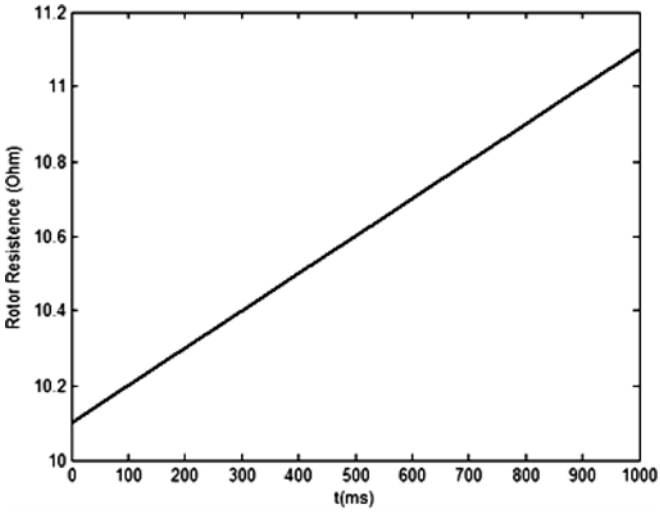

Fig. 3.5. Rotor resistance variation (R_r)

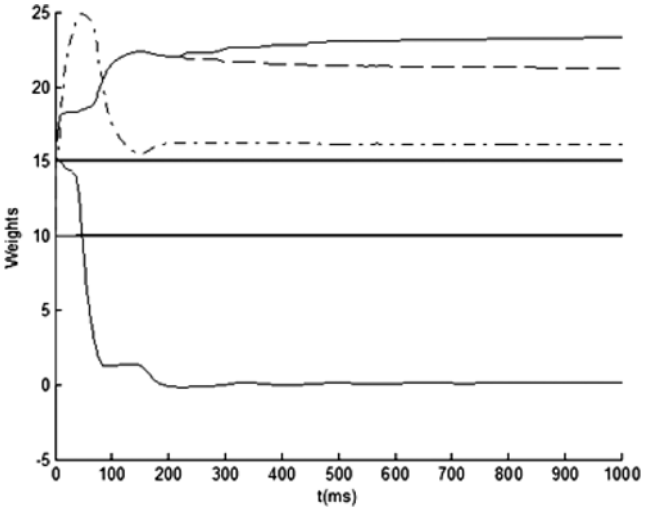

Fig. 3.6. Weights evolution

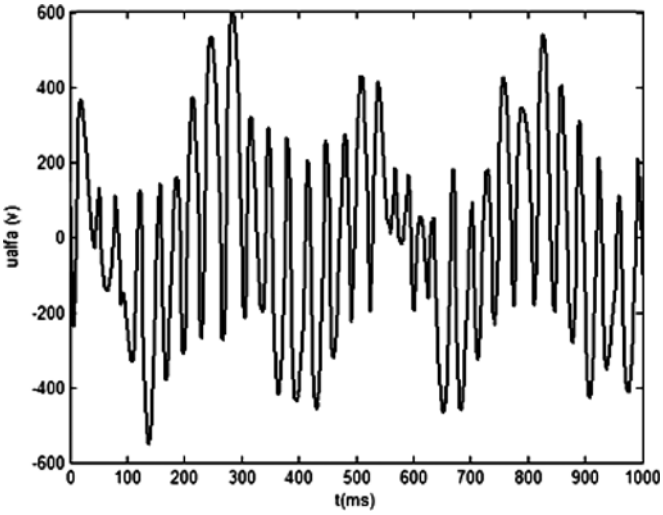

Fig. 3.7. Control law signal $u^{\alpha}(k)$

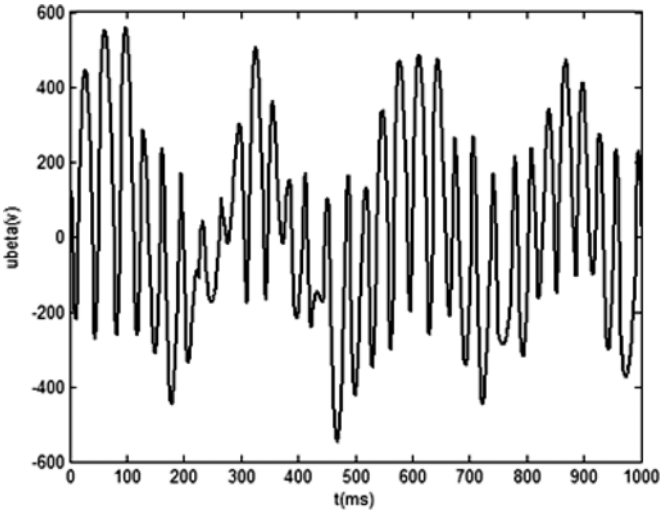

Fig. 3.8. Control law signal $u^{\beta}(k)$

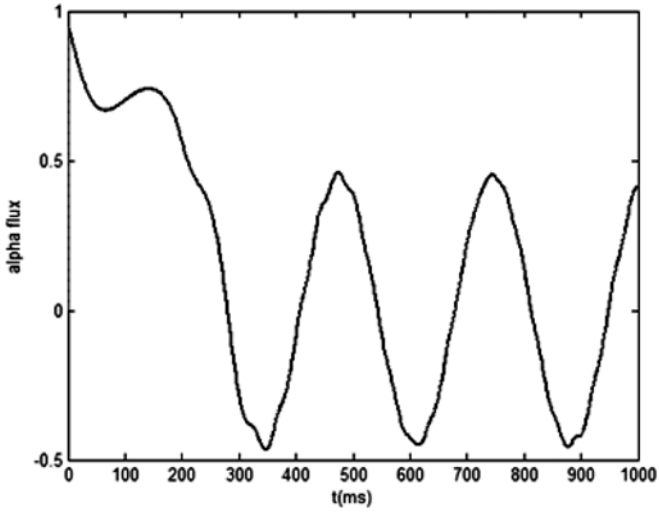

Fig. 3.9. Time evolution of $\psi^{\alpha}(k)$ and its estimate (real in *solid line* and estimated in *dashed line*)

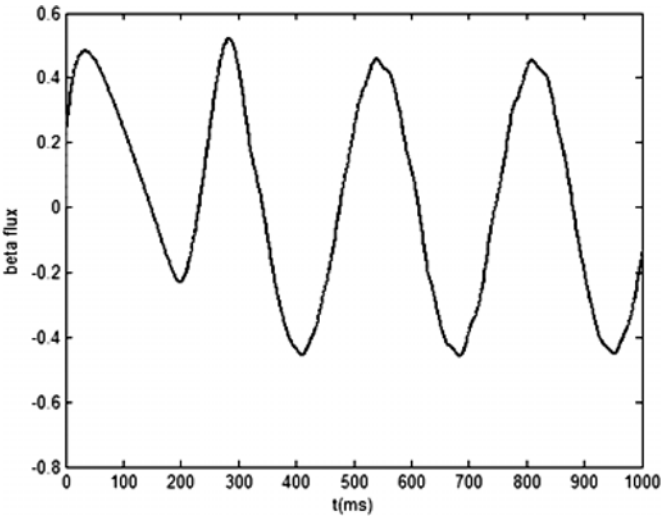

Fig. 3.10. Time evolution of $\psi^{\beta}(k)$ and its estimate (real in *solid line* and estimated in *dashed line*)

3.3 Conclusions

This chapter has presented the application of HONN to solve the tracking problem for a class of MIMO discrete-time nonlinear systems, using the backstepping technique. The training of the neural network is performed online using an extended Kalman filter. The boundness of the tracking error is established on the basis of the Lyapunov approach. The HONN training with the learning algorithm based in EKF presents good performance even in presence of larger bounded disturbances such as load torque variations and change on the plant parameters (resistance change). Based on the proposed control scheme, a robust neural controller is designed for an induction motor.

4

Discrete-Time Block Control

This chapter deals with the adaptive tracking problem for a class of MIMO discrete-time nonlinear systems in presence of bounded disturbances. In this chapter, a recurrent high order neural network is first used to identify the plant model, then based on this neural model, a discrete-time control law, which combines discrete-time block control and sliding modes techniques, is derived. The chapter also includes the respective stability analysis for the whole system. It is proposed too a strategy to avoid specific adaptive weights zero-crossing. Applicability of the proposed scheme is illustrated via simulation of a discrete-time nonlinear controller for an induction motor.

Frequently, modern control systems require a very structured knowledge about the system to be controlled; such knowledge should be represented in terms of differential or difference equations. This mathematical description of the dynamic system is named as the model. Basically there are two ways to obtain a model; it can be derived in a deductive manner using physics laws, or it can be inferred from a set of data collected during a practical experiment. The first method can be simple, but in many cases it is excessively time-consuming; some times, it would be unrealistic or impossible to obtain an accurate model in this way. The second method, which is commonly referred as system identification, could be a useful short cut for deriving mathematical models. Although system identification not always results in a equally accurate model, a satisfactory model can be often obtained with reasonable efforts. The main drawback is the requirement to conduct a practical experiment, which brings the system through its range of operation. Besides a certain knowledge about the plant is still required.

4.1 Identification

In this section, we consider the problem to identify the nonlinear system

$$x(k+1) = F(x(k), u(k)), \tag{4.1}$$

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 29–43 (2008)
www.springerlink.com

where $x \in \Re^n$, $u \in \Re^m$, and $F \in \Re^n \times \Re^m \to \Re^n$ are nonlinear functions. Now, to identify the system (4.1) we use an RHONN defined as

$$\widehat{x}_i(k+1) = w_i^\top z_i(x(k), u(k)), \quad i = 1, \ldots, n, \tag{4.2}$$

where $\widehat{x}_i$ $(i = 1, 2, \ldots, n)$ is the state of the ith neuron, L_i is the respective number of higher-order connections, $\{I_1, I_2, \ldots, I_{L_i}\}$ is a collection of nonordered subsets of $\{1, 2, \ldots, n+m\}$, n is the state dimension, m is the number of external inputs, w_i $(i = 1, 2, \ldots, n)$ is the respective online adapted weight vector, with $z_i(x(k), u(k))$ as defined in (2.8).

Consider the problem to approximate the general discrete-time nonlinear system (4.1) by the following discrete-time RHONN series–parallel representation [5]:

$$x_i(k+1) = w_i^{*\top} z_i(x(k), u(k)) + \epsilon_{z_i}, \quad i = 1, \ldots, n, \tag{4.3}$$

where x_i is the ith plant state, ϵ_{z_i} is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [5]. Assume that there exists an ideal weights vector w_i^* such that $\|\epsilon_{z_i}\|$ can be minimized on a compact set $\Omega_{z_i} \subset \Re^{L_i}$. The ideal weight vector w_i^* is an artificial quantity required for analytical purpose [5]. In general, it is assumed that this vector exists and is constant but unknown. Let us define its estimate as w_i and the estimation error as

$$\widetilde{w}_i(k) = w_i^* - w_i(k). \tag{4.4}$$

The estimate w_i is used for the stability analysis, which will be discussed later. Since w_i^* is constant, then

$$\widetilde{w}_i(k+1) - \widetilde{w}_i(k) = w_i(k) - w_i(k+1), \quad \forall k \in 0 \cup \mathbb{Z}^+.$$

The RHONN is trained with a modified extended Kalman filter (EKF) algorithm defined by

$$\begin{aligned} w_i(k+1) &= w_i(k) + \eta_i K_i(k) e_i(k), \\ K_i(k) &= \begin{cases} P_i(k) H_i(k) M_i(k) & \text{if } \|w_i(k)\| > c_i, \\ 0 \text{ if } \|w_i(k)\| < c_i, \end{cases} \\ P_i(k+1) &= P_i(k) - K_i(k) H_i^\top(k) P_i(k) + Q_i(k), \\ & i = 1, \ldots, n, \end{aligned} \tag{4.5}$$

with

$$M_i(k) = \left[R_i(k) + H_i^\top(k) P_i(k) H_i(k)\right]^{-1}, \tag{4.6}$$

$$e_i(k) = x_i(k) - \widehat{x}_i(k), \tag{4.7}$$

where $c_i > 0$ is a constrain used to avoid the zero-crossing, $e_i(k) \in \Re$ is the respective identification error, $P_i(k) \in \Re^{L_i \times L_i}$ is the prediction error associated covariance matrix at step k, $w_i \in \Re^{L_i}$ is the weight (state) vector, L_i

is the respective number of neural network weights, x_i is the ith plant state, $\widehat{x}_i$ is the ith neural network state, n is the number of states, $K_i \in \Re^{L_i}$ is the Kalman gain vector, $Q_i \in \Re^{L_i \times L_i}$ is the state noise associated covariance matrix, $R_i \in \Re$ is the measurement noise associated covariance; $H_i \in \Re^{L_i}$ is a vector, in which each entry $(H_{_{ij}})$ is the derivative of one of the neural network state $(\widehat{x}_{_i})$ with respect to one neural network weight, $(w_{_{ij}})$, defined as follows:

$$H_{_{ij}}(k) = \left[\frac{\partial \widehat{x}_{_i}(k)}{\partial w_{_{ij}}(k)}\right]^{\top}_{w_i(k)=w_i(k+1)}, \tag{4.8}$$

where $i = 1, ..., n$ and $j = 1, ..., L_i$. If we select $c_i = 0$, the modified EKF (4.5) becomes the standard extended Kalman filter [2, 8]. Usually P_i and Q_i are initialized as diagonal matrices, with entries $P_i(0)$ and $Q_i(0)$, respectively. It is important to remark that $H_i(k)$, $K_i(k)$, and $P_i(k)$ for the EKF are bounded; for a detailed explanation of this fact see [6].

Then the dynamics of (4.7) can be expressed as

$$e_i(k+1) = \widetilde{w}_i(k) z_i(x(k), u(k)) + \epsilon_{z_i}. \tag{4.9}$$

By the other hand, the dynamics of (4.4) is

$$\widetilde{w}_i(k+1) = \widetilde{w}_i(k) - \eta_i K_i(k) e(k). \tag{4.10}$$

Now, we establish the first main result of this chapter in the following theorem.

Theorem 4.1. *The RHONN* (4.2) *trained with the modified EKF-based algorithm* (4.5) *to identify the nonlinear plant* (4.1) *ensures that the identification error* (4.7) *is semiglobally uniformly ultimately bounded (SGUUB); moreover, the RHONN weights remain bounded.*

Proof. Case 4.1. $\|w_i(k)\| > c_i$: Consider the Lyapunov function candidate

$$\begin{aligned} V_i(k) &= \widetilde{w}_i^{\mathrm{T}}(k)\widetilde{w}_i(k) + e_i^2(k), \\ \Delta V_i(k) &= V(k+1) - V(k), \\ &= \widetilde{w}_i^{\mathrm{T}}(k+1)\widetilde{w}_i(k+1) + e_i^2(k+1) \\ &\quad - \widetilde{w}_i^{\mathrm{T}}(k)\widetilde{w}_i(k) - e_i^2(k), \end{aligned} \tag{4.11}$$

Using (4.9) and (4.10) in (4.11),

$$\begin{aligned} \Delta V_i(k) = &\left[\widetilde{w}_i(k) - \eta_i K_i(k) e_i(k)\right]^{\mathrm{T}} \left[\widetilde{w}_i(k) - \eta_i K_i(k) e_i(k)\right] \\ &+ \left[\widetilde{w}_i(k) z_i(x(k), u(k)) + \epsilon_{z_i}\right]^{\mathrm{T}} \left[\widetilde{w}_i(k) z_i(x(k), u(k)) + \epsilon_{z_i}\right] \\ &- \widetilde{w}_i(k)\widetilde{w}_i(k) - e_i^2(k). \end{aligned} \tag{4.12}$$

Then, (4.12) can be expressed as

$$\begin{aligned}
\Delta V_i(k) &= \widetilde{w}_i^{\mathrm{T}}(k)\widetilde{w}_i(k) - \widetilde{w}_i^{\mathrm{T}}(k)\widetilde{w}_i(k) \\
&\quad + \eta^2 e_i^2(k) K^{\mathrm{T}} K_i(k) + 2\epsilon_{z_i}\widetilde{w}_i(k) z_i(x(k), u(k)) \\
&\quad + z_i^{\mathrm{T}}(x(k), u(k))\widetilde{w}_i^{\mathrm{T}}(k)\widetilde{w}_i(k) z_i(x(k), u(k)) \\
&\quad + \epsilon_{z_i}^2 - 2\eta_i e_i(k)\widetilde{w}_i^{\mathrm{T}}(k) K_i(k) - e_i^2(k), \\
\Delta V_i(k) &\leq |e_i(k)|^2 \|\eta K_i\|^2 - |e_i(k)|^2 \\
&\quad - |2\eta_i||e_i(k)|\|\widetilde{w}_i(k)\|\|K_i(k)\| + |\epsilon_{z_i}|^2 \\
&\quad + |2\epsilon_{z_i}|\|\widetilde{w}_i(k)\|\|z_i(x(k), u(k))\| \\
&\quad + \|\widetilde{w}_i(k)\|^2 \|z_i(x(k), u(k))\|^2.
\end{aligned}$$

Then $\Delta V_i(k) < 0$ when the following conditions hold [3]:

$$\begin{aligned}
|e_i(k)| &> \frac{|\epsilon_{z_i}|^2}{1 - \|\eta K_i\|^2} \equiv \kappa_1, \\
\|\widetilde{w}_i(k)\| &> \frac{|2\eta_i e_{i\max}|\|K_i(k)\|}{\|z_i(x(k), u(k))\|^2} - \frac{|2\epsilon_{z_i}|}{\|z_i(x(k), u(k))\|} \equiv \kappa_2.
\end{aligned}$$

Therefore, the solution of (4.9) and (4.10) is stable; hence the identification error and the RHONN weights are SGUUB [3].

Case 4.2. $\|w_i(k)\| < c_i$: Consider the same Lyapunov function candidate as in case 1 (4.10); if $K_i = 0$ this implies that $\Delta V_i(k) = 0$, then the identification error and the weights are bounded. □

Comment 4.1. As well as many feedback linearization like controllers [1], the neural block controller may present some singularities, due to the zero crossing of some adaptive parameters. To overcome the controller singularity problem in this chapter is included the constraint c_i, which allows to eliminate the controller singularities for specific weights zero-crossing [1].

4.2 Neural Block Controller Design

Consider the following special case of system (4.1):

$$\begin{aligned}
x(k+1) &= f(x(k)) + B(x(k))u(k) + d(k), \\
y(k) &= Cx(k),
\end{aligned} \tag{4.13}$$

where $x \in \Re^n$ is the state vector of the system, $u(k) \in \Re^m$ is the input vector, $y(k) \in \Re^p$ is the output vector, the vector $f(\cdot)$, the columns of $B(\cdot)$ and $d(\cdot)$ are smooth vector fields, and $d(\cdot)$ is a disturbance vector. By means

of nonsingular transformation [4, 7], system (4.13) can be represented in the block controllable form as follows:

$$\begin{aligned} x_i(k+1) &= f_i(\overline{x}_i(k)) + B_i(\overline{x}_i(k))x_{i+1}(k) + d_i(k), \\ x_r(k+1) &= f_r(x(k)) + B_r(x(k))u(k) + d_r(k), \\ y(k) &= x_1(k),\ i = 1, \ldots, r-1, \end{aligned} \tag{4.14}$$

where $x(k) = [\, x_1(k) \ldots x_i(k) \ldots x_r(k) \,]^\top$, $\overline{x}_i(k) = [\, x_1(k) \ldots x_i(k) \,]^\top$, $d(k) = [\, d_1(k) \ldots d_i(k) \ldots d_r(k) \,]^\top$, $i = 1, \ldots, r-1$, and the set of numbers $(n_1, \ldots, n_r)$, which define the structure of system (4.14), satisfy $n_1 \le n_2 \le \cdots \le n_r \le m$.

Define the following transformation:

$$\begin{aligned} \mathbf{z}_1(k) &= x_1(k) - x_1^{\mathrm{d}}(k), \\ \mathbf{z}_2(k) &= x_2(k) - x_2^{\mathrm{d}}(k), \\ &= x_2(k) - [B_1(x_1(k))]^{-1}(\mathbf{K}_1\mathbf{z}_1(k) - (f_1(x_1(k)) - d_1(k))), \\ \mathbf{z}_3(k) &= x_3(k) - x_3^{\mathrm{d}}(k), \\ &= x_3(k) - [B_2(x_2(k))]^{-1}(\mathbf{K}_2\mathbf{z}_2(k) - (f_2(x_2(k)) - d_2(k))), \\ &\vdots \\ \mathbf{z}_r(k) &= x_r(k) - x_r^{\mathrm{d}}(k), \end{aligned} \tag{4.15}$$

where $y_{\mathrm{d}}(k) = x_1^{\mathrm{d}}(k)$ is the desired trajectory for tracking; x_i^{d} is the desired value for x_i $(i = 1, 2, \cdots, r)$, which will be defined later; and $\mathbf{K}_i$ is a Schur matrix. Using (4.15), system (4.14) can be rewritten as

$$\begin{aligned} z_1(k+1) &= \mathbf{K}_1 z_1(k) + B_1 z_2(k), \\ &\vdots \\ z_{r-1}(k+1) &= \mathbf{K}_{r-1} z_{r-1}(k) + B_{r-1} z_r(k), \\ z_r(k+1) &= f_r(x(k)) + B_r(x(k))u(k) + d_r(k) - x_r^{\mathrm{d}}(k+1). \end{aligned} \tag{4.16}$$

To design the control law, we use the sliding mode block control technique. The manifold can be derived from the block control procedure, and a natural selection for the sliding manifold is $S_{\mathrm{D}}(k) = \mathbf{z}_r(k) = 0$. Thus, system (4.16) is represented, in the new variables, as

$$\begin{aligned} \mathbf{z}_1(k+1) &= \mathbf{K}_1 \mathbf{z}_1(k) + B_1 \mathbf{z}_2(k), \\ &\vdots \\ \mathbf{z}_{r-1}(k+1) &= \mathbf{K}_{r-1} \mathbf{z}_{r-1}(k) + B_{r-1} S_{\mathrm{D}}(k), \\ S_{\mathrm{D}}(k+1) &= f_r(x(k)) + B_r(x(k))u(k) + d_r(k) - x_r^{\mathrm{d}}(k+1). \end{aligned} \tag{4.17}$$

Once the sliding manifold is selected, the next step is to define $u(k)$ as

$$u(k) = \begin{cases} u_{\mathrm{eq}}(k) & \text{for} \quad \|u_{\mathrm{eq}}(k)\| \leq u_0, \\ u_0 \frac{u_{\mathrm{eq}}(k)}{\|u_{\mathrm{eq}}(k)\|} & \text{for} \quad \|u_{\mathrm{eq}}(k)\| > u_0, \end{cases} \tag{4.18}$$

where the equivalent control is calculated from $S_{\mathrm{D}}(k+1) = 0$ as

$$u_{\mathrm{eq}}(k) = [B_r(x(k))]^{-1}(-f_r(x(k)) + x_r^{\mathrm{d}}(k+1) - d_r(k)).$$

To this end, we present a stability analysis to prove that the closed-loop system (4.17)–(4.18) motion over the manifold is stable, which is the second main result of this chapter.

Theorem 4.2. *The control law* (4.18) *ensures the sliding manifold* $S_{\mathrm{D}}(k) = \mathbf{z}_r(k) = 0$ *is stable for system* (4.14).

Proof. Write the last subsystem of (4.17) as

$$S_{\mathrm{D}}(k+1) = S_{\mathrm{D}}(k) - x_r(k) + x_r^{\mathrm{d}}(k) + f_r(x^1(k)) + B_r u(k) + d_r(k) - x_r^{\mathrm{d}}(k+1).$$

Note that when $\|u_{\mathrm{eq}}(k)\| \leq u_0$, the equivalent control is applied, yielding motion on the sliding manifold $S_{\mathrm{D}}(k) = 0$. In the case of $\|u_{\mathrm{eq}}(k)\| > u_0$, the proposed control strategy is $u_0 \frac{u_{\mathrm{eq}}(k)}{\|u_{\mathrm{eq}}(k)\|}$, and the closed-loop system is

$$\begin{aligned} S_{\mathrm{D}}(k+1) &= S_{\mathrm{D}}(k) - x_r(k) + x_r^{\mathrm{d}}(k) + f_r(x^1(k)) + B_r u_0 \frac{u_{\mathrm{eq}}(k)}{\|u_{\mathrm{eq}}(k)\|} \\ &\quad + d_r(k) - x_r^{\mathrm{d}}(k+1), \\ &= (S_{\mathrm{D}}(k) - x_r(k) + x_r^{\mathrm{d}}(k) + f_r(x^1(k)) + d_r(k) - x_r^{\mathrm{d}}(k+1)) \\ &\quad \times \left(1 - \frac{u_0}{\|u_{\mathrm{eq}}(k)\|}\right). \end{aligned}$$

Along any solution of the system, the Lyapunov candidate function $V(k) = S_{\mathrm{D}}^{\top}(k) S_{\mathrm{D}}(k)$ gives

$$\begin{aligned} \Delta V(k) &= S_{\mathrm{D}}^{\top}(k+1) S_{\mathrm{D}}(k+1) - S_{\mathrm{D}}^{\top}(k) S_{\mathrm{D}}(k), \\ &= \left[(S_{\mathrm{D}}(k) + f_{\mathrm{s}}(k)) \left(1 - \frac{u_0}{\|u_{\mathrm{eq}}(k)\|}\right)\right]^{\top} \\ &\quad \times (S_{\mathrm{D}}(k) + f_{\mathrm{s}}(k)) \left(1 - \frac{u_0}{\|u_{\mathrm{eq}}(k)\|}\right) - S_{\mathrm{D}}^{\top}(k) S_{\mathrm{D}}(k), \\ &\leq \left[\|S_{\mathrm{D}}(k) + f_s(k)\| \left(1 - \frac{u_0}{\|u_{eq}(k)\|}\right)\right]^{\top} \\ &\quad \times \|S_{\mathrm{D}}(k) + f_s(k)\| \left(1 - \frac{u_0}{\|u_{\mathrm{eq}}(k)\|}\right) - \|S_{\mathrm{D}}(k)\|^2. \end{aligned}$$

Then

$$\begin{aligned}\Delta V(k) &\leq \left[\|S_{\mathrm{D}}(k)+f_{\mathrm{s}}(k)\|-\frac{u_0}{\|B_r^{-1}\|}\right]^{\top}\left(\|S_{\mathrm{D}}(k)+f_{\mathrm{s}}(k)\|-\frac{u_0}{\|B_r^{-1}\|}\right)\\ &\quad -\|S_{\mathrm{D}}(k)\|^2,\\ &\leq \left(\|S_{\mathrm{D}}(k)+f_{\mathrm{s}}(k)\|-\frac{u_0}{\|B_r^{-1}\|}\right)^2-\|S_{\mathrm{D}}(k)\|^2,\\ &\leq \|S_{\mathrm{D}}(k)\|^2+2\|S_{\mathrm{D}}(k)\|^2 f_{\mathrm{s}}(k)-2u_0\frac{\|S_{\mathrm{D}}(k)\|}{\|B_r^{-1}\|}+\|f_{\mathrm{s}}(k)\|^2\\ &\quad -2u_0\frac{\|f_{\mathrm{s}}(k)\|}{\|B_r^{-1}\|}+\frac{u_0^2}{\|B_r^{-1}\|^2}-\|S_{\mathrm{D}}(k)\|^2,\\ &\leq -2\|S_{\mathrm{D}}(k)\|\left(\frac{u_0}{\|B_r^{-1}\|}-\|f_{\mathrm{s}}(k)\|\right)+\left(\frac{u_0}{\|B_r^{-1}\|}-\|f_{\mathrm{s}}(k)\|\right)^2,\end{aligned}$$

where $f_{\mathrm{s}}(k) = -x_r(k)+x_{\mathrm{d}}(k)+f_r(x(k))+d_r(k)-x_r^{\mathrm{d}}(k+1)$, and if $\|B_r^{-1}\|\|f_{\mathrm{s}}(k)\| \leq u_0 \leq \|B_r^{-1}\|(2\|S_{\mathrm{D}}(k)\|+\|f_{\mathrm{s}}(k)\|)$ holds, then $\Delta V(k) \leq 0$ [4]; hence $\|S_{\mathrm{D}}(k)\|$ and $\|u_{\mathrm{eq}}(k)\|$ both decreases monotonically. Note that $\|B_r^{-1}\|(2\|S_{\mathrm{D}}(k)\|+\|f_{\mathrm{s}}(k)\|) \geq u_0$ is a greater bound than that established by the current case, i.e., $\|u_{\mathrm{eq}}(k)\| > u_0$, due to the fact that $\|B_r^{-1}\|(2\|S_{\mathrm{D}}(k)\|+\|f_{\mathrm{s}}(k)\|) \geq \|u_{\mathrm{eq}}(k)\|$. Therefore, the only condition drawn from the Lyapunov analysis is $\|B_r^{-1}\|\|f_{\mathrm{s}}(k)\| \leq u_0$ [4].

The proposed control scheme is shown in Fig. 4.1. To this end, the third main result of this chapter is the following:

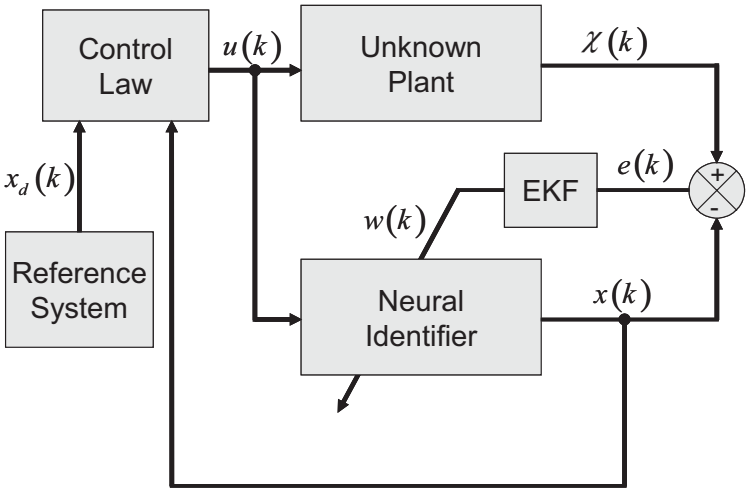

Fig. 4.1. Neural block control scheme

Proposition 4.1. *Given a desired output trajectory $y_{\mathrm{d}} = x_r^{\mathrm{d}}$, a dynamic system with output y, and a neural network with output y_n, then it is possible to establish the following inequality [1]:*

$$\|y_{\mathrm{d}} - y\| \leq \|y_n - y\| + \|y_{\mathrm{d}} - y_n\|,$$

where $y_{\mathrm{d}} - y$ is the system output tracking error, $y_n - y$ is the output identification error, and $y_{\mathrm{d}} - y_n$ is the RHONN output tracking error.

Based on this proposition, it is possible to divide the tracking error in two parts [1]:

1. Minimization of $y_n - y$, which can be achieved by the proposed online identification algorithm (4.1) on the basis of Theorem 4.1.
2. Minimization of $y_{\mathrm{d}} - y_n$, for which a tracking algorithm is developed on the basis of the neural identifier (4.2). This can be reached by designing a control law based on the RHONN model. To design such controller we propose to use the NBC methodology [1, 4].

Comment 4.2. Proposition 4.1 can be seen as a special case $C = I$ discussed in detail in Chap. 6.

4.3 Applications

In this section we apply the above developed scheme (Fig. 4.1) to control a three-phase induction motor, for which model is described in Chap. 3.

4.3.1 Neural Network Identification

The RHONN proposed for this application is as follows:

$$\begin{aligned}
\widehat{x}_1(k+1) &= w_{11}(k)S(\omega(k)) + w_{12}(k)S(\omega)S(\psi^{\beta}(k))i^{\alpha}(k) \\
&\quad + w_{13}(k)S(\omega)S(\psi^{\alpha}(k))i^{\beta}(k), \\
\widehat{x}_2(k+1) &= w_{21}(k)S(\omega(k))S(\psi^{\beta}(k)) + w_{22}(k)i^{\beta}(k), \\
\widehat{x}_3(k+1) &= w_{31}(k)S(\omega(k))S(\psi^{\alpha}(k)) + w_{32}(k)i^{\alpha}(k), \\
\widehat{x}_4(k+1) &= w_{41}(k)S(\psi^{\alpha}(k)) + w_{42}(k)S(\psi^{\beta}(k)) \\
&\quad + w_{43}(k)S(i^{\alpha}(k)) + w_{44}(k)u^{\alpha}(k), \\
\widehat{x}_5(k+1) &= w_{51}(k)S(\psi^{\alpha}(k)) + w_{52}(k)S(\psi^{\beta}(k)) \\
&\quad + w_{53}(k)S(i^{\beta}(k)) + w_{54}(k)u^{\beta}(k).
\end{aligned} \tag{4.19}$$

The training is performed online, using a series–parallel configuration. All the NN states are initialized in a random way as well as the weights vectors. It is an important remark that the initial conditions of the plant are completely different from the initial conditions for the NN.

4.3.2 Neural Block Controller Design

Given full state measurements, the control objective is to develop velocity and flux amplitude tracking for the discrete-time induction motor model (4.19), using the discrete-time control algorithm developed above. Let us define the following states as

$$x^1(k) = \begin{bmatrix} \widehat{x}_1(k) - \omega_r(k) \\ \Psi(k) - \Psi_r(k) \end{bmatrix}, \quad x^2(k) = \begin{bmatrix} i^{\alpha}(k) \\ i^{\beta}(k) \end{bmatrix}, \tag{4.20}$$

where $\Psi(k) = \widehat{x}_2^2(k) + \widehat{x}_3^2(k)$ is the rotor flux identifying magnitude, $\Psi_{\mathrm{r}}(k)$ and $\omega_{\mathrm{r}}(k)$ are reference signals. Then

$$\begin{aligned} \Psi(k+1) = {} & w_{21}^2(k) S^2(\omega(k)) S^2(\psi^{\beta}(k)) + w_{22}^2(k) i^{\beta^2}(k) + w_{32}^2(k) i^{\alpha^2}(k) \\ & + w_{31}^2(k) S^2(\omega(k)) S^2(\psi^{\alpha}(k)) \\ & + 2 w_{21}(k) S(\omega(k)) S(\psi^{\beta}(k)) w_{22}(k) i^{\beta}(k) \\ & + 2 w_{31}(k) S(\omega(k)) S(\psi^{\alpha}(k)) w_{32}(k) i^{\alpha}(k). \end{aligned}$$

Using (4.20), (4.19) can be represented in the block control form consisting of two blocks

$$\begin{aligned} x^1(k+1) &= f_1(x^1(k)) + B_1(x^1(k)) x^2(k), \\ x^2(k+1) &= f_2(x^1(k), x^2(k)) + B_2(k) u(k), \end{aligned} \tag{4.21}$$

with $u(k) = [\, u^{\alpha}(k) \; u^{\beta}(k) \,]^{\top}$ and

$$\begin{aligned} f_1(x^1(k)) &= \begin{bmatrix} w_{11}(k) S(\omega(k)) - \omega_r(k+1) \\ f_{11}(k) \end{bmatrix}, \\ f_{11}(k) &= w_{21}^2(k) S^2(\omega(k)) S^2(\psi^{\beta}(k)) + w_{31}^2(k) S^2(\omega(k)) S^2(\psi^{\alpha}(k)) \\ &\quad + w^2 I_m^2(k) - \Psi_r(k+1), \\ I_m(k) &= \sqrt{w_{22}^2(k) i^{\alpha^2}(k) + w_{32}^2(k) i^{\beta^2}(k)}, \\ B_1(x^1(k)) &= \begin{bmatrix} b_{11}(k) \; b_{12}(k) \\ b_{21}(k) \; b_{22}(k) \end{bmatrix}, \\ b_{11}(k) &= w_{12}(k) S(\omega) S(\psi^{\beta}(k)), \\ b_{12}(k) &= w_{13}(k) S(\omega) S(\psi^{\alpha}), \\ b_{21}(k) &= 2 w_{31}(k) w_{32}(k) S(\omega(k)) S(\psi^{\alpha}(k)), \\ b_{22}(k) &= 2 w_{21}(k) w_{22}(k) S(\omega(k)) S(\psi^{\beta}(k)), \\ f_2(x^2(k)) &= \begin{bmatrix} f_{21}(k) \\ f_{22}(k) \end{bmatrix}, \quad B_2(k) = \begin{bmatrix} w_{44}(k) & 0 \\ 0 & w_{54}(k) \end{bmatrix}, \\ f_{21}(k) &= w_{41}(k) S(\psi^{\alpha}(k)) + w_{42}(k) S(\psi^{\beta}(k)) + w_{43}(k) S(i^{\alpha}(k)), \\ f_{22}(k) &= w_{51}(k) S(\psi^{\alpha}(k)) + w_{52}(k) S(\psi^{\beta}(k)) + w_{53}(k) S(i^{\beta}(k)). \end{aligned}$$

Applying the block control technique, we define the following vector $\mathbf{z}_1(k) = x^1(k)$. Then

$$\mathbf{z}_1(k+1) = f_1(x^1(k)) + B_1(x^1(k))x^2(k) = \mathbf{K}\mathbf{z}_1(k), \tag{4.22}$$

where $\mathbf{K} = \text{diag}\{\mathbf{k}_1, \mathbf{k}_2\}$, with $|\mathbf{k}_i| < 1$ $(i = 1, 2)$; then the desired value $x^{2\text{d}}(k)$ of $x^2(k)$ is calculated from (4.22) as

$$x^{2\text{d}}(k) = B_1^{-1}(x^1(k))[-f_1(x^1(k)) + \mathbf{K}\mathbf{z}_1(k)].$$

It is desired that $x^2(k) = x^{2\text{d}}(k)$. In this way, it is defined as a second new error vector

$$\mathbf{z}_2(k) = x^2(k) - x^{2d}(k).$$

Then

$$\mathbf{z}_2(k+1) = f_3(x^1(k)) + B_2(k)u(k),$$

with

$$f_3(x^1(k)) = f_2(x^2(k)) - B_1^{-1}(x^1(k+1))[-f_1(x^1(k+1)) + \mathbf{K}\mathbf{z}_1(k+1)].$$

Let us select the manifold for the sliding mode as $S_{\text{D}}(k) = \mathbf{z}_2(k)$. To design a control law, a discrete-time sliding mode version is implemented as

$$u(k) = \begin{cases} u_{\text{eq}}(k) & \text{if } \|u_{\text{eq}}(k)\| \leq u_0, \\ u_0 \frac{u_{\text{eq}}(k)}{\|u_{\text{eq}}(k)\|} & \text{if } \|u_{\text{eq}}(k)\| > u_0, \end{cases}$$

where $u_{\text{eq}}(k) = -B_2^{-1}(k)f_3(x^1(k))$ is calculated from $S_{\text{D}}(k) = 0$ and u_0 is the control resources that bound the control. This system represents the sliding mode dynamics, which achieves the control objectives. It is an obvious fact that the proposed control $u(k)$ depends on $i^{\alpha^2}(k)$ and $i^{\beta^2}(k)$, which appears in $f_1(\bullet)$, making the system insolvable [4]. To overcome this problem is designed an observer only with current measurements for the new variable $I_m(k)$ [4]. Because of the varying time of RHONN weights, we need to guarantee that $B_1(\bullet)$ and $B_2(\bullet)$ are not singular; then it is necessary to avoid the zero-crossing of the weights $w_{13}(k)$, $w_{22}(k)$, $w_{32}(k)$, $w_{44}(k)$, and $w_{54}(k)$, which are the so-called controllability weights [1]. It is important to remark that in this application only the weights $w_{44}(k)$ and $w_{54}(k)$ tend to cross zero.

4.3.3 Simulation Results

Simulations are performed for the system (4.19), using the parameters given in Table 4.1.

For simulations the full state measurement assumption is necessary [4]. However, rotor fluxes measurement is a difficult task. Here, the reduced order nonlinear observer designed in Chap. 2 is used to perform the simulation. The

Table 4.1. Induction motor parameters

Parameter	Value	Description
R_{s}	$14\,\Omega$	Stator resistance
L_{s}	400 mH	Stator inductance
M	377 mH	Mutual inductance
R_{r}	$10.1\,\Omega$	Rotor resistance
L_{r}	412.8 mH	Rotor inductance
n_{p}	2	Number of pole pairs
J	$0.01\,\mathrm{Kg\,m^2}$	Moment of inertia
ω_{n}	$168.5\,\mathrm{rad\,s^{-1}}$	Nominal speed
$T_{L_{\mathrm{n}}}$	1.1 N m	Nominal load
T	0.0001 s	Sampling period

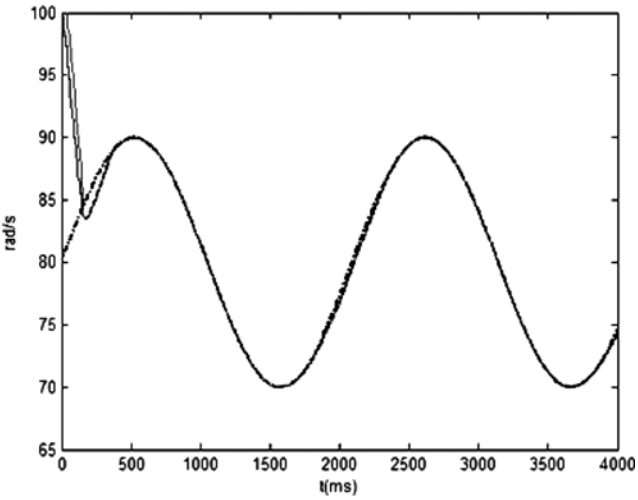

Fig. 4.2. Tracking performance $\omega(k)$ (*solid line*), $x_1(k)$ (*dash-dot line*), and $\omega_{\mathrm{r}}(k)$ (*dashed line*)

tracking results are presented in Figs. 4.2 and 4.3. There the tracking and identification performance can be verified for the two plant outputs. Figure 4.4 displays the load torque applied as an external disturbance. Figure 4.5 presents the parametric variation introduced in the rotor resistance (R_{r}) as a variation of $1\,\Omega\,s^{-1}$. Figure 4.6 shows the weights evolution. Figures 4.7 and 4.8 portray the fluxes and their estimates.

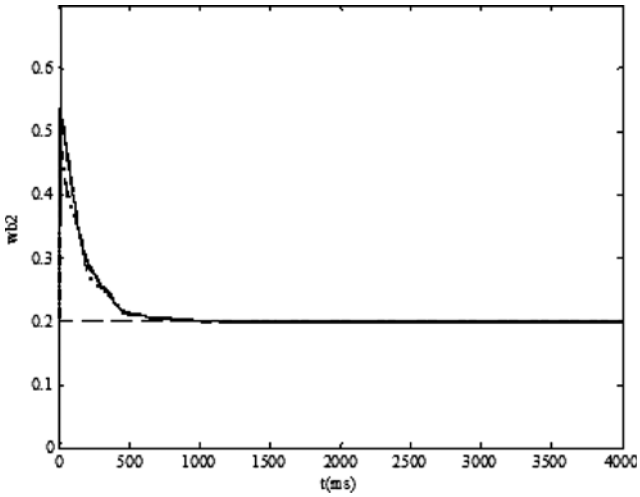

Fig. 4.3. Tracking performance $\Psi(k)$ (*solid line*), $x_2^2 + x_3^2$ (*dash-dot line*), and $\Psi_{\mathrm{d}}(k)$ (*dashed line*)

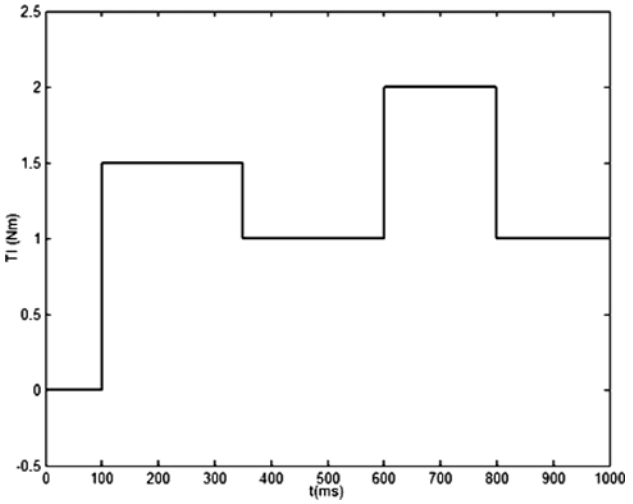

Fig. 4.4. Load torque $T_{\mathrm{L}}(k)$

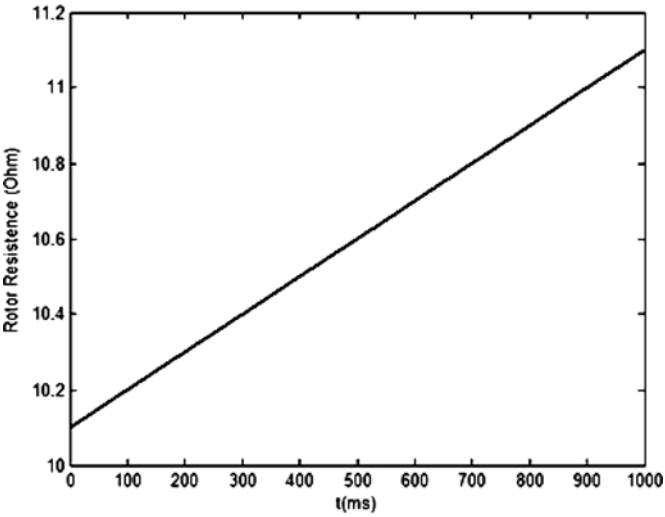

Fig. 4.5. Rotor resistance variation (R_r)

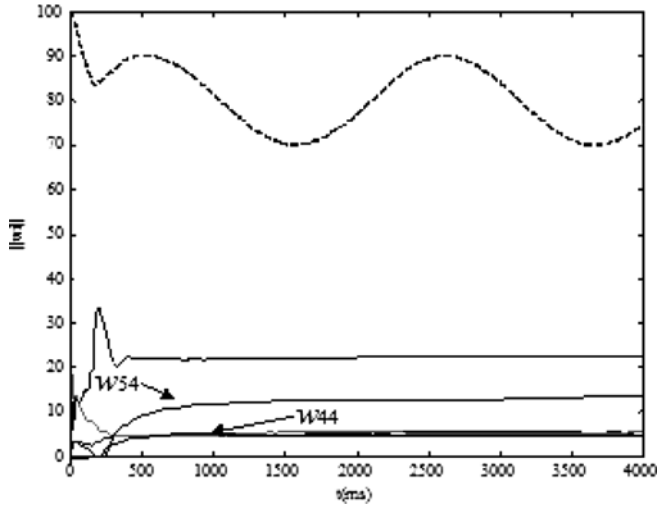

Fig. 4.6. Weights evolution

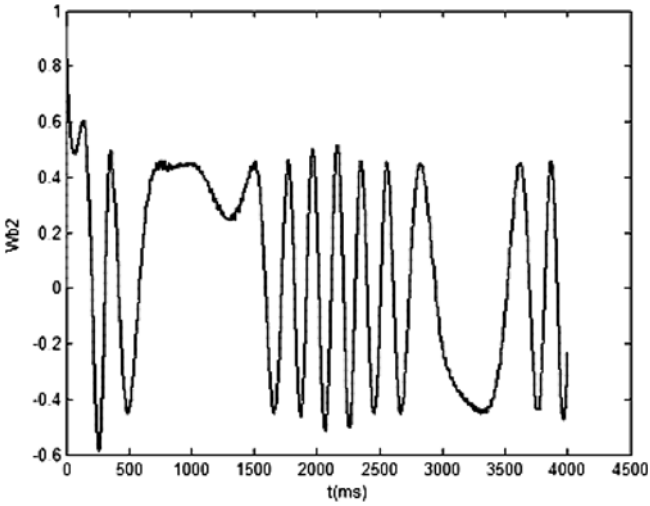

Fig. 4.7. Time evolution of $\psi^{\alpha}(k)$ and its estimate (real in *solid line* and estimated in *dashed line*)

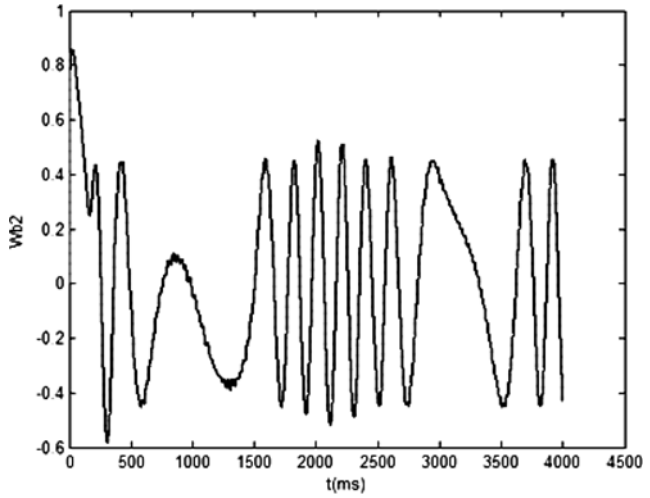

Fig. 4.8. Time evolution of $\psi^{\beta}(k)$ and its estimate (real in *solid line* and estimated in *dashed line*)

4.4 Conclusions

This chapter has presented the application of recurrent high order neural networks to design a block control algorithm for a class of discrete-time nonlinear systems. The RHONN is used to perform the system identification; the training of the neural networks is performed online using an extended Kalman filter in a series–parallel configuration. The boundness of the identification error is established on the basis of the Lyapunov approach. The proposed training algorithm avoids singularities in the control law due to weight zero-crossings. Simulation results illustrate the robustness of the proposed control methodology with respect to external disturbances as well as parametric variations.

5

Discrete-Time Neural Observers

This chapter presents the design of an adaptive recurrent neural observer for nonlinear systems, whose mathematical model is assumed to be unknown. The observer is based on a recurrent high order neural network (RHONN), which estimates the state vector of the unknown plant dynamics and it has a Luenberger structure. The learning algorithm for the RHONN is implemented using an extended Kalman filter (EKF). The respective stability analysis, on the basis of the Lyapunov approach, is included for the observer trained with an EKF and simulation results are included to illustrate the applicability of the proposed scheme.

Many of the nonlinear control publications assume the complete accessibility of the system state; this is not always possible. For that reason the solution of the nonlinear state estimation problem is a very important topic for nonlinear control [9].

The state estimation problem has received much attention by many authors, who have obtained interesting results in different directions. Most of those results need the use of a special nonlinear transformation [7] or a linearization technique [1, 4]. Such approaches can be considered as a relatively simple method to construct nonlinear observers; however, they do not consider uncertainties. In practice, we work in presence of external and internal uncertainties. Observers that have a good performance even in the presence of model and disturbance uncertainties are called robust; their design process is too complex [11].

All the approaches mentioned above need the previous knowledge of the plant dynamics. Recently, other kind of observer has emerged: neural observers [3, 5, 6, 9, 10], for unknown plant dynamics.

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 45–57 (2008)
www.springerlink.com

5.1 Recurrent High Order Neural Observer (RHONO) Design

In this section, we consider to estimate the state of a discrete-time nonlinear system, which is assumed to be observable, given by

$$\begin{aligned} x(k+1) &= F(x(k), u(k)) + d(k), \\ y(k) &= Cx(k), \end{aligned} \tag{5.1}$$

where $x \in \Re^n$ is the state vector of the system, $u(k) \in \Re^m$ is the input vector, $y(k) \in \Re^p$ is the output vector, $C \in \Re^{p \times n}$ is a known output matrix, $d(k) \in \Re^n$ is a disturbance vector, and $F(\bullet)$ is a smooth vector field and $F_i(\bullet)$ its entries; hence (5.1) can be rewritten as

$$\begin{aligned} x(k) &= \left[x_1(k) \ldots x_i(k) \ldots x_n(k) \right]^\top, \\ d(k) &= \left[d_1(k) \ldots d_i(k) \ldots d_n(k) \right]^\top, \\ x_i(k+1) &= F_i(x(k), u(k)) + d_i(k), \qquad i = 1, \cdots, n, \\ y(k) &= Cx(k). \end{aligned} \tag{5.2}$$

For system (5.2), we propose a Luenberger neural observer (RHONO) with the following structure:

$$\begin{aligned} \widehat{x}(k) &= \left[\widehat{x}_1(k) \ldots \widehat{x}_i(k) \ldots \widehat{x}_n(k) \right]^\top, \\ \widehat{x}_i(k+1) &= w_i^\top z_i(\widehat{x}(k), u(k)) + g_i e(k), \\ \widehat{y}(k) &= C\widehat{x}(k), \qquad i = 1, \cdots, n, \end{aligned} \tag{5.3}$$

with $g_i \in \Re^p$, $z_i(x(k), u(k))$ as defined in (2.8).

As discussed in [8], the general discrete-time nonlinear system (5.1), which is assumed to be observable, can be approximated by the following discrete-time RHONN parallel representation:

$$x(k+1) = W^{*\top} z(\widehat{x}(k), u(k)) + \epsilon_z, \tag{5.4}$$

or in the single output form

$$x_i(k+1) = w_i^{*\top} z_i(\widehat{x}(k), u(k)) + \epsilon_{z_i}, \qquad i = 1, \cdots, n, \tag{5.5}$$

where x_i is the ith plant state, ϵ_{z_i} is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [8]. Let us assume that there exists ideal weights vector $w_i^* \in \Re^{L_i}$ such that $\|\epsilon_{z_i}\|$ can be minimized on a compact set $\Omega_{z_i} \subset \Re^{L_i}$. The ideal weight vector w_i^* is an artificial quantity required for analytical purpose [8]. In general it is assumed that this vector exists and it is constant but unknown. Let us define

its estimate as w_i. Then the weights estimation $\widetilde{w}_i(k)$ error is defined as

$$\widetilde{w}_i(k) = w_i^* - w_i(k). \tag{5.6}$$

Since w_i^* is constant, then

$$\widetilde{w}_i(k+1) - \widetilde{w}_i(k) = w_i(k) - w_i(k+1), \quad \forall k \in 0 \cup \mathbb{Z}^+.$$

Then the weight vectors are updated online with a decoupled EKF, described by

$$\begin{aligned} w_i(k+1) &= w_i(k) + \eta_i K_i(k) e(k), \\ K_i(k) &= P_i(k) H_i(k) M_i(k), \\ P_i(k+1) &= P_i(k) - K_i(k) H_i^\top(k) P_i(k) + Q_i(k), \quad i = 1, \cdots, n, \end{aligned} \tag{5.7}$$

with

$$M_i(k) = [R_i(k) + H_i^\top(k) P_i(k) H_i(k)]^{-1}. \tag{5.8}$$

The output error is defined by

$$e(k) = y(k) - \widehat{y}(k), \tag{5.9}$$

and the state estimation error as

$$\widetilde{x}(k) = x(k) - \widehat{x}(k). \tag{5.10}$$

Then the dynamics of (5.10) can be expressed as

$$\widetilde{x}_i(k+1) = \widetilde{w}_i(k) z_i(\widehat{x}(k), u(k)) + \epsilon'_{z_i} - g_i C \widetilde{x}(k), \tag{5.11}$$

with $\epsilon'_{z_i} = \epsilon_{z_i} + d_i(k)$. On the other hand, the dynamics of (5.6) is

$$\widetilde{w}_i(k+1) = \widetilde{w}_i(k) - \eta_i K_i(k) e(k). \tag{5.12}$$

Considering (5.7)–(5.12), we establish the main result of this chapter in the following theorem.

Theorem 5.1. *For the system* (5.2) *the RHONO* (5.3)*, trained with the EKF-based algorithm* (5.7)*, ensures that the estimation error* (5.10) *and the output error* (5.9) *are semiglobally uniformly ultimately bounded (SGUUB); moreover, the RHONO weights remain bounded.*

Proof. Consider the Lyapunov function candidate.

$$\begin{aligned} V_i(k) &= \widetilde{w}_i(k) P_i(k) \widetilde{w}_i(k) + \widetilde{x}_i(k) P_i(k) \widetilde{x}_i(k), \\ \Delta V_i(k) &= V(k+1) - V(k), \\ &= \widetilde{w}_i(k+1) P_i(k+1) \widetilde{w}_i(k+1) + \widetilde{x}_i(k+1) P_i(k+1) \widetilde{x}_i(k+1) \\ &\quad - \widetilde{w}_i(k) P_i(k) \widetilde{w}_i(k) - \widetilde{x}_i(k) P_i(k) \widetilde{x}_i(k). \end{aligned} \tag{5.13}$$

Using (5.7) and (5.6) in (5.13)

$$\begin{aligned}\Delta V_i(k) = &[\widetilde{w}_i(k) - \eta_i K_i(k) e(k)]^{\mathrm{T}}[A_i(k)][\widetilde{w}_i(k) - \eta_i K_i(k) e(k)] \\ &+ [f(k) - g_i C\widetilde{x}(k)]^{\mathrm{T}}[A_i(k)][f(k) - g_i C\widetilde{x}(k)] \\ &- \widetilde{w}_i(k) P_i(k) \widetilde{w}_i(k) - \widetilde{x}_i(k) P_i(k) \widetilde{x}_i(k), \end{aligned} \tag{5.14}$$

with

$$\begin{aligned} A_i(k) &= P_i(k) - D_i(k) + Q_i D_i(k) = K_i(k) H_i^{\top}(k) P_i(k), \\ f(k) &= \widetilde{w}_i(k) z_i(\widehat{x}(k), u(k)) + \epsilon'_{z_i}, \end{aligned}$$

Then, (5.14) can be expressed as

$$\begin{aligned}\Delta V_i(k) = &\widetilde{w}_i^{\mathrm{T}}(k) P_i(k) \widetilde{w}_i(k) - \widetilde{w}_i^{\mathrm{T}}(k)[B_i(k)]\widetilde{w}_i(k) \\ &+ \eta^2 \widetilde{x}^{\mathrm{T}}(k) C^{\mathrm{T}} K^{\mathrm{T}}[A_i(k)] K_i(k) C\widetilde{x}(k) \\ &+ f^{\mathrm{T}}(k) P_i(k) f(k) - f^{\mathrm{T}}(k)[B_i(k)] f(k) \\ &+ \widetilde{x}^{\mathrm{T}}(k) C^{\mathrm{T}} g_i^{\mathrm{T}}[A_i(k)] g_i C\widetilde{x}(k) \\ &- \widetilde{w}_i^{\mathrm{T}}(k) P_i(k) \widetilde{w}_i(k) - \widetilde{x}_i^{\mathrm{T}}(k) P_i(k) \widetilde{x}_i(k), \end{aligned}$$

$$\begin{aligned}\Delta V_i(k) \leq &\|\widetilde{x}(k)\|^2 \|\eta K_i C\|^2 \|A_i(k)\| - \|\widetilde{x}(k)\|^2 \|g_i C\|^2 \|A_i(k)\| \\ &- \|\widetilde{x}(k)\|^2 P_i(k) - \|\widetilde{w}_i(k)\|^2 \|B_i(k)\| + |\epsilon'_{z_i}|^2 \|A_i(k)\| \\ &+ 2\|\widetilde{w}_i(k)\| \|z_i(\widehat{x}(k), u(k))\| |\epsilon'_{z_i}| \|A_i(k)\| \\ &+ \|\widetilde{w}_i(k)\|^2 \|z_i(\widehat{x}(k), u(k))\|^2 \|A_i(k)\|, \end{aligned}$$

with $B_i(k) = D_i(k) - Q_i$,

$$\Delta V_i(k) \leq -\|\widetilde{x}(k)\|^2 E_i(k) - \|\widetilde{w}_i(k)\|^2 F_i(k) + |\epsilon'_{z_i}|^2 \|A_i(k)\| + 2G_i(k),$$

with

$$\begin{aligned} E_i(k) &= P_i(k) - \|\eta K_i C\|^2 \|A_i(k)\| - \|g_i C\|^2 \|A_i(k)\|, \\ F_i(k) &= \|B_i(k)\| - \|z_i(\widehat{x}(k), u(k))\|^2 \|A_i(k)\|, \\ G_i(k) &= \|w_i^* - w_{i\max}\| \|z_i(\widehat{x}(k), u(k))\| |\epsilon'_{z_i}| \|A_i(k)\|. \end{aligned}$$

Then $\Delta V_i(k) < 0$ when

$$\|\widetilde{x}(k)\| > \sqrt{\frac{|\epsilon'_{z_i}|^2 \|A_i(k)\| + 2G_i(k)}{E_i(k)}} \equiv \kappa_1$$

or

$$\|\widetilde{w}_i(k)\| > \sqrt{\frac{|\epsilon'_{z_i}|^2 \|A_i(k)\| + 2G_i(k)}{F_i(k)}} \equiv \kappa_2.$$

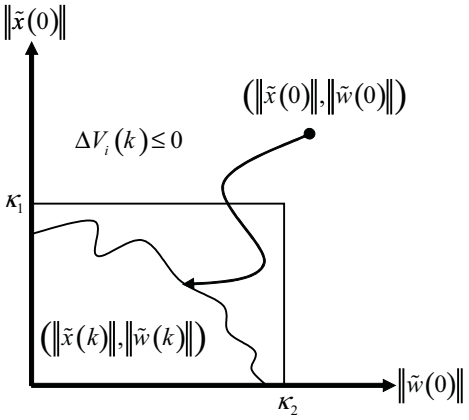

Fig. 5.1. Graphical illustration of Theorem 5.1

Therefore, the solution of (5.11) and (5.12) is stable; hence the estimation error and the RHONO weights are SGUUB [3] (A graphical illustration is shown in Fig. 5.1). Considering (5.3) and (5.9) it is easy to see that the output error has an algebraic relation with $\widetilde{x}(k)$ for that reason if $\widetilde{x}(k)$ is bounded $e(k)$ is bounded too.

$$e(k) = C\widetilde{x}(k),$$
$$\|e(k)\| = \|C\|\|\widetilde{x}(k)\|.$$

5.2 Applications

5.2.1 RHONO for the Van der Pol Oscillator

In this section, the neural observer is applied to a modified Van der Pol oscillator, whose nonlinear dynamics is represented by the following equation [12]:

$$\begin{aligned} x_1(k+1) &= x_1(k) + Tx_2(k) + d_1(k), \\ x_2(k+1) &= x_2(k) + T(-\xi(x_1^2(k)-1)x_2(k)) + T(-x_1(k)+u(k)) + d_2(k), \\ y(k) &= x_1(k), \\ d_1(k) &= 0.1\sin(k), \\ d_2(k) &= 0.1\cos(k), \end{aligned} \tag{5.15}$$

where variables $x \in \Re^2$, $u \in \Re$, and $y \in \Re$ are the state, input, and output of the system, respectively; $d_1(k)$ and $d_2(k)$ are bounded external disturbances; T is the sampling period, which is fixed at 0.1 s and ξ is a parameter for which nominal value is equal to 2.

Simulation Results

To estimate the state x_2, we use the RHONO (5.3) with $n = 2$ trained with the EKF (5.7).

$$\begin{aligned}
\widehat{x}_1(k+1) &= w_{11}(k)S^2(\widehat{x}_1(k)) + w_{12}(k)S(\widehat{x}_1(k))S(\widehat{x}_2(k)) \\
&\quad + w_{13}(k)S^2(\widehat{x}_2(k)) + w_{14}(k)S^4(\widehat{x}_2(k)) + g_1 e(k), \\
\widehat{x}_2(k+1) &= w_{21}(k)S^2(\widehat{x}_1(k)) + w_{22}(k)S^3(\widehat{x}_2(k)) \\
&\quad + w_{23}(k)S(\widehat{x}_1(k))S(\widehat{x}_2(k)) \\
&\quad + w_{24}(k)S^2(\widehat{x}_2(k)) + w_{25}(k)S^3(u(k)) + g_2 e(k), \\
\widehat{y}(k) &= \widehat{x}_1(k), \\
u(k) &= \cos\left(\frac{2\pi k}{25}\right).
\end{aligned} \tag{5.16}$$

The training is performed online, using a parallel configuration as displayed in Fig. 5.2. All the NN states are initialized in a random way. The associated covariances matrices are initialized as diagonals, and the nonzero elements are $P_1(0) = P_2(0) = 10{,}000$; $Q_1(0) = Q_2(0) = 500$; and $R_1(0) = R_2(0) = 10{,}000$, respectively. The simulation results are presented in Figs. 5.3 and 5.4. They display the time evolution of the estimated states $x_1(k)$ and $x_2(k)$, respectively. Figure 5.5. shows the estimation errors. Figure 5.6 displays the parametric variation for ξ increment, and Fig. 5.7 portrays the bounded external disturbances.

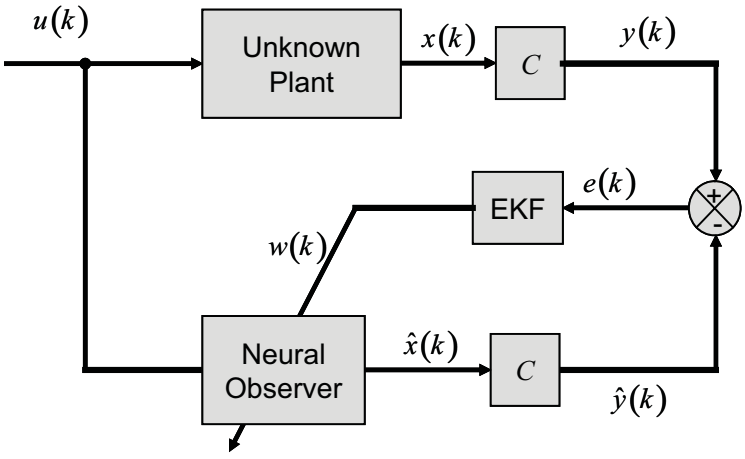

Fig. 5.2. Neural observer scheme

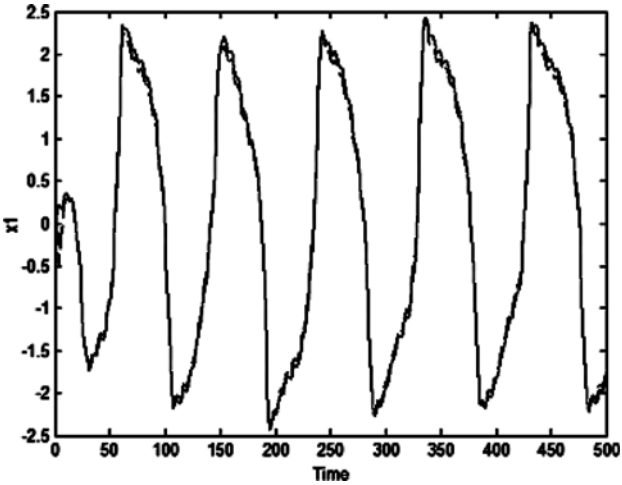

Fig. 5.3. Time evolution of the state $x_1(k)$ (*solid line*) and its estimated $\widehat{x}_1(k)$ (*dashed line*)

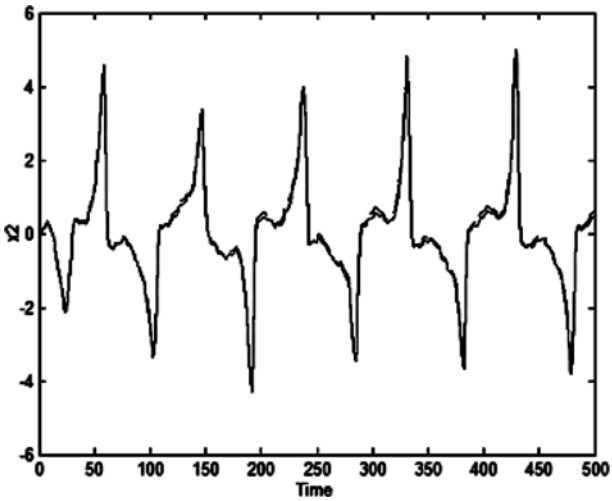

Fig. 5.4. Time evolution of the state $x_2(k)$ (*solid line*) and its estimated $\widehat{x}_2(k)$ (*dashed line*)

5.2.2 RHONO for Induction Motors

Literature in induction motors control is extensive, including field oriented controller, VSC sliding mode controller, passivity-based controllers, and more recently dynamic feedback linearization method, but many of the strategies

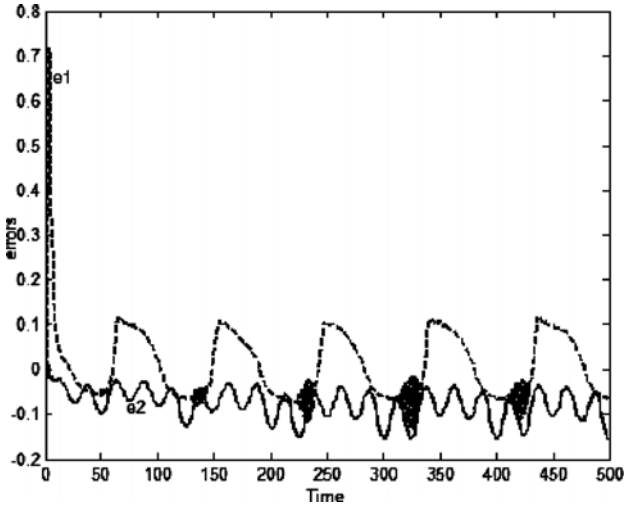

Fig. 5.5. Estimation errors $\widetilde{x}_1(k)$ (*dashed line*) and $\widetilde{x}_2(k)$ (*solid line*)

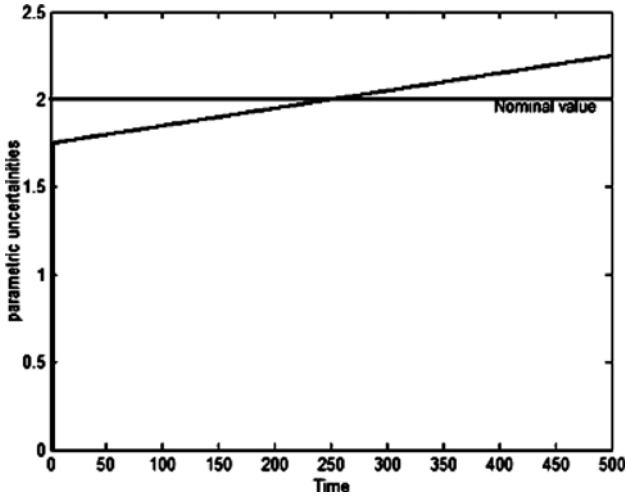

Fig. 5.6. Uncertainties in parameter ξ

mentioned above assumed all the state is available for measurement and all motor parameters are known; additionally most of those works were developed for continuous-time systems ([2] and references therein).

In this section, we propose the use of the RHONO developed in Sect. 5.1 to estimate the state of the discrete-time induction motor model presented in Sect. 3.2.

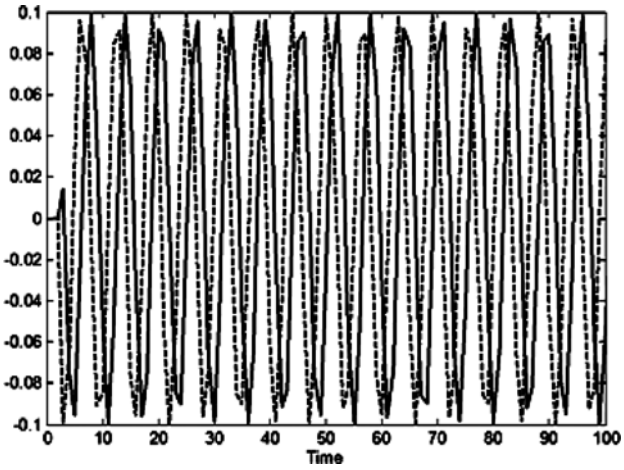

Fig. 5.7. Disturbances $d_1(k)$ (*solid line*) and $d_2(k)$ (*dashed line*)

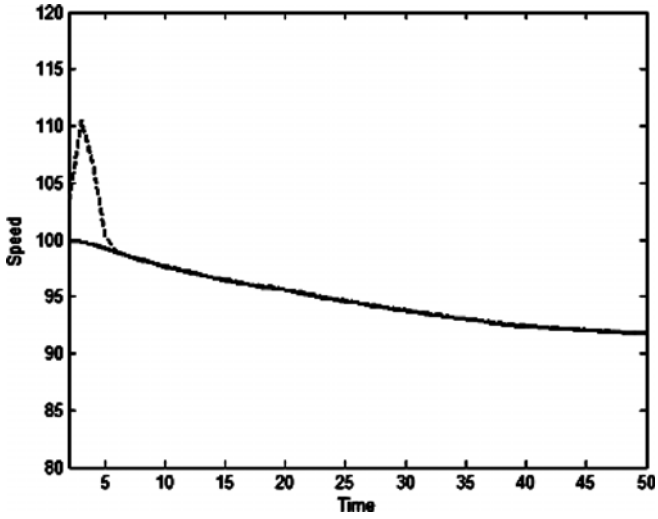

Fig. 5.8. Angular speed ω (*solid line*) and its estimated $\widehat{x}_1$ (*dashed line*)

Simulation Results

Now, we apply the RHONO (Fig. 5.2), developed in Sect. 5.1, to estimate the state of a three-phase induction motor (3.33). Simulations are performed for the system (3.33), using the following parameters: $R_\mathrm{s} = 14\,\Omega$; $L_\mathrm{s} = 400\,\mathrm{mH}$; $M = 377\,\mathrm{mH}$; $R_\mathrm{r} = 10.1\,\Omega$; $L_\mathrm{r} = 412.8\,\mathrm{mH}$; $n_\mathrm{p} = 2$; $J = 0.01\,\mathrm{Kg\,m^2}$; $T = 0.0001\,\mathrm{s}$. To estimate the state of the system (3.33), we use the RHONO (5.3) with $n = 6$ trained with the EKF (5.7).

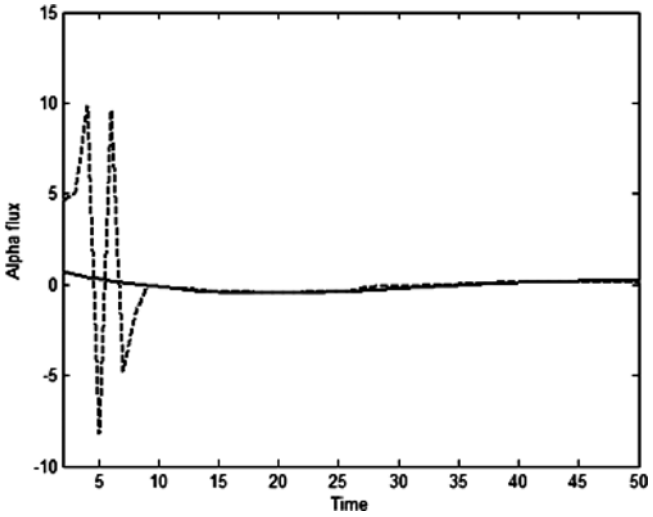

Fig. 5.9. Alpha flux ψ^{α} (*solid line*) and its estimated $\widehat{x}_2$ (*dashed line*)

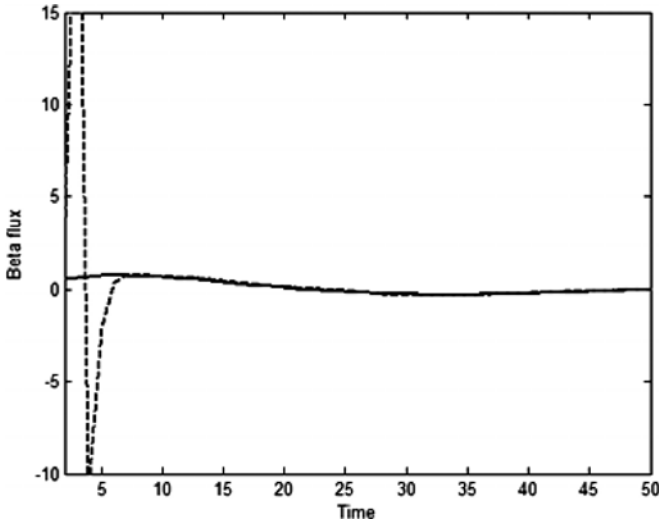

Fig. 5.10. Beta flux ψ^{β} (*solid line*) and its estimated $\widehat{x}_3$ (*dashed line*)

$$
\begin{aligned}
\widehat{x}_1(k+1) &= w_{11}(k)S(\widehat{x}_1(k)) + w_{12}(k)S(\widehat{x}_1)S(\widehat{x}_3(k))\widehat{x}_4(k) \\
&\quad + w_{13}(k)S(\widehat{x}_1)S(\widehat{x}_2(k))\widehat{x}_5(k) + g_1 e(k), \\
\widehat{x}_2(k+1) &= w_{21}(k)S(\widehat{x}_1(k))S(\widehat{x}_3(k)) + w_{22}(k)\widehat{x}_5(k) + g_2 e(k), \\
\widehat{x}_3(k+1) &= w_{31}(k)S(\widehat{x}_1(k))S(\widehat{x}_2(k)) + w_{32}(k)\widehat{x}_4(k) + g_3 e(k), \\
\widehat{x}_4(k+1) &= w_{41}(k)S(\widehat{x}_2(k)) + w_{42}(k)S(\widehat{x}_3(k)) + w_{43}(k)S(\widehat{x}_4(k)) \\
&\quad + w_{44}(k)u^{\alpha}(k) + g_4 e(k),
\end{aligned}
$$

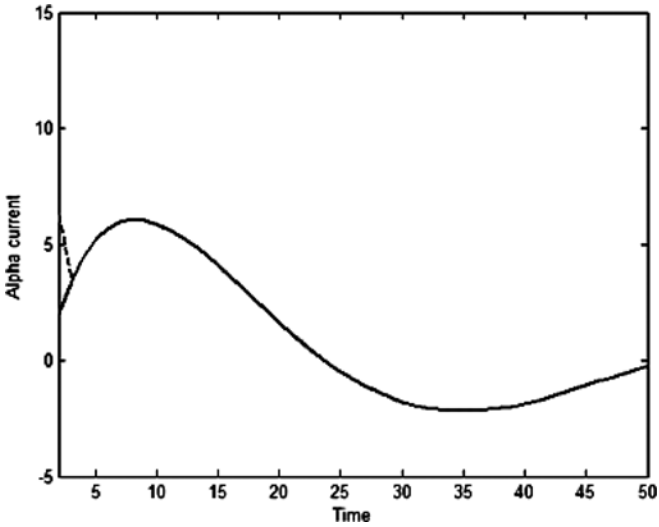

Fig. 5.11. Alpha current i^{α} (*solid line*) and its estimated $\widehat{x}_4$ (*dashed line*)

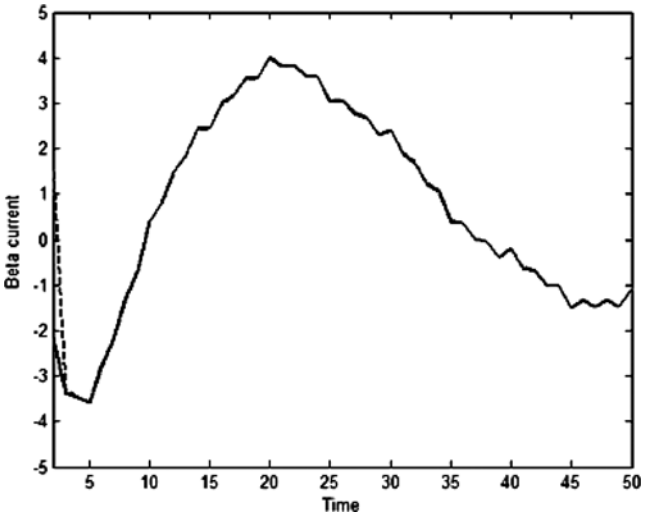

Fig. 5.12. Beta current i^{β} (*solid line*) and its estimated $\widehat{x}_5$ (*dashed line*)

$$\begin{aligned}\widehat{x}_5(k+1) &= w_{51}(k)S(\widehat{x}_2(k)) + w_{52}(k)S(\widehat{x}_3(k)) + w_{53}(k)S(\widehat{x}_5(k)) \\ &\quad + w_{54}(k)u^{\beta}(k) + g_5 e(k), \\ \widehat{x}_6(k+1) &= w_{61}(k)S(\widehat{x}_2(k)) + w_{62}(k)S(\widehat{x}_3(k)) + w_{63}(k)S(\widehat{x}_6(k)) \\ &\quad + g_6 e(k), \end{aligned} \tag{5.17}$$

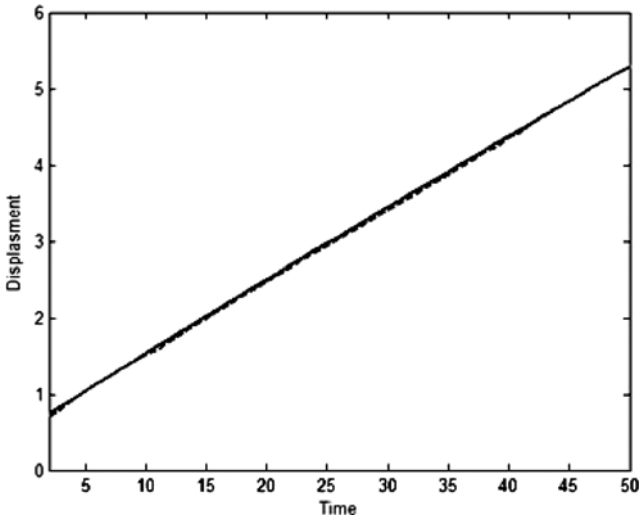

Fig. 5.13. Angular displacement θ (*solid line*) and its estimated $\widehat{x}_6$ (*dashed line*)

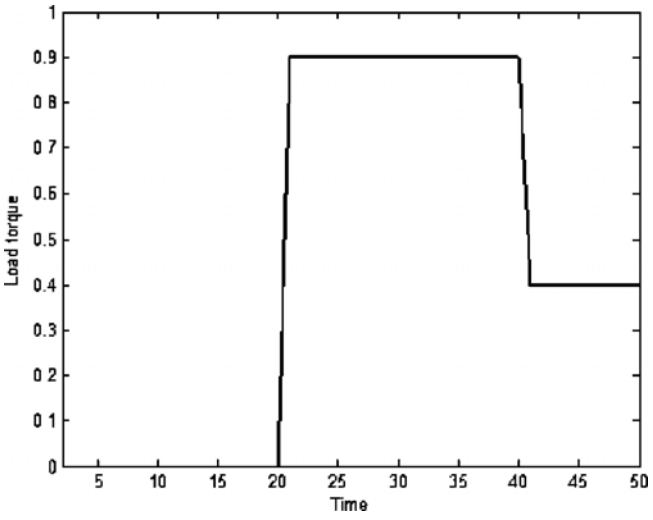

Fig. 5.14. Load torque (T_{L})

where $\widehat{x}_1$ estimates the angular speed ω; $\widehat{x}_2$ and $\widehat{x}_3$ estimates the fluxes ψ^{α} and ψ^{β}, respectively; $\widehat{x}_4$ and $\widehat{x}_5$ estimates the currents i^{α} and i^{β}, respectively; finally $\widehat{x}_6$ estimates the angular displacement θ. The inputs u^{α} and u^{β} are selected as chirp functions.

The training is performed online, using a parallel configuration. All the NN states are initialized in a random way. The associated covariance matrices

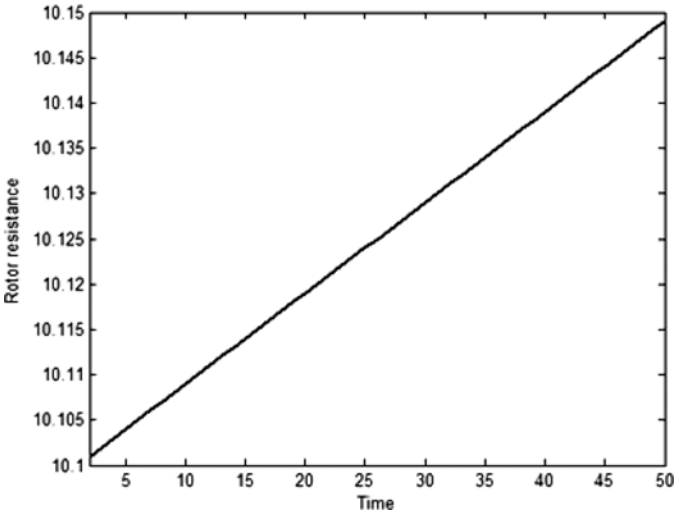

Fig. 5.15. Rotor resistance variation (R_{r})

are initialized as diagonals, and the nonzero elements are $P_i(0) = 10{,}000$; $Q_i(0) = 500$, and $R_i(0) = 10{,}000$, $(i = 1, \cdots, 6)$, respectively. The simulation results are presented in Figs. 5.8–5.13 and they display the time evolution of the estimated states $\widehat{x}_i(k)$, $(i = 1, \cdots, 6)$, respectively. Figures 5.14 and 5.15 display the load torque applied as an external disturbance and the parametric variation introduced in the rotor resistance (R_{r}) as a variation of $1\,\Omega\,\mathrm{s}^{-1}$, respectively.

5.3 Conclusions

In this chapter, a RHONN is used to design a Luenberger-like observer (RHONO) for a class of MIMO discrete-time nonlinear systems. The RHONO proposed is trained with an EKF-based algorithm. The training of the RHONO is performed online in a parallel configuration. The boundness of the output and estimation errors is established on the basis of the Lyapunov approach. Simulation results show the effectiveness of the proposed RHONO. The results presented above seems important due to the need of observers for unknown or partially unknown nonlinear systems in discrete-time.

6

Discrete-Time Output Trajectory Tracking

In this chapter, two schemes for trajectory tracking based on the backstepping and the block control techniques, respectively, are proposed, using an RHONO. This observer is based on a discrete-time recurrent high-order neural network (RHONN), which estimates the state of the unknown plant dynamics. The learning algorithm for the RHONN is based on an EKF. Once the neural network structure is determined, the backstepping and the block control techniques are used to develop the corresponding trajectory tracking controllers. The respective stability analyzes, using the Lyapunov approach, for the neural observer trained with the EKF and the controllers are included. Finally, the applicability of the proposed design is illustrated by an example: output trajectory tracking for an induction motor.

Nonlinear trajectory tracking is an important research subject ([1, 3, 4, 6, 8], and some references cited therein; mostly for continuous-time systems). In the recent literature on adaptive and robust controls, numerous approaches have been proposed for nonlinear trajectory tracking; among them the backstepping and the block control strategies provide well-suited design methodologies [2]. For most nonlinear control designs, it is usually assumed that the whole system state are measurable. In practice, however, it is very difficult to measure all the state variables.

For this reason, nonlinear state estimation remains an important topic for study in the nonlinear systems theory [9]. Recurrent neural-network observers have also been proposed, and they do not require a precise plant model. This technique is therefore attractive and actually has been successfully applied to state estimation [9, 10].

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 59–72 (2008)
www.springerlink.com

6.1 Backstepping Control Using an RHONO

In this section, an RHONO is used to estimate the plant state as in Sect. 5.1, and based on the backstepping technique developed in Sect. 3.1 the trajectory tracking problem is solved. The proposed control scheme is shown in Fig. 6.1. The main result of this chapter is established in the following proposition

Proposition 6.1. *Given a desired output trajectory* y_{d}*, a dynamic system with output* y*, and a neural network with output* $\widehat{y}$*, the following inequality holds* [2]:

$$\|y_{\mathrm{d}} - y\| \leq \|\widehat{y} - y\| + \|y_{\mathrm{d}} - \widehat{y}\|,$$

where $y_{\mathrm{d}} - y$ *is the system output tracking error,* $\widehat{y} - y$ *is the output estimation error, and* $y_{\mathrm{d}} - \widehat{y}$ *is the output tracking error of the nonlinear observer.*

Based on this proposition, it is possible to divide the tracking objective into two parts [2]:

1. Minimization of $\widehat{y} - y$, which can be achieved by the proposed online nonlinear observer algorithm trained with the EKF as shown in Theorem 5.1.
2. Minimization of $y_{\mathrm{d}} - \widehat{y}$. For this, a tracking algorithm is developed on the basis of the nonlinear observer (5.3). This minimization is obtained by designing the control law (3.5), as shown in Theorem 3.1.

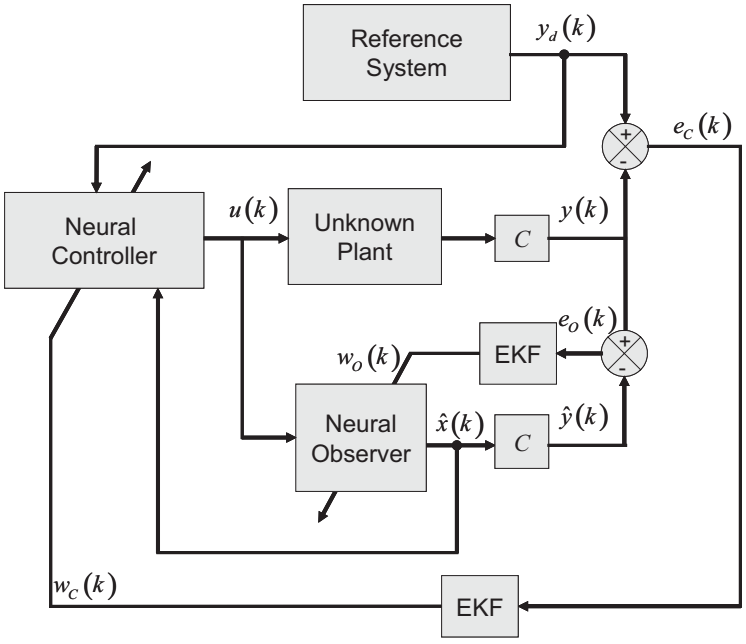

Fig. 6.1. Backstepping control scheme using an RHONO

It is possible to establish Proposition 6.1 due to the separation principle for discrete-time nonlinear systems [7], as stated in Theorem 2.2 and Corollary 2.1.

6.1.1 Application to an Induction Motor

In this section, the control objective is to achieve velocity and flux amplitude tracking for the discrete-time induction motor model (3.33), using the backstepping technique control algorithm developed in Chap. 3 and the RHONO (5.17), as is shown in Fig. 6.1.

Block-Strict-Feedback-Form (BSFF) for an Induction Motor

Let us define the following states:

$$\begin{aligned} x^1(k) &= \begin{bmatrix} \widehat{x}_1(k) \\ \Psi(k) \end{bmatrix}; \quad x^2(k) = \begin{bmatrix} \widehat{x}_4(k) \\ \widehat{x}_5(k) \end{bmatrix}, \\ u(k) &= \begin{bmatrix} u^{\alpha}(k) \\ u^{\beta}(k) \end{bmatrix}; \quad y_{\mathrm{d}}(k) = \begin{bmatrix} \omega_{\mathrm{d}}(k) \\ \Psi_{\mathrm{d}}(k) \end{bmatrix}, \\ y(k) &= x^1(k), \end{aligned} \tag{6.1}$$

where $\Psi(k) = \widehat{x}_2^2(k) + \widehat{x}_3^2(k)$ is the rotor flux magnitude, $\omega_{\mathrm{d}}(k)$ and $\Psi_{\mathrm{d}}(k)$ are the reference signals. The control objective is to force the output $y(k)$ to track the reference $y_{\mathrm{d}}(k)$. Using (6.1), the system (3.33) can be represented in the BSFF consisting of two blocks

$$\begin{aligned} x^1(k+1) &= f^1(x^1(k)) + g^1(x^1(k))x^2(k) + d^1(k), \\ x^2(k+1) &= f^2(\overline{x}^2(k)) + g^2(\overline{x}^2(k))u(k), \end{aligned}$$

where $f^1(x^1(k))$, $g^1(x_1(k))$, $f_2(\overline{x}_2(k))$, and $g_2(\overline{x}_2(k))$ are assumed to be unknown and $d_1(k)$ is an unknown bounded disturbance; in this case, this disturbance is the load torque. Now we use the HONN to approximate the desired virtual controls and the ideal practical controls described as

$$\begin{aligned} \alpha^{1*}(k) \triangleq x^2(k) &= \varphi^1(x^1(k), y_{\mathrm{d}}(k+2)), \\ u^*(k) &= \varphi^2(x^1(k), x^2(k), \alpha^{1*}(k)), \\ y(k) &= x^1(k). \end{aligned}$$

The HONN proposed for this application is as follows:

$$\begin{aligned} \alpha^1(k) &= {w^1}^{\top} z^1(\varrho^1(k)), \\ u(k) &= {w^2}^{\top} z^2(\varrho^2(k)), \end{aligned}$$

with

$$\varrho^1(k) = [x^1(k), y_\mathrm{d}(k+2)]^\top,$$
$$\varrho^2(k) = [x^1(k), x^2(k), \alpha^1(k)]^\top.$$

The weights are updated using the EKF:

$$w^i(k+1) = w^i(k) + \eta^i K^i(k) e^i(k), \qquad i = 1,\ 2,$$
$$K^i(k) = P^i(k) H^i(k) \left[R^i(k) + H^{i^\top}(k) P^i(k) H^i(k) \right]^{-1},$$
$$P^i(k+1) = P^i(k) - K^i(k) H^{i^\top}(k) P^i(k) + Q^i(k),$$

with

$$e^1(k) = y_\mathrm{d}(k) - y(k),$$
$$e^2(k) = x^2(k) - \alpha^1(k),$$

The training is performed online, using a parallel configuration. All the NN states are initialized in a random way. The associated covariances matrices are initialized as diagonals, and the nonzero elements are $P_1(0) = P_2(0) = 10{,}000$; $Q_1(0) = Q_2(0) = 5{,}000$; and $R_1(0) = R_2(0) = 10{,}000$, respectively.

Simulation Results

The simulations are performed for the system (3.33) using the following parameters: $R_\mathrm{s} = 14\,\Omega$; $L_\mathrm{s} = 400\,\mathrm{mH}$; $M = 377\,\mathrm{mH}$; $R_\mathrm{r} = 10.1\,\Omega$; $L_\mathrm{r} = 412.8\,\mathrm{mH}$; $n_\mathrm{p} = 2$; $J = 0.01\,\mathrm{Kg\,m^2}$; $T = 0.0001\,\mathrm{s}$. To estimate the state of system (3.33), we use the RHONO (5.3) with $n = 5$ trained with the EKF (5.7).

The tracking results are presented in Figs. 6.2 and 6.3. There the tracking performance can be verified for the two plant outputs. Figure 6.4 displays the load torque applied as an external disturbance. Figure 6.5 portrays a parametric variation introduced in the rotor resistance (R_r) as an increment. Figure 6.6 shows the weights evolution. Figures 6.7 and 6.8 portray the fluxes and their estimates.

6.2 Block Control Using an RHONO

In this section, an RHONO is used to estimate the plant state as in Sect. 5.1, and based on the block control technique developed in Sect. 4.2 the trajectory tracking problem is solved on the same basis of Proposition 6.1, dividing the tracking objective into two parts, as in Sect. 6.1 [2]. The proposed control scheme is shown in Fig. 6.9.

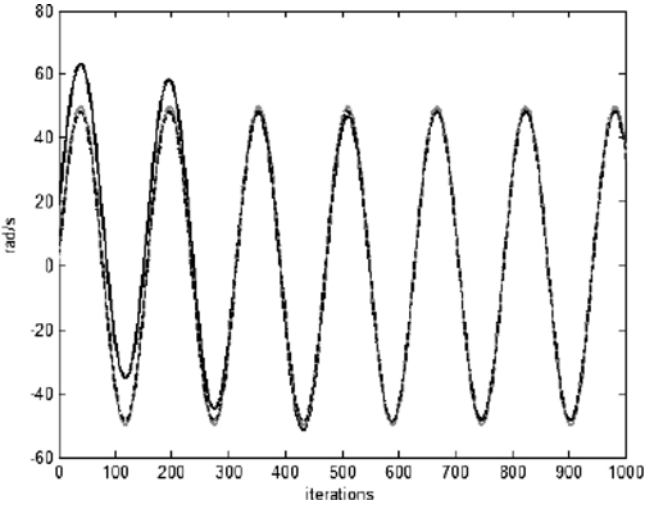

Fig. 6.2. Tracking performance $\omega(k)$ (*solid line*), $\widehat{x}_1(k)$ (*dash-dot line*), and $\omega_{\mathrm{r}}(k)$ (*dashed line*)

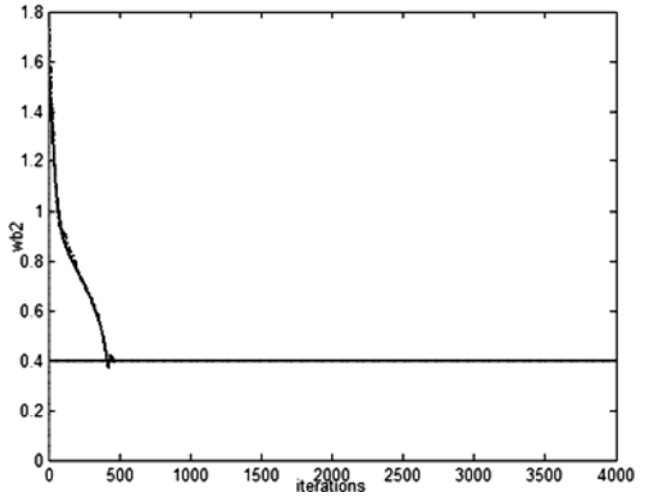

Fig. 6.3. Tracking performance $\Psi(k)$ (*solid line*), $\widehat{x}_2^2(k)+\widehat{x}_3^2(k)$ (*dash-dot line*), and $\Psi_{\mathrm{r}}(k)$ (*dashed line*)

6.2.1 Application to an Induction Motor

To this end we use the RHONO developed for the discrete-time induction motor model, developed in Sect. 5.2, which is described as

$$\begin{aligned}\widehat{x}_1(k+1) = {} & w_{11}(k)S(\widehat{x}_1(k)) + w_{12}(k)S(\widehat{x}_1)S(\widehat{x}_3(k))\widehat{x}_4(k) \\ & + w_{13}(k)S(\widehat{x}_1)S(\widehat{x}_2(k))\widehat{x}_5(k) + g_1 e(k),\end{aligned}$$

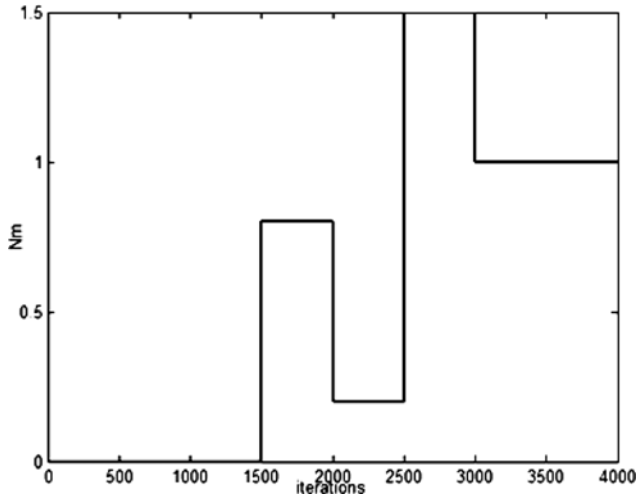

Fig. 6.4. Load torque $T_{\mathrm{L}}(k)$

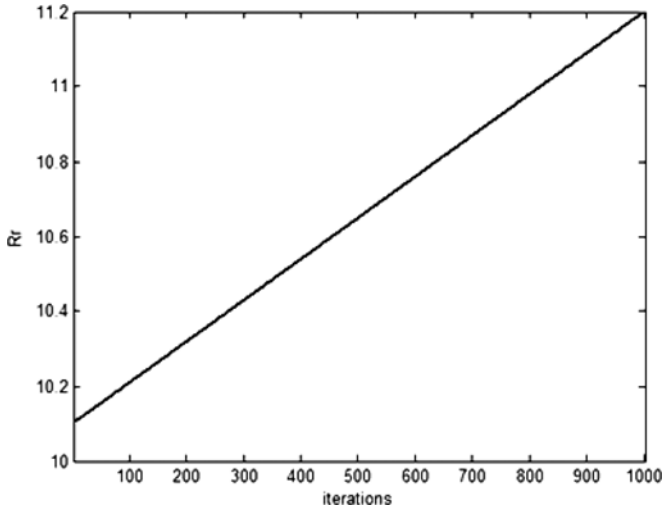

Fig. 6.5. Rotor resistance variation (R_{r})

$$
\begin{aligned}
\widehat{x}_2(k+1) &= w_{21}(k)S(\widehat{x}_1(k))S(\widehat{x}_3(k)) + w_{22}(k)\widehat{x}_5(k) + g_2e(k),\\
\widehat{x}_3(k+1) &= w_{31}(k)S(\widehat{x}_1(k))S(\widehat{x}_2(k)) + w_{32}(k)\widehat{x}_4(k) + g_3e(k),\\
\widehat{x}_4(k+1) &= w_{41}(k)S(\widehat{x}_2(k)) + w_{42}(k)S(\widehat{x}_3(k)) + w_{43}(k)S(\widehat{x}_4(k))\\
&\quad + w_{44}(k)u^{\alpha}(k) + g_4e(k),\\
\widehat{x}_5(k+1) &= w_{51}(k)S(\widehat{x}_2(k)) + w_{52}(k)S(\widehat{x}_3(k)) + w_{53}(k)S(\widehat{x}_5(k))\\
&\quad + w_{54}(k)u^{\beta}(k) + g_5e(k),
\end{aligned}
\tag{6.2}
$$

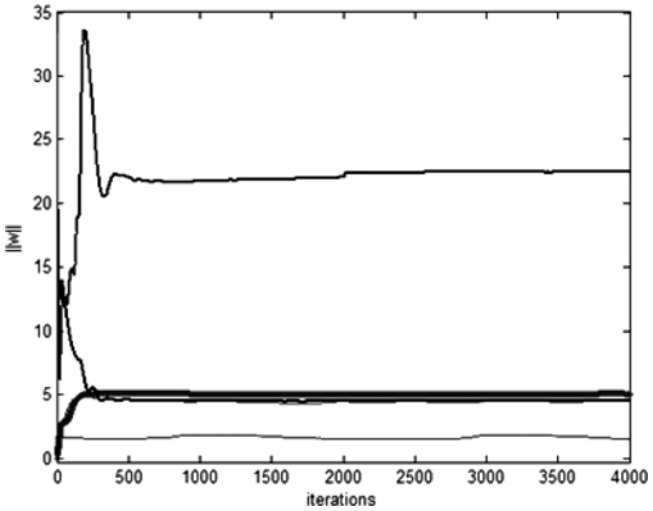

Fig. 6.6. Weights evolution

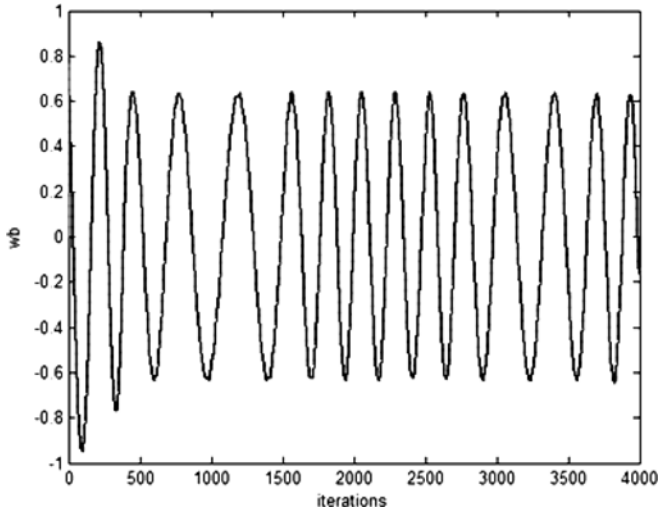

Fig. 6.7. Time evolution of $\psi^{\alpha}(k)$ (*solid line*) and its estimated $\widehat{x}_2(k)$ (*dashed line*)

where $\widehat{x}_1$ estimates the angular speed ω; $\widehat{x}_2$ and $\widehat{x}_3$ estimates the fluxes ψ^{α} and ψ^{β}, respectively; $\widehat{x}_4$ and $\widehat{x}_5$ estimates the currents i^{α} and i^{β}, respectively. The training is performed online, using a parallel configuration. All the NN states are initialized in a random way as well as the weights vectors. It is important to remark that the initial conditions of the plant are completely different from the initial conditions for the NN.

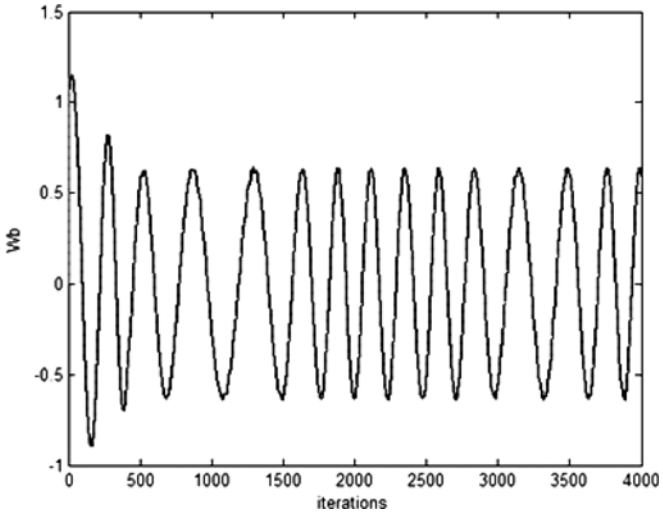

Fig. 6.8. Time evolution of $\psi^{\beta}(k)$ (*solid line*) and its estimated $\widehat{x}_3(k)$ (*dashed line*)

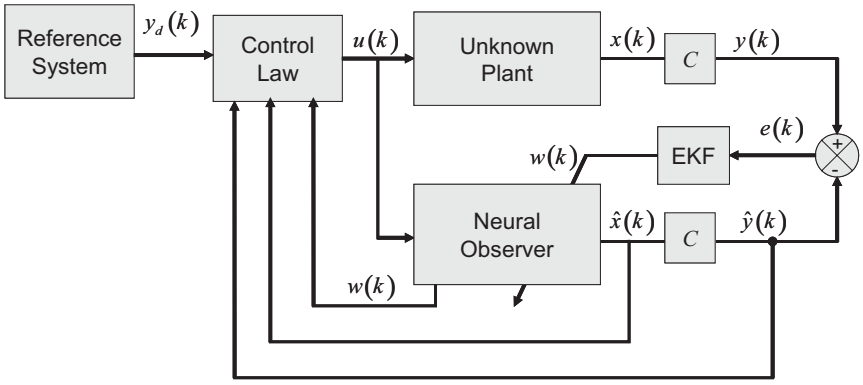

Fig. 6.9. Block control scheme using an RHONO

Comment 6.1. It is important to remark that as in Chap. 4, to apply the Block control and the sliding modes techniques it is necessary to use the modified EKF (4.5), to avoid the zero-crossing for $w_{44}(k)$ and $w_{54}(k)$. The proof for the RHONN trained with the modified EKF is similar to the proof of Theorem 4.1.

Neural Block Controller Design

The control objective is to achieve velocity and flux amplitude tracking for the discrete-time induction motor model (3.33), using the discrete-time block control and sliding mode techniques control algorithm developed in Chap. 4. Let us define the following states as

$$x^1(k) = \begin{bmatrix} \widehat{x}_1(k) - \omega_{\rm r}(k) \\ \Psi(k) - \Psi_{\rm r}(k) \end{bmatrix}, \quad x^2(k) = \begin{bmatrix} \widehat{x}_4(k) \\ \widehat{x}_5(k) \end{bmatrix}, \tag{6.3}$$

where $\Psi(k) = \widehat{x}_2^2(k) + \widehat{x}_3^2(k)$ is the rotor flux identify magnitude, $\Psi_{\rm r}(k)$ and $\omega_{\rm r}(k)$ are reference signals. Then

$$\begin{aligned}
\Psi(k+1) = {} & w_{21}^2(k) S^2(\widehat{x}_1(k)) S^2(\widehat{x}_3(k)) + w_{22}^2(k) \widehat{x}_5^2(k) + w_{32}^2(k) \widehat{x}_4^2(k) \\
& + w_{31}^2(k) S^2(\widehat{x}_1(k)) S^2(\widehat{x}_2(k)) \\
& + 2 w_{21}(k) S(\widehat{x}_1(k)) S(\widehat{x}_3(k)) w_{22}(k) \widehat{x}_5(k) \\
& + 2 w_{31}(k) S(\widehat{x}_1(k)) S(\widehat{x}_2(k)) w_{32}(k) \widehat{x}_4(k) \\
& + 2 w_{22}(k) \widehat{x}_5(k) g_2 e(k) + (g_2 e(k))^2 \\
& + 2 w_{21}(k) S(\widehat{x}_1(k)) S(\widehat{x}_3(k)) g_2 e(k) + (g_3 e(k))^2 \\
& + 2 w_{32}(k) \widehat{x}_4(k) g_3 e(k) + 2 w_{31}(k) S(\widehat{x}_1(k)) S(\widehat{x}_2(k)) g_3 e(k).
\end{aligned}$$

Using (6.3), (6.2) can be represented in the block control form consisting of two blocks

$$\begin{aligned}
x^1(k+1) &= f_1(x^1(k)) + B_1(x^1(k)) x^2(k), \\
x^2(k+1) &= f_2(x^1(k), x^2(k)) + B_2(k) u(k),
\end{aligned} \tag{6.4}$$

with $u(k) = \left[u^\alpha(k) \; u^\beta(k) \right]^\top$ and

$$\begin{aligned}
f_1(x^1(k)) &= \begin{bmatrix} w_{11}(k) S(x_1(k)) + g_1 e(k) - \omega_{\rm r}(k+1) \\ f_{11}(k) \end{bmatrix}, \\
f_{11}(k) &= w_{21}^2(k) S^2(\widehat{x}_1(k)) S^2(\widehat{x}_3(k)) + w_{31}^2(k) S^2(\widehat{x}_1(k)) S^2(\widehat{x}_2(k)) \\
&\quad + 2 w_{22}(k) \widehat{x}_5(k) g_2 e(k) + (g_2 e(k))^2 \\
&\quad + 2 w_{21}(k) S(\widehat{x}_1(k)) S(\widehat{x}_3(k)) g_2 e(k) \\
&\quad + (g_3 e(k))^2 + 2 w_{32}(k) \widehat{x}_4(k) g_3 e(k) \\
&\quad + (g_3 e(k))^2 + 2 w_{32}(k) \widehat{x}_4(k) g_3 e(k) \\
&\quad + w^2 I_m^2(k) - \Psi_r(k+1), \\
I_m(k) &= \sqrt{w_{22}^2(k) \widehat{x}_4^2(k) + w_{32}^2(k) \widehat{x}_5^2(k)}, \\
B_1(x^1(k)) &= \begin{bmatrix} b_{11}(k) \; b_{12}(k) \\ b_{21}(k) \; b_{22}(k) \end{bmatrix}, \\
b_{11}(k) &= w_{12}(k) S(\widehat{x}_1(k)) S(\widehat{x}_3(k)), \\
b_{12}(k) &= w_{13}(k) S(\widehat{x}_1(k)) S(\widehat{x}_2(k)), \\
b_{21}(k) &= 2 w_{31}(k) w_{32}(k) S(\widehat{x}_1(k)) S(\widehat{x}_2(k)), \\
b_{22}(k) &= 2 w_{21}(k) w_{22}(k) S(\widehat{x}_1(k)) S(\widehat{x}_2(k)), \\
f_2(x^2(k)) &= \begin{bmatrix} f_{21}(k) \\ f_{22}(k) \end{bmatrix}, B_2(k) = \begin{bmatrix} w_{44}(k) & 0 \\ 0 & w_{54}(k) \end{bmatrix}, \\
f_{21}(k) &= w_{41}(k) S(\widehat{x}_2(k)) + w_{42}(k) S(\widehat{x}_3(k)) + w_{43}(k) S(\widehat{x}_4(k)),
\end{aligned}$$

$$f_{22}(k) = w_{51}(k)S(\widehat{x}_2(k)) + w_{52}(k)S(\widehat{x}_3(k)) + w_{53}(k)S(\widehat{x}_5(k)).$$

Applying the block control technique, we define the following vector $\mathbf{z}_1(k) = x^1(k)$. Then

$$\mathbf{z}_1(k+1) = f_1(x^1(k)) + B_1(x^1(k))x^2(k) = \mathbf{K}\mathbf{z}_1(k), \tag{6.5}$$

where $\mathbf{K} = \text{diag}\{\mathbf{k}_1, \mathbf{k}_2\}$, with $|\mathbf{k}_i| < 1$ $(i = 1, 2)$; then the desired value $x^{2\text{d}}(k)$ of $x^2(k)$ is calculated from (6.5) as

$$x^{2\text{d}}(k) = B_1^{-1}(x^1(k))[-f_1(x^1(k)) + \mathbf{K}\mathbf{z}_1(k)].$$

It is desired that $x^2(k) = x^{2\text{d}}(k)$. Hence, second new error vector is defined as

$$\mathbf{z}_2(k) = x^2(k) - x^{2\text{d}}(k).$$

Then

$$\mathbf{z}_2(k+1) = f_3(x^1(k)) + B_2(k)u(k),$$

with

$$f_3(x^1(k)) = f_2(x^2(k)) - B_1^{-1}(x^1(k+1))[-f_1(x^1(k+1)) + \mathbf{K}\mathbf{z}_1(k+1)].$$

Let us select the manifold for the sliding mode as $S_{\text{D}}(k) = \mathbf{z}_2(k)$. To design a control law, a discrete-time sliding mode version is implemented as

$$u(k) = \begin{cases} u_{\text{eq}}(k) & \text{if } \|u_{\text{eq}}(k)\| \le u_0, \\ u_0 \frac{u_{\text{eq}}(k)}{\|u_{\text{eq}}(k)\|} & \text{if } \|u_{\text{eq}}(k)\| > u_0, \end{cases}$$

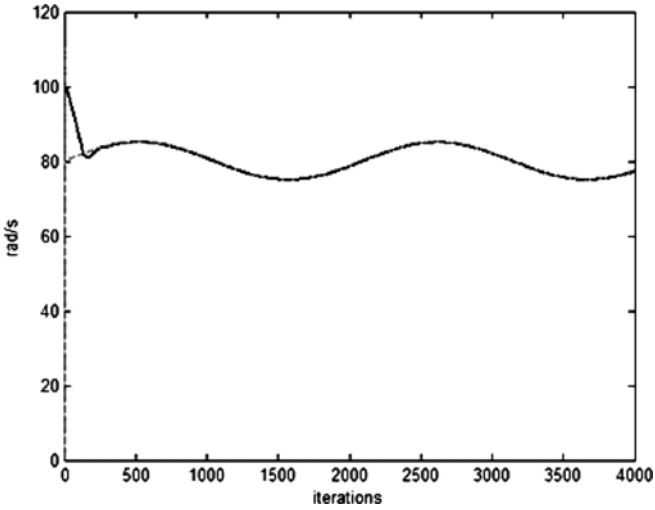

Fig. 6.10. Tracking performance $\omega(k)$ (*solid line*), $\widehat{x}_1(k)$ (*dash-dot line*) and $\omega_r(k)$ (*dashed line*)

where $u_{\mathrm{eq}}(k) = -B_2^{-1}(k) f_3(x^1(k))$ is calculated from $S_{\mathrm{D}}(k) = 0$ and u_0 is the control resources that bound the control. Because of the time varying of RHONO weights, we need to guarantee that $B_1(\bullet)$ and $B_2(\bullet)$ are not singular; then it is necessary to avoid the zero-crossing of the weights $w_{13}(k)$, $w_{22}(k)$, $w_{32}(k)$, $w_{44}(k)$, and $w_{54}(k)$, which are the so-called controllability weights [2]. It is important to remark that in this application only the weights $w_{44}(k)$ and $w_{54}(k)$ tend to cross zero.

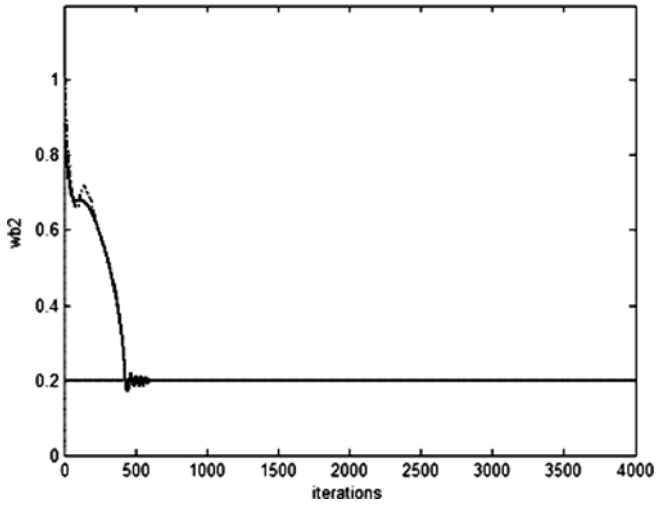

Fig. 6.11. Tracking performance $\Psi(k)$ (*solid line*), $\widehat{x}_2^2(k) + \widehat{x}_3^2(k)$ (*dash-dot line*), and $\Psi_r(k)$ (*dashed line*)

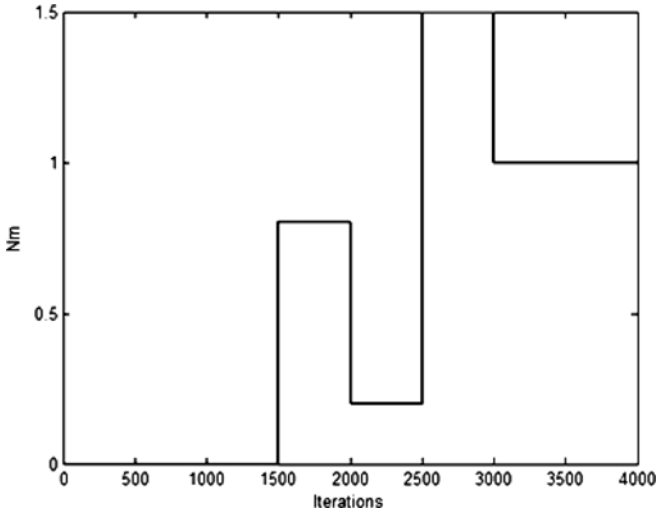

Fig. 6.12. Load torque $T_{\mathrm{L}}(k)$

Simulation Results

Simulations are performed for the system (3.33), using the following parameters: $R_\mathrm{s} = 14\,\Omega$; $L_\mathrm{s} = 400\,\mathrm{mH}$; $M = 377\,\mathrm{mH}$; $R_\mathrm{r} = 10.1\,\Omega$; $L_\mathrm{r} = 412.8\,\mathrm{mH}$; $n_\mathrm{p} = 2$; $J = 0.01\,\mathrm{Kg\,m^2}$; $T = 0.001\,\mathrm{s}$. To estimate the state of system (3.33) we use the RHONO (5.3) with $n = 5$ trained with the EKF (5.7).

The tracking results are presented in Figs. 6.10 and 6.11. There the tracking and state estimation performance can be verified for the two plant outputs.

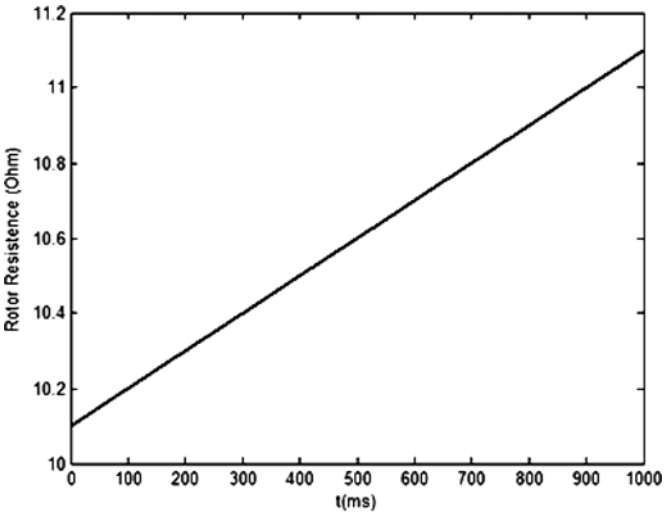

Fig. 6.13. Rotor resistance variation (R_r)

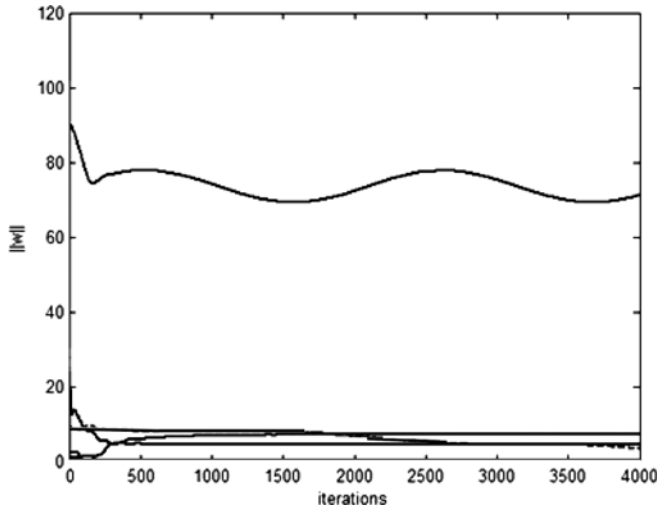

Fig. 6.14. Weights evolution

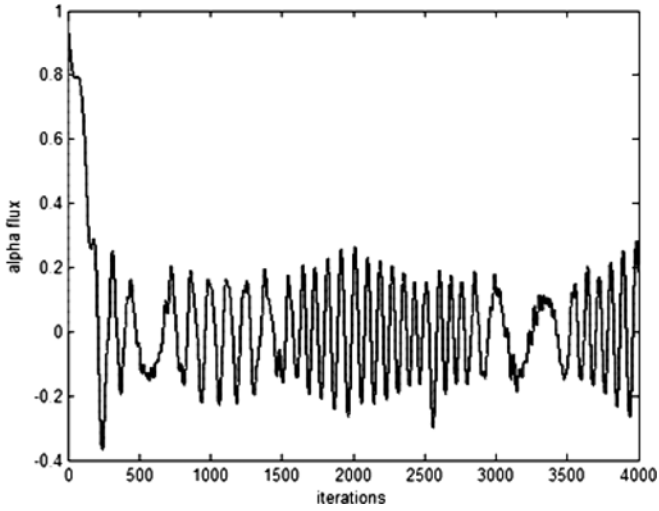

Fig. 6.15. Time evolution of $\psi^{\alpha}(k)$ (*solid line*) and its estimated $\widehat{x}_2(k)$ (*dashed line*)

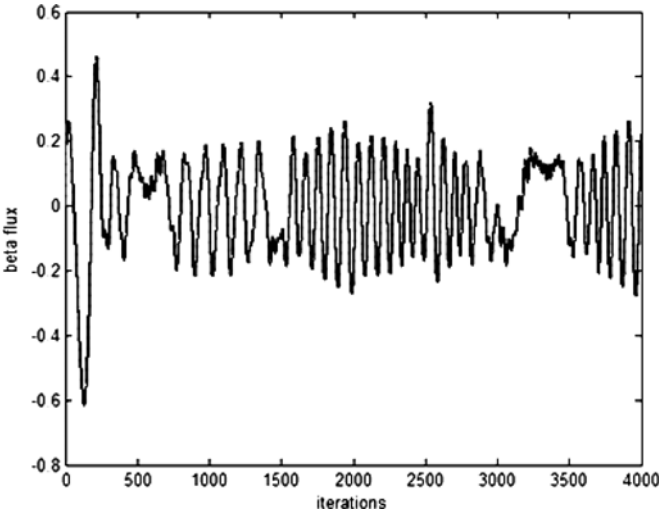

Fig. 6.16. Time evolution of $\psi^{\beta}(k)$ (*solid line*) and its estimated $\widehat{x}_3(k)$ (*dashed line*)

Figure 6.12 displays the load torque applied as an external disturbance. Figure 6.13 presents the parametric variation introduced in the rotor resistance (R_{r}) as a variation of $1\,\Omega\,\mathrm{s}^{-1}$. Figure 6.14 shows the weights evolution. Figures 6.15 and 6.16 portray the fluxes and their estimates.

6.3 Conclusions

In this chapter, the discrete-time output trajectory tracking is solved via the design of two neural controllers based on the backstepping and the block control techniques, respectively. First, a nonlinear observer is designed based on a RHONN trained with a modified EKF-based algorithm, where the training of the nonlinear observer is performed online in a parallel configuration. Then, based on the RHONO, the backstepping and the block control techniques are designed, respectively. Simulation results for an induction motor are included to illustrate the applicability of the proposed control schemes.

7

Real Time Implementation

In this chapter real time implementation is presented in order to validate the theoretical results discussed in previous chapters. The results presented in this chapter include the Neural Network Identification scheme presented in Chap. 4, the RHONO presented in Chap. 5, the Neural Backstepping Approach analyzed in Chap. 3, the Neural Bock Control Technique discussed in Chap. 4 and the modifications of the last two controllers treated in Chap. 6 to include the RHONO. All these applications was performed using a three phase induction motor.

The experiments are performed using a benchmark, which includes a PC for supervising, a PWM unit for the power stage, a dSPACE DS1104 board for data acquisition and control of the system (dSPACE is a trademark of dSPACE GmbH), and a three phase induction motor as the plant has to be controlled, with the following characteristics: 220 V, 60 Hz, 0.19 kW, 1,660 rpm, 1.3 A [1]. Series of photographs and figures of the benchmark are included. Figure 7.1 presents a schematic representation of the benchmark used in these experiments. Figure 7.2 displays the encoder coupled with an induction motor, Fig. 7.3 presents a view of the PC and the DS1104 board, and Fig. 7.4 shows the PWM driver. The DS1104 board allows to download applications directly from Simulink (Matlab and Simulink are trademarks of the MathWorks Inc.) as is shown in Fig. 7.5. In Fig. 7.6, a Desktop interface for the DS1104 board is included in order to clarify the visualization of the experiments. The experiments performed in the benchmark includes the Neural Network Identification scheme presented in Chap. 4, the RHONO presented in Chap. 5, the Neural Backstepping Approach analyzed in Chap. 3, the Neural Bock Control Technique discussed in Chap. 4, and the modifications of the last two controllers treated in Chap. 6 to include the RHONO. Finally, the mentioned experiments are tested with a constant load torque applied with a DC generator coupled to an induction motor as shown in Figs. 7.7 and 7.8.

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 73–94 (2008)
www.springerlink.com

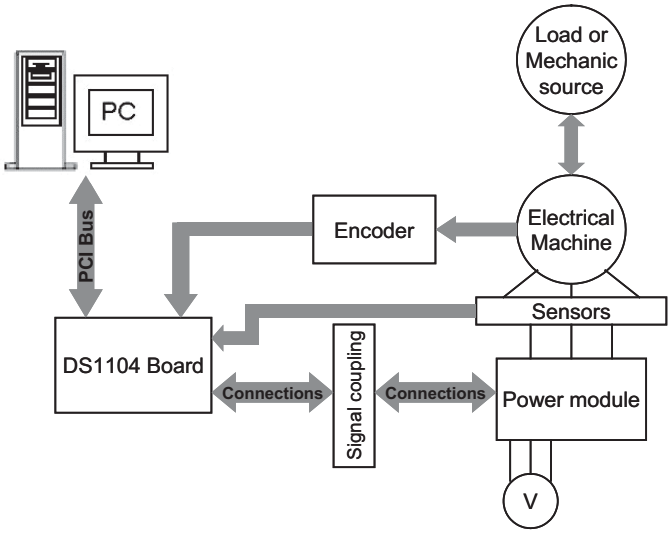

Fig. 7.1. Schematic representation of the control prototype

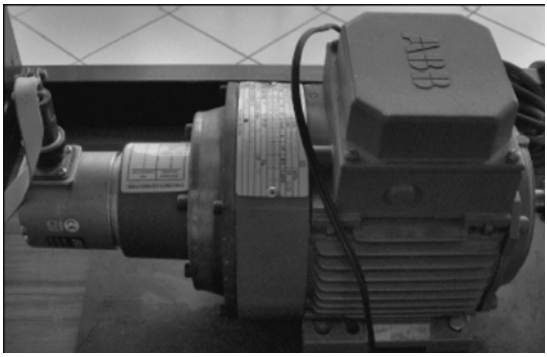

Fig. 7.2. Encoder coupled with the induction motor

7.1 Neural Identification

In this section the neural network identification scheme proposed in Chap. 4 for the discrete-time induction motor model is applied in real time to the benchmark described above. During the identification process the plant and the NN operates in open-loop. Both of them (plant and NN) have the same input vector $\left[\, u_\alpha \; u_\beta \,\right]^\top$; u_α and u_β are chirp functions with 200 V of amplitude and incremental frequencies from 0 to 150 Hz and 0 to 200 Hz, respectively. The implementation is performed with a sampling time of 0.0005 s. The results of the real-time implementation are presented as follows: Fig. 7.9 shows the identification of rotor angular displacement; Fig. 7.10 displays the identification

Fig. 7.3. View of the PC and the DS1104 board

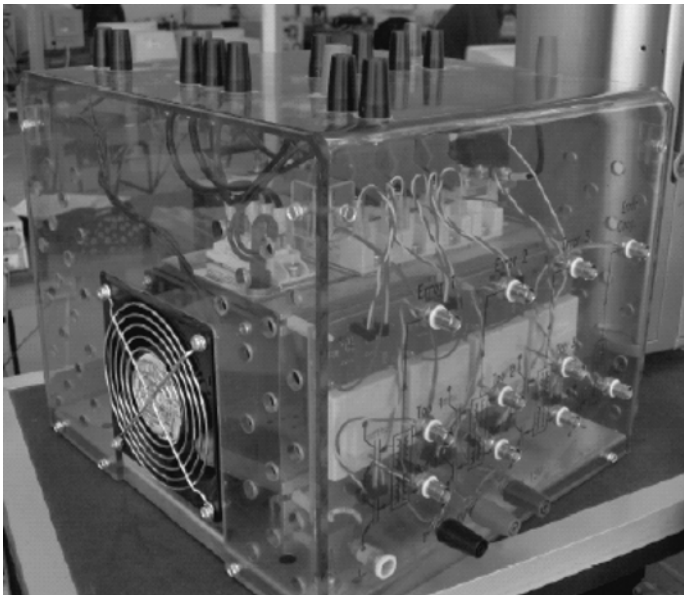

Fig. 7.4. PWM driver

performance for the speed rotor; Figs. 7.11 and 7.12 present the identification performance for the fluxes in phase α and β, respectively. Figures 7.13 and 7.14 portray the identification performance for currents in phase α and β, respectively. Finally, the input signals are presented in Fig. 7.15.

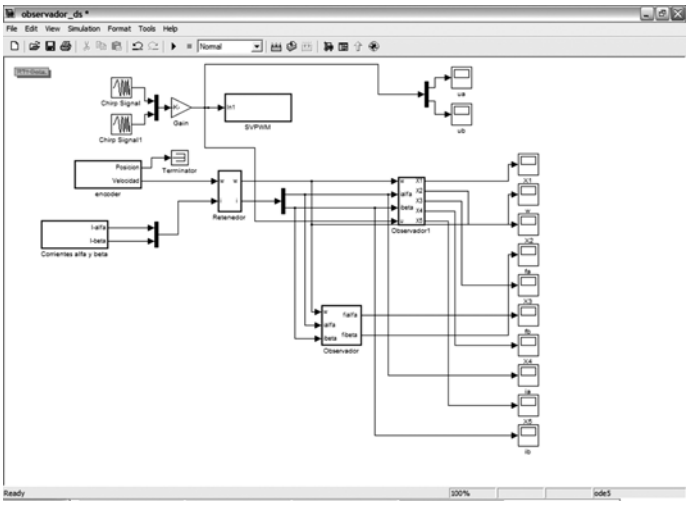

Fig. 7.5. Simulink program to be downloaded to the DS1104 board directly

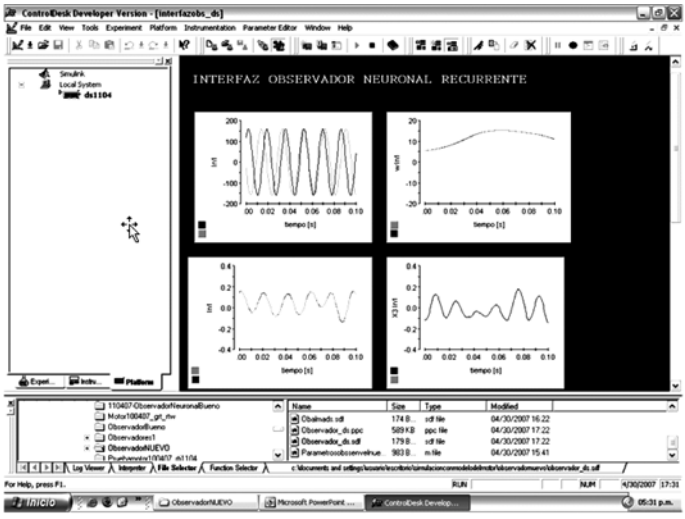

Fig. 7.6. Desktop interface for the DS1104 board

7.2 Neural State Estimation

This section presents the neural network observer (RHONO) scheme proposed in Chap. 5 for the discrete-time induction motor model as applied in real time to the benchmark described above. During the estimation process, the plant and the NN operates in open-loop. Both of them (plant and NN) have the same input vector $\left[u_\alpha \; u_\beta\right]^\top$; u_α and u_β are chirp functions with

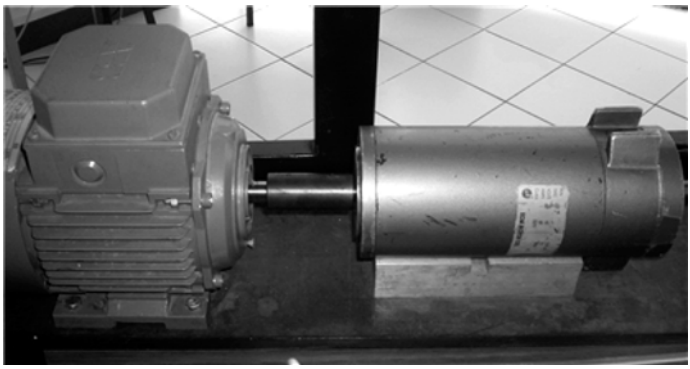

Fig. 7.7. DC generator coupled to the induction motor as a constant load torque

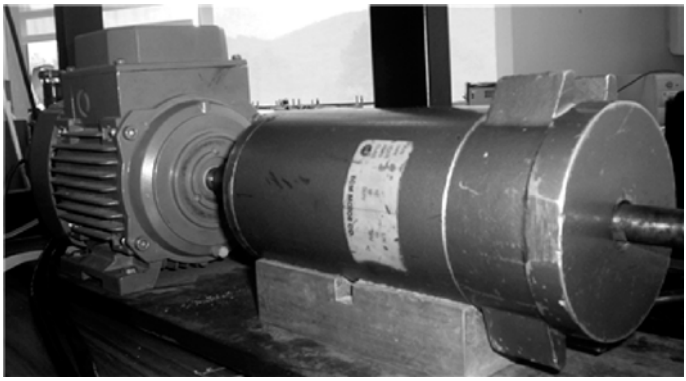

Fig. 7.8. Complete view of the DC generator coupled to the induction motor as a constant load torque

200 V of amplitude and incremental frequencies from 0 to 150 Hz and 0 to 200 Hz, respectively. The implementation is performed with a sampling time of 0.0005 s. The results of the real-time implementation are presented as follows: Fig. 7.16 displays the estimation performance for the speed rotor; Figs. 7.17 and 7.18 present the estimation performance for the fluxes in phase α and β, respectively. Figures 7.19 and 7.20 portray the estimation performance for currents in phase α and β, respectively. Finally, the input signals are presented in Fig. 7.21.

7.3 Neural Backstepping Control

This section describes the real time results of the control law designed in Chap. 3, based on the backstepping technique approximated by a HONN, for the discrete-time induction motor model with a sampling time of 0.0005 s as

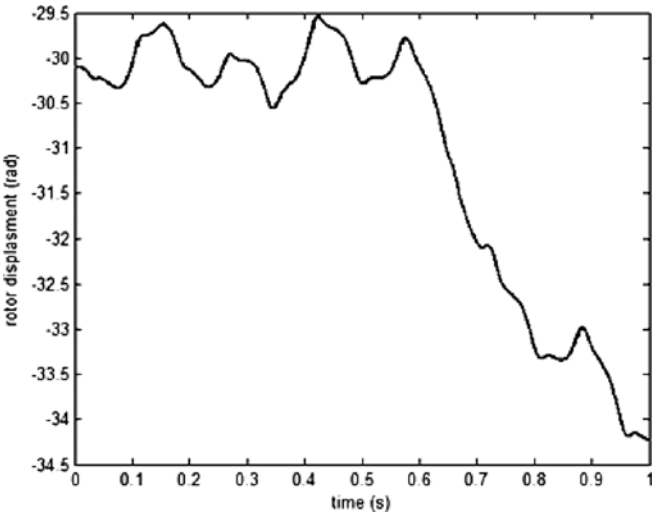

Fig. 7.9. Real time rotor displacement identification (plant signal in *solid line* and neural signal in *dashed line*)

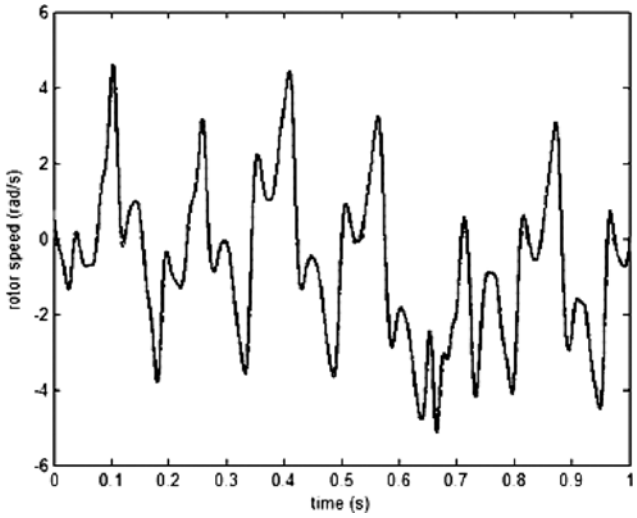

Fig. 7.10. Real time rotor speed identification (plant signal in *solid line* and neural signal in *dashed line*)

follows: The tracking results for the rotor speed and for the flux magnitude are presented in Figs. 7.22 and 7.23 for the induction motor working without load, respectively; Fig. 7.24 shows the control law in phases α and β; Fig. 7.25 presents the tracking result for the rotor speed under the presence of a load. Finally, Fig. 7.26 displays the tracking result under the presence of an external disturbance.

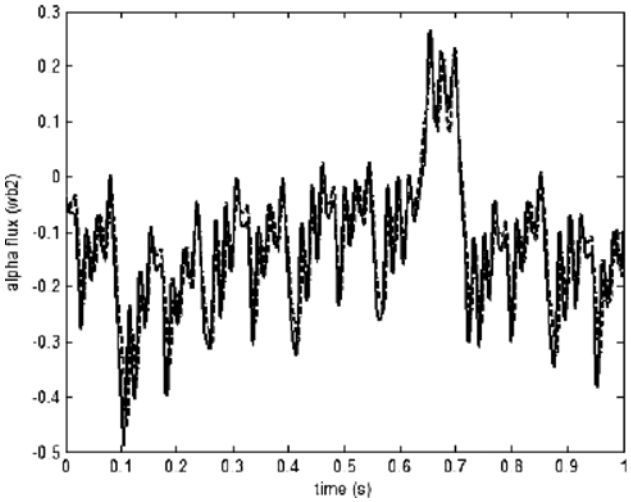

Fig. 7.11. Real time alpha flux identification (plant signal in *solid line* and neural signal in *dashed line*)

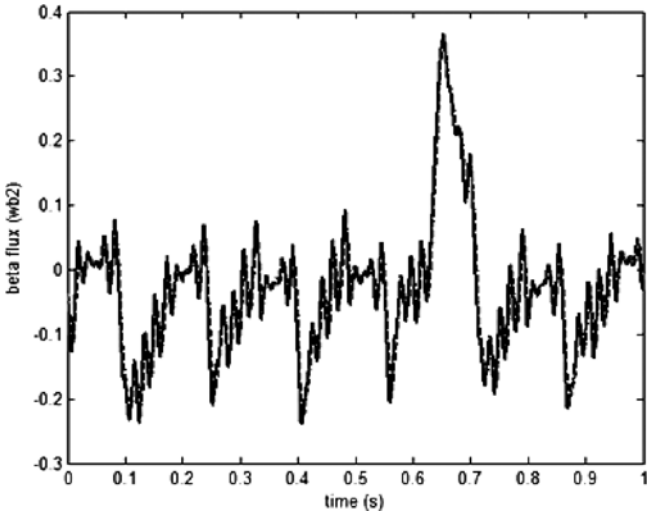

Fig. 7.12. Real time rotor beta flux identification (plant signal in *solid line* and neural signal in *dashed line*)

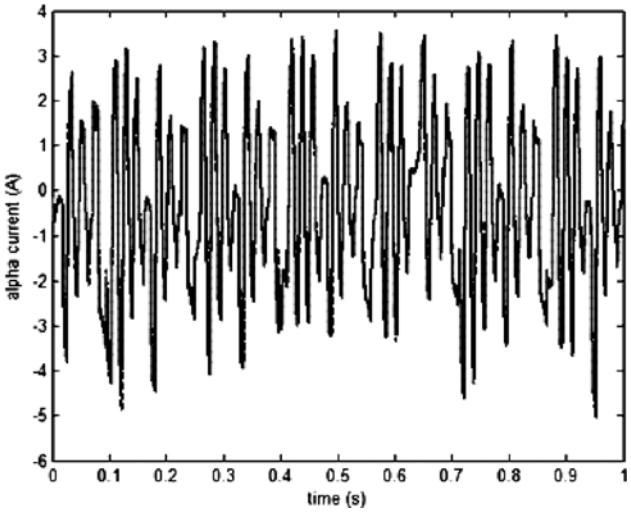

Fig. 7.13. Real time rotor alpha current identification (plant signal in *solid line* and neural signal in *dashed line*)

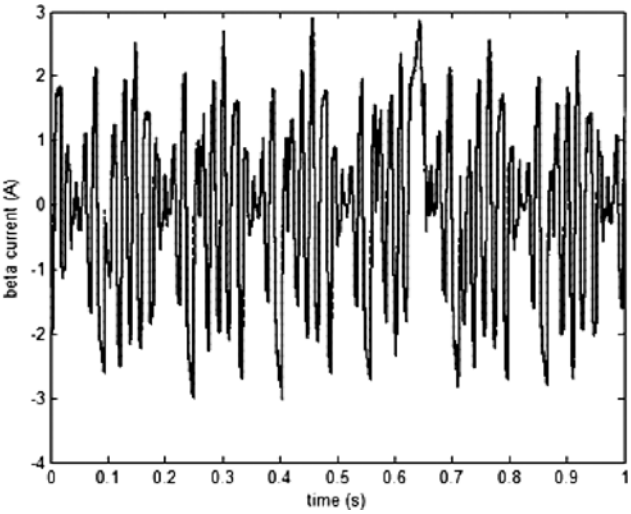

Fig. 7.14. Real time beta current speed identification (plant signal in *solid line* and neural signal in *dashed line*)

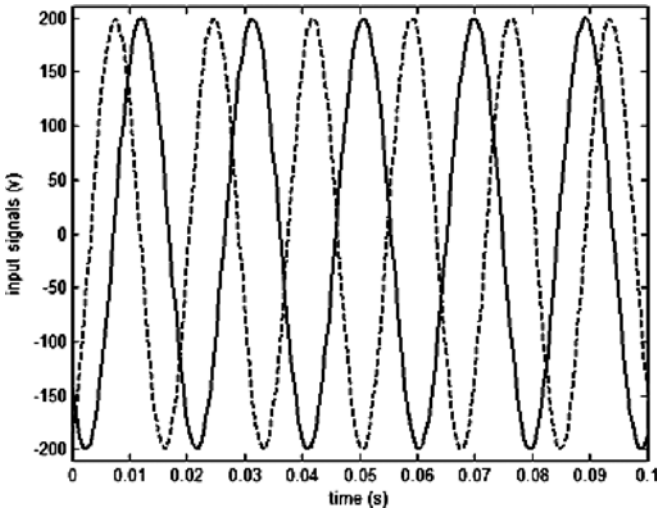

Fig. 7.15. Input signals applied during the identification process ($u^{\alpha}(k)$ in *solid line* and $u^{\beta}(k)$ in *dashed line*)

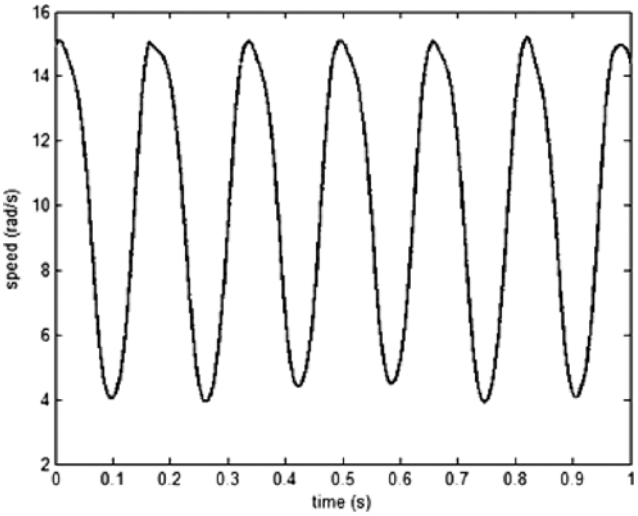

Fig. 7.16. Real time rotor speed estimation (plant signal in *solid line* and neural signal in *dashed line*)

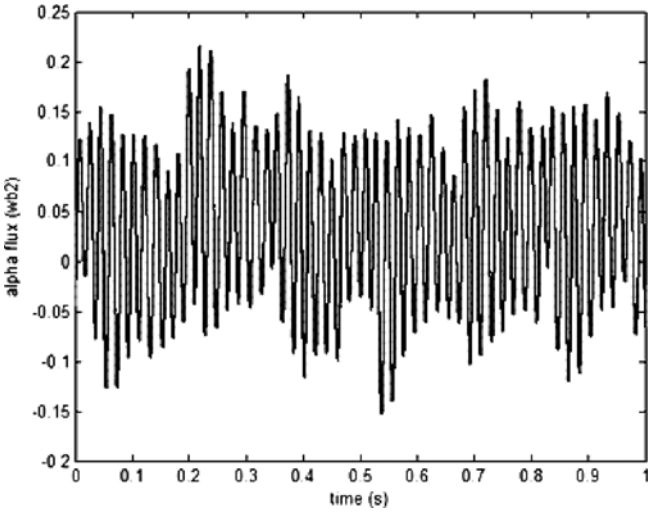

Fig. 7.17. Real time alpha flux estimation (plant signal in *solid line* and neural signal in *dashed line*)

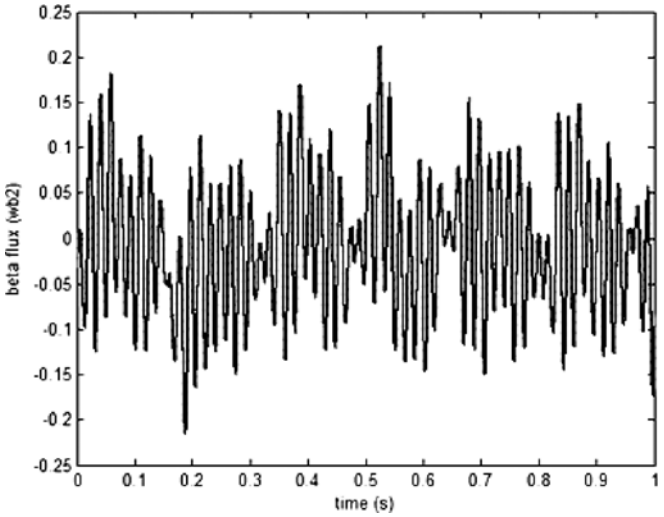

Fig. 7.18. Real time beta flux estimation (plant signal in *solid line* and neural signal in *dashed line*)

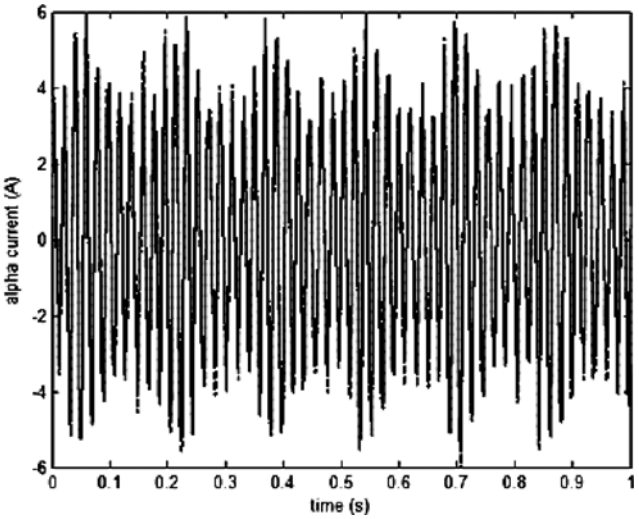

Fig. 7.19. Real time alpha current estimation (plant signal in *solid line* and neural signal in *dashed line*)

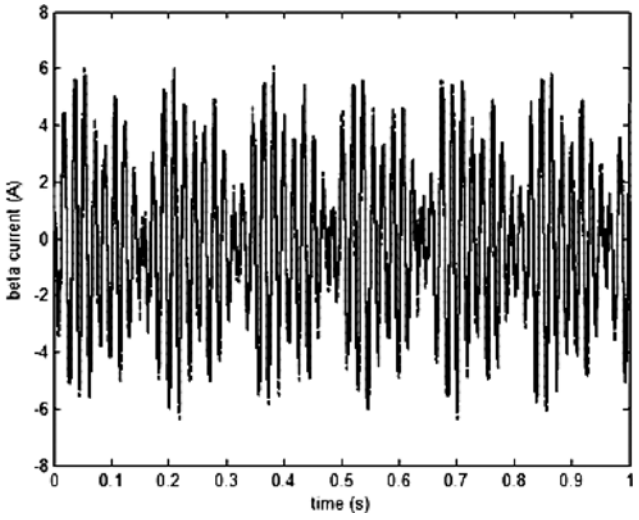

Fig. 7.20. Real time beta current estimation (plant signal in *solid line* and neural signal in *dashed line*)

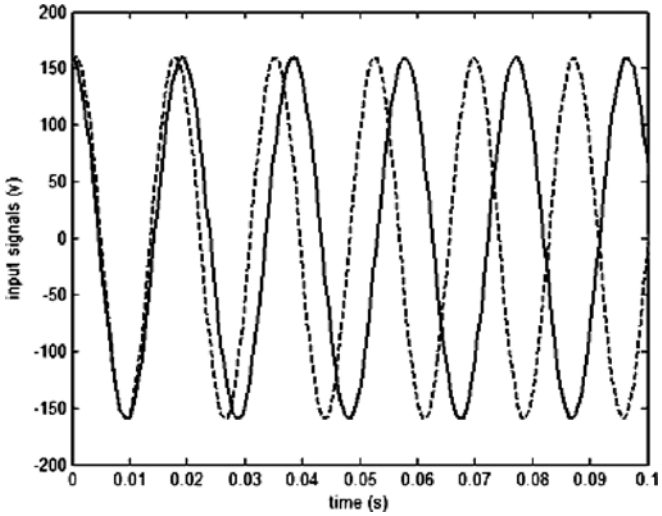

Fig. 7.21. Input signals applied during the state estimation process ($u^{\alpha}(k)$ in *solid line* and $u^{\beta}(k)$ in *dashed line*)

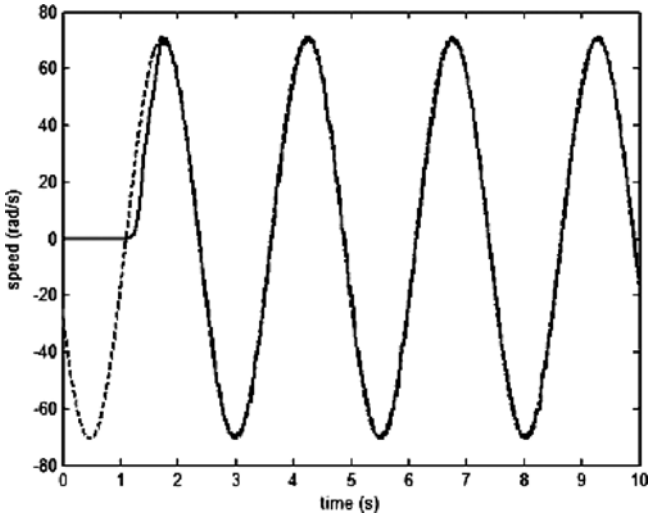

Fig. 7.22. Speed tracking performance (plant signal in *solid line* and reference signal in *dashed line*)

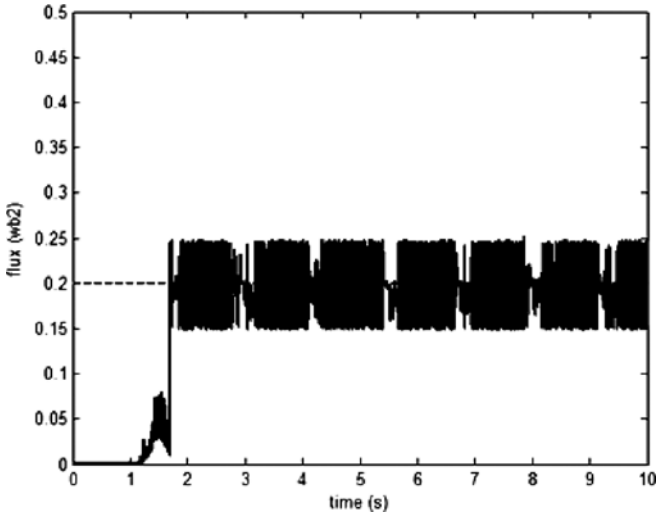

Fig. 7.23. Flux magnitude tracking performance (plant signal in *solid line* and reference signal in *dashed line*)

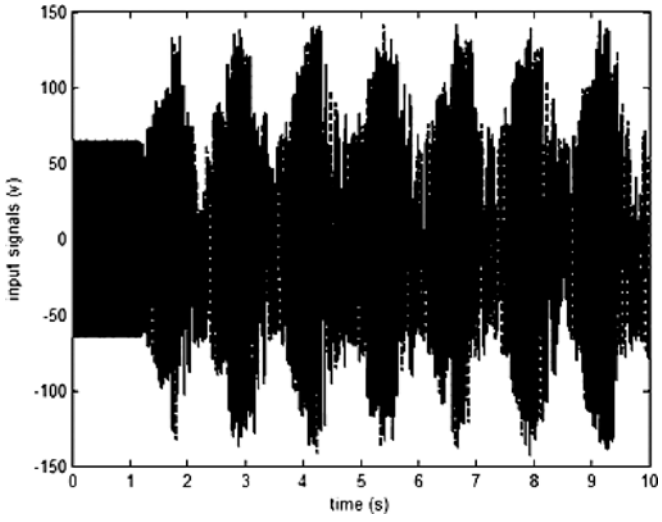

Fig. 7.24. Control law signals $u^{\alpha}(k)$ (*solid line*) and $u^{\beta}(k)$ (*dashed line*)

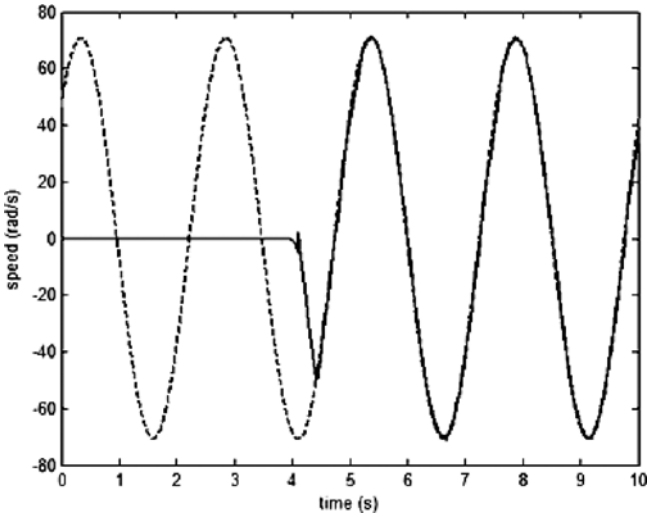

Fig. 7.25. Speed tracking performance (plant signal in *solid line* and reference signal in *dashed line*) with a constant load torque

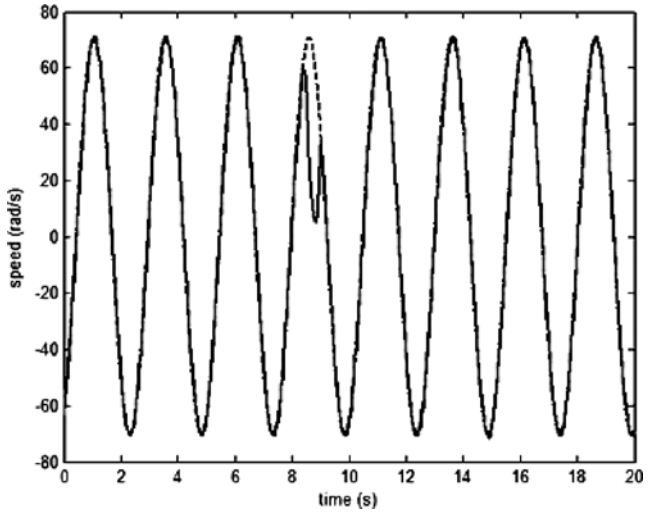

Fig. 7.26. Speed tracking performance (plant signal in *solid line* and reference signal in *dashed line*) under the presence of disturbances

7.4 Backstepping Control Using an RHONO

The real time results of the control law designed in Chap. 6, based on the backstepping technique approximated by a HONN using an RHONO, for the discrete-time induction motor model with a sampling time of 0.001 s are

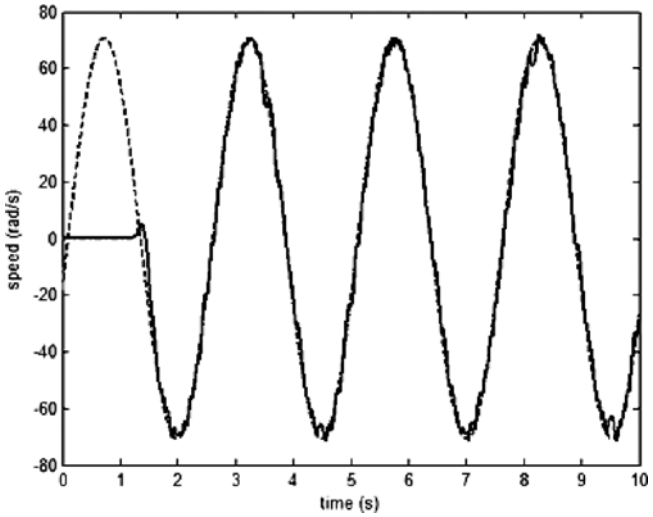

Fig. 7.27. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

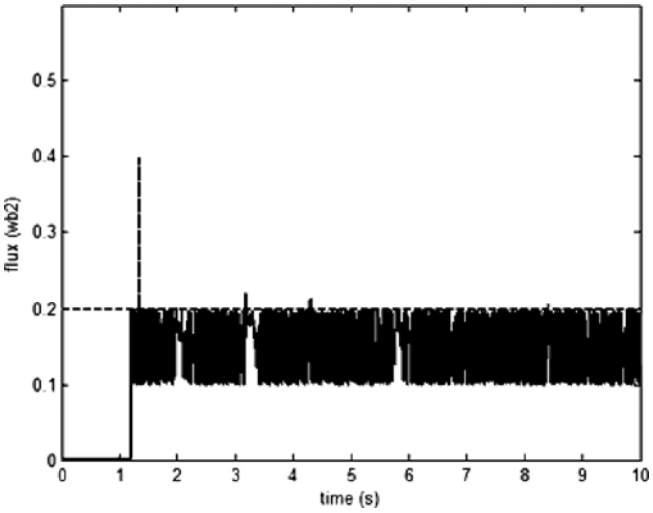

Fig. 7.28. Flux magnitude tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

presented as follows: The tracking results for the rotor speed and for the flux magnitude are presented in Figs. 7.27 and 7.28 for the induction motor working without load, respectively; Fig. 7.29 shows the control law in phases α and β; Fig. 7.30 presents the tracking result for the rotor speed under the presence of a load. Finally, Fig. 7.31 displays the tracking result under the presence of an external disturbance.

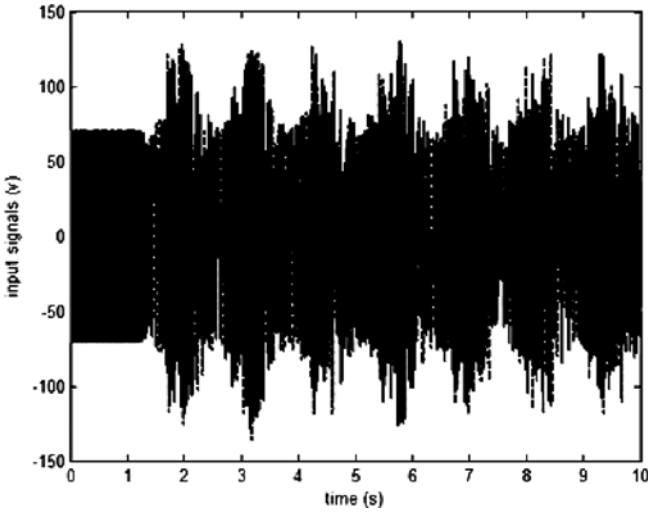

Fig. 7.29. Control law signals $u^{\alpha}(k)$ (*solid line*) and $u^{\beta}(k)$ (*dashed line*)

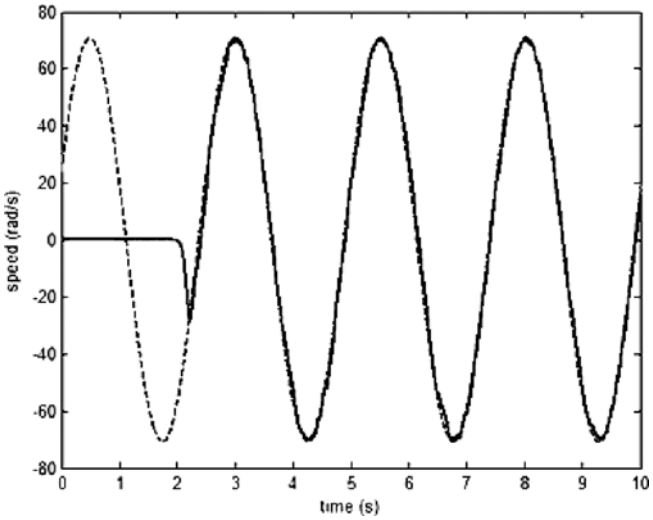

Fig. 7.30. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) with a constant load torque

7.5 Neural Block Control with Sliding Modes

The corresponding real time results of the control law designed in Chap. 4, based on the block control and sliding modes techniques, for the discrete-time induction motor with a sampling time of 0.001 s are presented as follows: The tracking results for the rotor speed and for the flux magnitude are presented

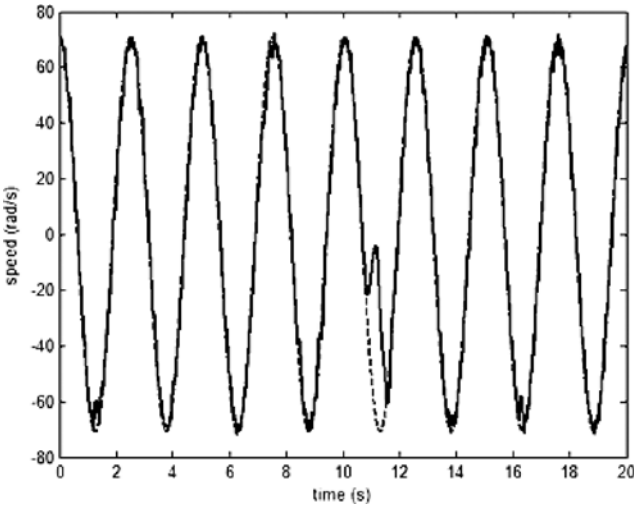

Fig. 7.31. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) under the presence of disturbances

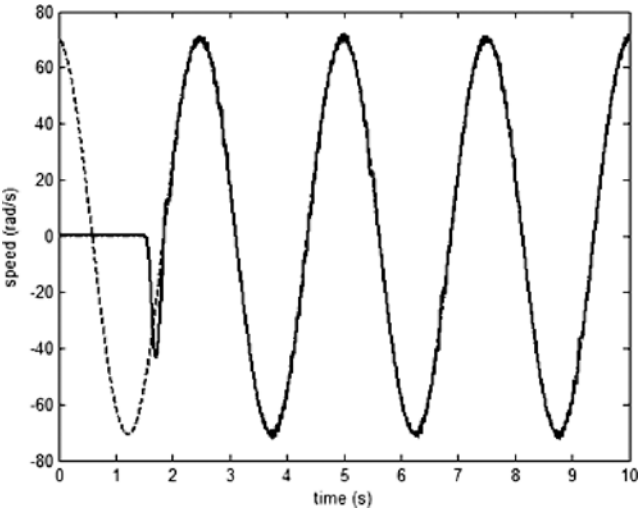

Fig. 7.32. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

in Figs. 7.32 and 7.33 for the induction motor working without load, respectively; Fig. 7.34 shows the control law in phases α and β; Fig. 7.35 presents the tracking result for the rotor speed with a constant load. Finally, Fig. 7.36 displays the tracking result under the presence of an external disturbance.

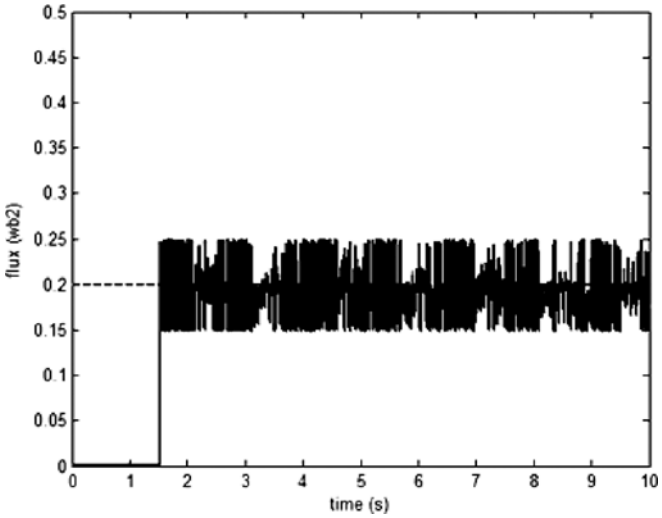

Fig. 7.33. Flux magnitude tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

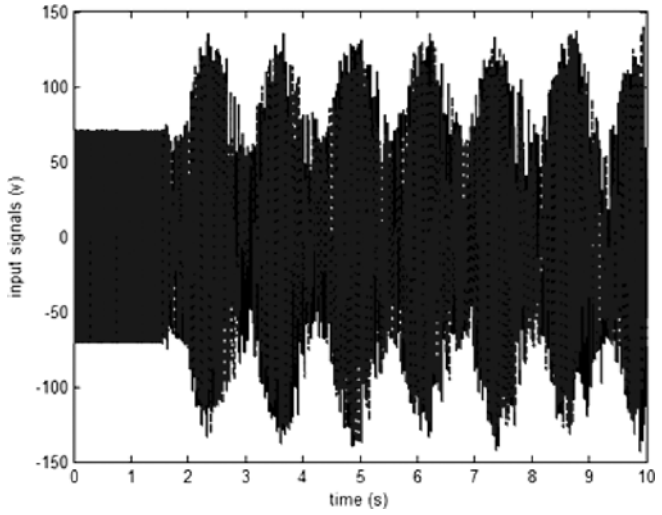

Fig. 7.34. Control law signals $u^{\alpha}(k)$ (*solid line*) and $u^{\beta}(k)$ (*dashed line*)

7.6 Block Control with Sliding Modes Using an RHONO

This section presents the real time results of the control law designed in Chap. 6 using the block control and sliding modes techniques, based on an RHONO model, for the discrete-time induction motor model with a sampling time of 0.001 s as follows: The tracking results for the rotor speed and for the

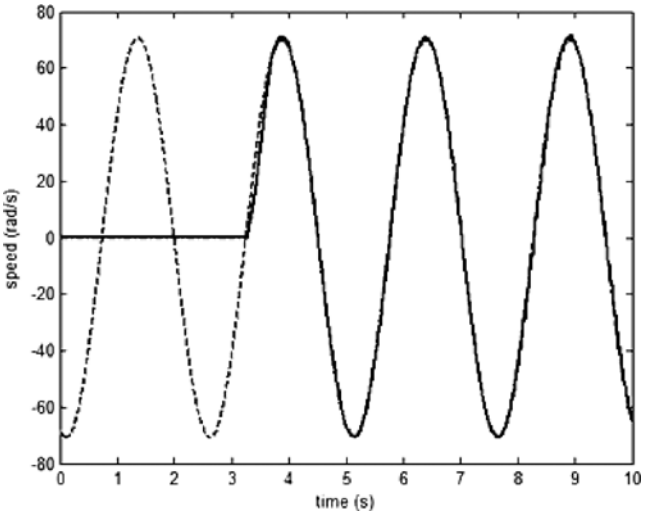

Fig. 7.35. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) with constant load torque

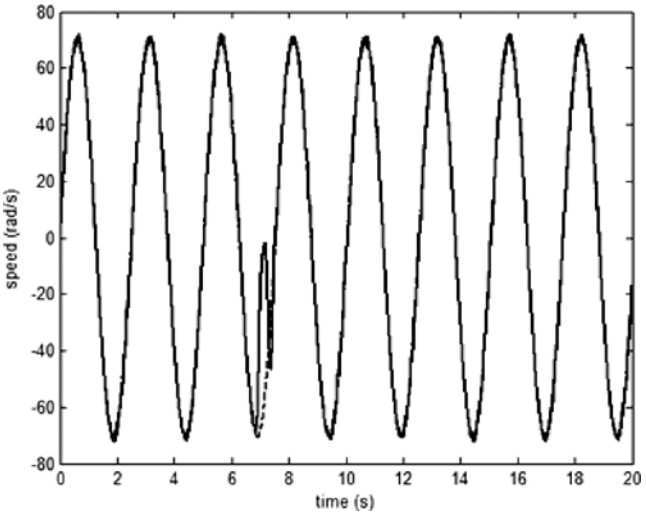

Fig. 7.36. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) under the presence of disturbances

flux magnitude are presented in Figs. 7.37 and 7.38 for the induction motor working without load, respectively; Fig. 7.39 shows the control law in phases α and β; Fig. 7.40 presents the tracking result for the rotor speed under the presence of a load. Finally, Fig. 7.41 displays the tracking result under the presence of an external disturbance.

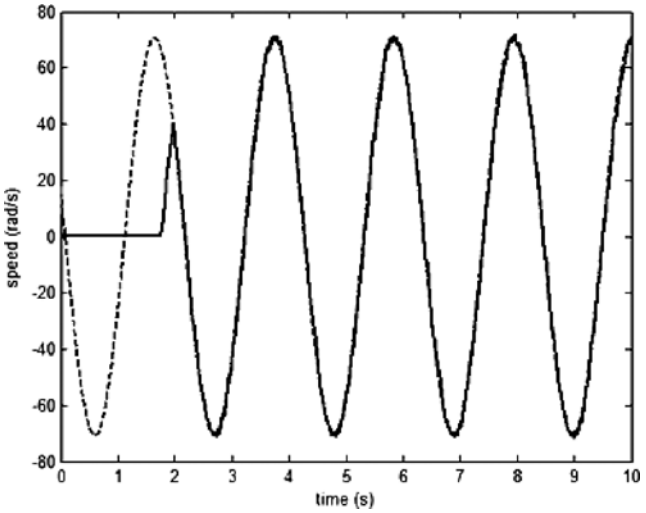

Fig. 7.37. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

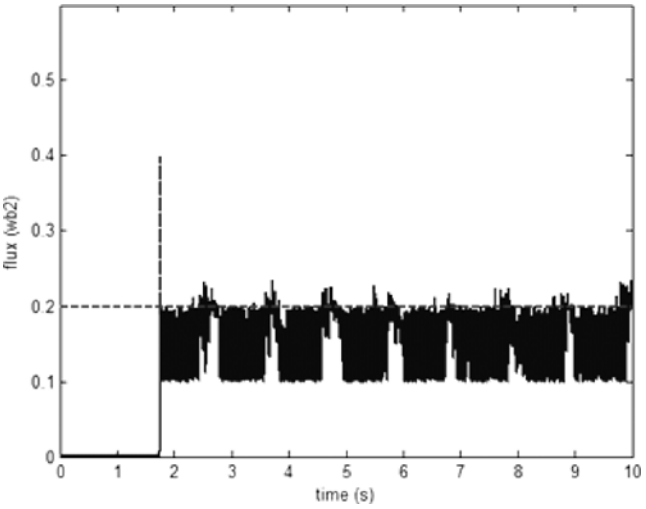

Fig. 7.38. Flux magnitude tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*)

7.7 Conclusions

To end this chapter, a comparative analysis of the four proposed schemes of control is included. In Table 7.1, the four schemes are compared with an induction motor operating without load and Table 7.2 establishes the comparison

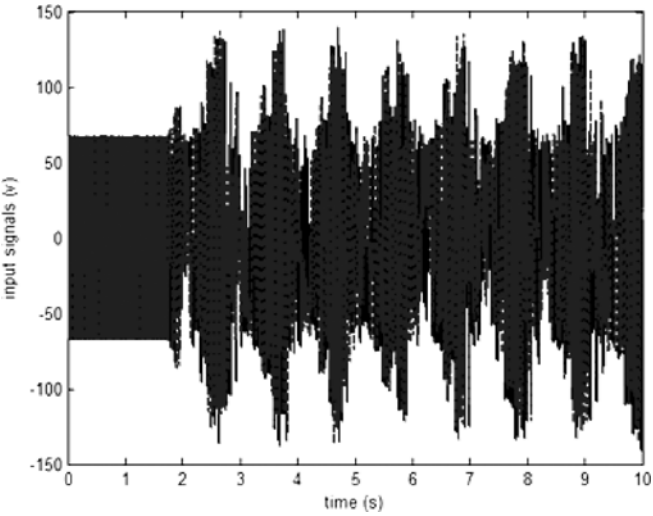

Fig. 7.39. Control law signals $u^{\alpha}(k)$ (*solid line*) and $u^{\beta}(k)$ (*dashed line*)

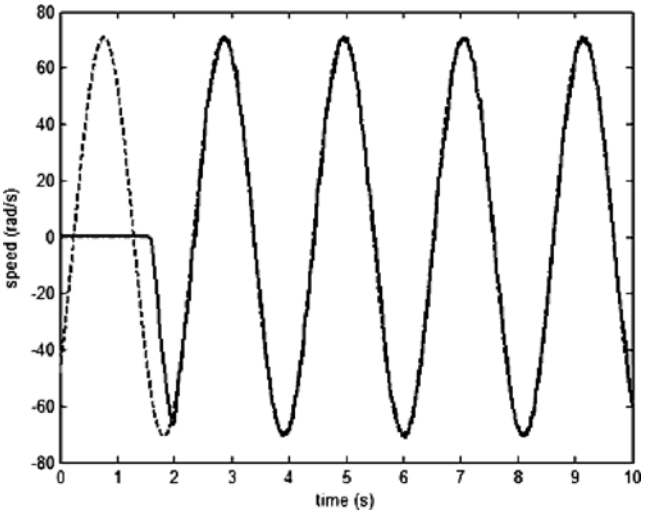

Fig. 7.40. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) with constant load torque

between the four schemes with the motor operating in presence of a constant load.

For Tables 7.1 and 7.2, B means backstepping technique, BNO means backstepping technique using an RHONO, BCNI means block control and sliding modes techniques, and BCNO means block control and sliding modes techniques using an RHONO.

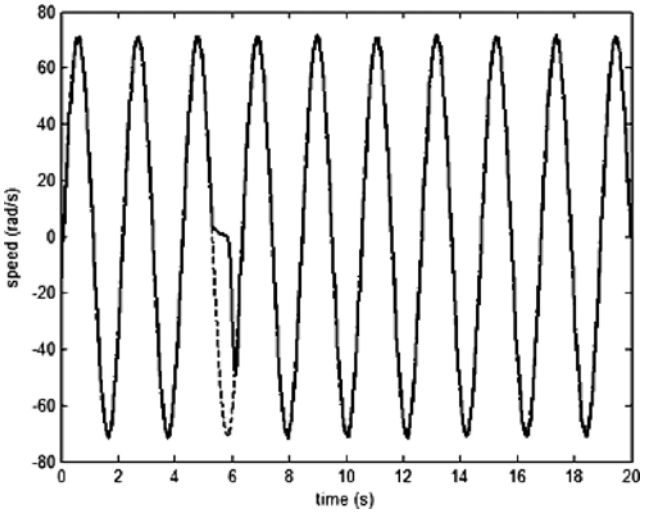

Fig. 7.41. Speed tracking performance (plant signal in *solid line*, neural signal in *dash-dot line*, and reference signal in *dashed line*) under the presence of disturbances

Table 7.1. Comparison of the mean square error for the controllers without load

Control algorithm	Mean square error
B	4.0130
BNO	8.4350
BCNI	1.9504
BCNO	4.5363

Table 7.2. Comparison of the mean square error for the controllers with load

Control algorithm	Mean square error
B	3.9158
BNO	4.6576
BCNI	1.7160
BCNO	5.2302

According to the mean square error presented above, the scheme with better performance are the ones based on the block control using the neural identifier and backstepping techniques, on the other hand the scheme with worse performance is the one based on the backstepping technique using the neural observer. It is important to remark that all the schemes present an excellent performance. However, the technique with the smaller computational complexity is the one based on the backstepping technique, since it allows to use a sampling time of 0.5 ms, whereas the other three schemes requires 1 ms.

8

Conclusions and Future Work

8.1 Conclusions

In this work, based on the neural network and feedback linearization techniques, a novel method to design robust control for a class of MIMO discrete-time nonlinear uncertain systems is proposed. This method includes four different control schemes, which can be applied depending on the state vector measurement viability:

a) The first designed robust *direct* neural control scheme is based on the backstepping technique, approximated by a high order neural network. On the basis of the Lyapunov approach, the respective stability analysis, for the whole closed-loop system, including the extended Kalman filter (EKF)-based NN learning algorithm, is also performed.
b) The second robust *indirect* control is designed with a recurrent high order neural network, which enables to identify the plant model. A strategy to avoid specific adaptive weights zero-crossing and conserve the identifier controllability property is proposed. Based on this neural identifier and applying the discrete-time block control approach, a nonlinear sliding manifold with a desired asymptotically stable motions was formulated. Using a Lyapunov functions approach, a discrete-time sliding mode control that makes the designed sliding manifold to be attractive was introduced.

Both the first and second control schemes require only the plant model structure knowledge, but the plant state vector must be available for the measurement. In the case when only the plant output is measured, some plant parameters are needed to design an observer.

c) For nonlinear plants whose mathematical model is assumed to be unknown and the only output vector can be measured, the third robust control scheme was designed. This strategy includes an adaptive recurrent neural observer, which estimates the state vector of the unknown plant dynamics. This observer has a Luenberger structure and is based on a recurrent high

E.N. Sanchez et al.: *Discrete-Time High Order Neural Control – Trained with Kalman Filtering*, Studies in Computational Intelligence (SCI) **112**, 95–96 (2008)
www.springerlink.com

order neural network (RHONN) trained by the EKF learning algorithm. Using the separation principle, stability conditions for the complete closed-loop system were derived.

d) Based on the above proposed RHONN observer, two *trajectory tracking* control policies were formulated. The first one was achieved by using the backstepping technique, while the second version was designed implementing the block control linearization approach techniques. The respective stability analysis carried out for the neural observer trained by the EKF learning algorithm and block controller has proved robustness of the closed-loop system.

Effectiveness of all the proposed schemes was illustrated via design, computer simulation, and real-time implementation of four discrete-time controllers designed for an induction motor. It was established that

e) For the controllers based on the first two proposed schemes ((a) and (b)), only the rotor loop parameters are needed to design the rotor flux nonlinear observer. The obtained flux estimates were then used in the *direct* backstepping high order neural network controller (a) and to train the RHONN identifier in the *indirect* controller (b).

f) For the controllers based on the RHONN observer ((c) and (d)), the knowledge of the motor parameters is not required.

The simulation and real-time implementation of the schemes proposed in this book are presented, validating the theoretical results, using a benchmark for a three phase induction motor.

The experimental results illustrate the robustness of the designed controllers with respect to motor parameter variations and the load torque (external disturbance).

8.2 Future Work

As a future work, it is worth to mention the following:

- Design robust controllers for electric power system using the developed in the book robust control method
- Design a discrete-time indirect control scheme of speed and limited current for an induction motor
- Design discrete-time neural observers of reduced order for nonlinear uncertain systems
- Design a sensorless neural observer in the benchmark of an induction motor
- Design of a bounded neural controller based on the backstepping technique

A

Causality Contradiction in Backstepping

For the design of adaptive controllers to be applied to discrete-time nonlinear systems in strict feedback form through backstepping, it is common to find the noncausal problem [5]. There are many ways to avoid the causality contradiction [4–6, 13]; in this appendix one of them is presented, basic idea for which is to consider the original system description as a one-step ahead predictor; then it is possible to transform the one-step ahead predictor into a equivalent maximum r-step ahead predictor.

The model of many practical nonlinear systems can be expressed in (or transformed into) a special state-space form named block strict feedback form (BSFF) [8] as follows:

$$\begin{aligned} x^i(k+1) &= f^i(\overline{x}^i(k)) + g^i(\overline{x}^i(k))x^{i+1}(k) + d^i(k), \qquad i = 1, 2, \ldots, r-1, \\ x^r(k+1) &= f^r(x(k)) + g^r(x(k))u(k) + d^r(k), \\ y(k) &= x^1(k), \end{aligned} \tag{A.1}$$

where $x(k) = [x^{1\top}(k), \ldots, x^{r\top}(k)]^\top$ are the state variables, and $\overline{x}^i(k) = [x^{1\top}, x^{2\top}, \ldots, x^{i\top}]^\top$, $x^i \in \Re^{n_i}$, $r \geq 2$, r is the number of blocks, $u(k) \in \Re^m$ is the system input, $y(k) \in \Re^m$ is the system output, for simplicity of notation through the remain of this book $d^i(k) = d^i(x(k), k) \in \Re^{n_i}$ is the bounded unknown disturbance vector, then there exists a constant $\overline{d}_i$ such that $\|d_i(k)\| \leq \overline{d}_i$ for $0 < k < \infty$, $f^i(\bullet)$ and $g^i(\bullet)$ are unknown smooth nonlinear functions.

In this section, as in [6], coordinate transformation are used to avoid the noncausal problem, which often appears in discrete-time nonlinear system control. We have assumed that the system (A.1) is in strict feedback form. It seems that the backstepping can be used to construct stable control. However, unlike in continuous-time systems, the causality contradiction is one of the major problems that will be encountered when we construct controls for strict-feedback discrete-time nonlinear system through backstepping, as detailed in the following.

Consider the system (A.1), if we design the ideal fictitious control for that system

$$\alpha_1^*(k) = -\frac{1}{g^1(\overline{x}^1(k))}[f^1(\overline{x}^1(k)) + d^1(k) - y_{\mathrm{d}}(k+1)],$$

the first block in (A.1) can be stabilized. Similarly, we can construct another ideal fictitious control

$$\alpha_2^*(k) = -\frac{1}{g^2(\overline{x}^2(k))}[f^2(\overline{x}^2(k)) + d^2(k) - \alpha_1^*(k+1)]$$

to stabilize the second equation in (A.1). But unfortunately, $\alpha_1^*(k+1)$ in (A.1) is a fictitious control of the future. This means that the fictitious control $\alpha_2^*(k)$ is infeasible in practice. If we continue the process to construct the final desired control $u^*(k)$, we end up with a $u^*(k)$ that is infeasible due to unavailable future information. However, the above problem can be avoided if we transform the system equation into a special form which is suitable for backstepping design. The basic idea is as follows. If we consider the original system description as a one-step ahead predictor, then we can transform the one-step ahead predictors into an equivalent maximum r-step ahead predictor which can predict the future states $x_1(k+r)$, $x_2(k+r-1), \ldots, x_r(k+1)$, then the causality contradiction is avoided when the controller is constructed based on the maximum r-step ahead predictor by backstepping. The transformation procedure is detailed as follows.

Consider the system (A.1)

$$x^i(k+1) = f^i(\overline{x}^i(k)) + g^i(\overline{x}^i(k))x^{i+1}(k) + d^i(k).$$

It can be easily obtained that $x^i(k+1)$ is a function of $\overline{x}^{i+1}(k)$. For convenience of analysis, we define

$$x^i(k+1) = f^{i(1)}(\overline{x}^{i+1}(k)), \tag{A.2}$$

with

$$f^{i(1)}(\overline{x}^{i+1}(k)) = f^i(\overline{x}^i(k)) + g^i(\overline{x}^i(k))x^{i+1}(k) + d^i(k).$$

Thus, we have

$$\overline{x}^i(k+1) = \begin{bmatrix} x^1(k+1) \\ \vdots \\ x^i(k+1) \end{bmatrix} = \begin{bmatrix} f^{1(1)}(\overline{x}^2(k)) \\ \vdots \\ f^{i(1)}(\overline{x}^{i+1}(k)) \end{bmatrix}, \quad i = 1, \ldots, r-1.$$

Then it is possible to define

$$\overline{x}^i(k+1) = \overline{f}^{i(1)}(\overline{x}^{i+1}(k)), \quad i = 1, \ldots, r-1. \tag{A.3}$$

After one more step, the first $r-1$ blocks of (A.1) can be expressed as

$$\begin{cases} x^i(k+2) = f^i(\overline{x}^i(k+1)) + g^i(\overline{x}^i(k+1))x^{i+1}(k+1) \\ \qquad\qquad + d^i(k+1), \\ x^{r-1}(k+2) = f^{r-1}(\overline{x}^{r-1}(k+1)) + g^{r-1}(\overline{x}^{r-1}(k+1))x^r(k+1) \\ \qquad\qquad + d^{r-1}(k+1), \end{cases} \tag{A.4}$$

$i = 1, 2, \ldots, r-2.$

Substituting (A.2) and (A.3) into (A.4), we can obtain

$$\begin{cases} x^i(k+2) = f^i(\overline{f}^{i(1)}(\overline{x}^{i+1}(k))) \\ \qquad\qquad + g^i(\overline{f}^{i(1)}(\overline{x}^{i+1}(k)))f^{i+1(1)}(\overline{x}^{i+2}(k)) + d^i(k+1) \\ \qquad\qquad \triangleq f^{i,2}(\overline{x}^{i+2}(k)), \\ x^{r-1}(k+2) = f^{r-1}(\overline{f}^{r-1(1)}(\overline{x}^{r-1}(k))) \\ \qquad\qquad + g^{r-1}(\overline{f}^{r-1(1)}(\overline{x}^{r-1}(k)))x^r(k+1) + d^{r-1}(k+1) \\ \qquad\qquad \triangleq \overline{f}^{r-1}(\overline{x}^{r-1}(k)) + \overline{g}^{r-1}(\overline{x}^{r-1}(k))x^r(k+1), \end{cases} \tag{A.5}$$

$i = 1, 2, \ldots, r-2,$

where

$$\begin{aligned} f^{i(2)}(\overline{x}^{i+2}(k)) &= f^i(\overline{f}^{i(1)}(\overline{x}^{i+1}(k))) \\ &\quad + g^i(\overline{f}^{i(1)}(\overline{x}^{i+1}(k)))f^{i+1(1)}(\overline{x}^{i+2}(k)) + d^i(k+1), \\ \overline{f}^{r-1}(x(k)) &= f^{r-1}(\overline{f}^{r-1(1)}(x(k))) + d^{r-1}(k+1), \\ \overline{g}^{r-1}(x(k))x^r(k+1) &= g^{r-1}(\overline{f}^{r-1(1)}(x(k)))x^r(k+1). \end{aligned}$$

Following the same procedure, the first $(r-2)$ blocks in (A.1) can be described by

$$\overline{x}^i(k+2) = \begin{bmatrix} x^1(k+2) \\ \vdots \\ x^i(k+2) \end{bmatrix} = \begin{bmatrix} f^{1,2}(\overline{x}^3(k)) \\ \vdots \\ f^{i,2}(\overline{x}^{i+2}(k)) \end{bmatrix}, \quad i = 1, \ldots, r-2,$$

which is a function of $\overline{x}^{i+2}(k)$ and is denoted as

$$\overline{x}^i(k+2)\overline{f}^{i(2)}(\overline{x}^{i+1}(k)), \quad i = 1, \ldots, r-2,$$

Continuing the above procedure recursively, after $(r-2)$ steps, the first two blocks of system (A.1) can be written as

$$\begin{cases} x^1(k+r-1) = f^{1(2)}(\overline{x}^1(k)), \\ x^2(k+r-1) = \overline{f}^2(\overline{x}^2(k)) + \overline{g}^2(\overline{x}^2(k))x^3(k+r-2), \end{cases} \tag{A.6}$$

where

$$\begin{aligned} f^{1,2}(\overline{x}^1(k)) &= f^1\big(\overline{f}^{1(3)}(\overline{x}^1(k))\big) + g^1\big(\overline{f}^{1(3)}(\overline{x}^1(k))\big) f^{2(3)}(\overline{x}^2(k)) \\ &\quad + d(k+r-1), \\ \overline{g}^2(\overline{x}^{2(2)}(k)) &= f^2\big(\overline{f}^{2(3)}(\overline{x}^2(k))\big), \\ \overline{g}^2(\overline{x}^2(k)) &= g^2\big(\overline{f}^{2(3)}(\overline{x}^2(k))\big). \end{aligned}$$

After one more step, the first block of (A.1) becomes

$$x^1(k+r) = \overline{f}^1(\overline{x}^1(k)) + \overline{g}^1(\overline{x}^1(k))x^2(k+r-1), \tag{A.7}$$

where

$$\begin{aligned} \overline{f}^1(X(k)) &= f^1(f^{1(2)}(\overline{x}^1(k))) + d(k+r), \\ \overline{g}^1(X(k)) &= g^1(f^{1(2)}(\overline{x}^1(k))). \end{aligned}$$

Equations (A.4)–(A.7) are all derived from the original system, which is equivalent to

$$\begin{aligned} x^1(k+r) &= \overline{f}^1(\overline{x}^1(k)) + \overline{g}^1(\overline{x}^1(k))x^2(k+r-1) + d^1(k+r-1), \\ &\vdots \\ x^{r-1}(k+2) &= \overline{f}^{r-1}(\overline{x}^{r-1}(k)) + \overline{g}^{r-1}(\overline{x}^{r-1}(k))x^r(k+1) + d^{r-1}(k+2), \\ x^r(k+1) &= \overline{f}^r(x(k)) + \overline{g}^r(x(k))u(k) + d^r(k), \\ y(k) &= x^1(k). \end{aligned} \tag{A.8}$$

For convenience of analysis, let us define $i = 1, \ldots, r-1$

$$\begin{aligned} \overline{f}^i(k) &\triangleq \overline{f}^i(\overline{x}_i(k)), \\ \overline{g}^i(k) &\triangleq \overline{g}^i(\overline{x}_i(k)), \\ \overline{f}^r(k) &\triangleq \overline{f}^r(x(k)), \\ \overline{g}^r(k) &\triangleq \overline{g}^r(x(k)), \end{aligned}$$

where $\overline{f}^i(\bullet)$ and $\overline{g}^i(\bullet)$ are unknown functions of $f^i(\overline{x}^i(k))$ and $g^i(\overline{x}^i(k))$, respectively.

Then, system (A.8) can be written as

$$\begin{aligned} x^1(k+r) &= \overline{f}^1(k) + \overline{g}^1(k)x^2(k+r-1) + d^1(k+r-1), \\ &\vdots \\ x^{r-1}(k+2) &= \overline{f}^{r-1}(k) + \overline{g}^{r-1}(k)x^r(k+1) + d^{r-1}(k+2), \\ x^r(k+1) &= \overline{f}^r(k) + \overline{g}^r(k)u(k) + d^r(k), \\ y(k) &= x_1(k). \end{aligned} \tag{A.9}$$

Comment A.1. For more detailed explanation, please see [4–6].

Comment A.2. A different very interesting solution to the causality contradiction is presented in [1–3,12]; there, the noncausality problem for discrete-time backstepping control is solved by using a filtered prediction.

References

Chapter 1

1. A. Y. Alanis, *Neural network training using Kalman filtering*, Master Dissertation, Cinvestav, Unidad Guadalajara, Guadalajara Jalisco Mexico, 2004 (in Spanish)
2. F. Chen and H. Khalil, Adaptive control of a class of nonlinear discrete-time systems using neural networks, *IEEE Transactions on Automatic Control*, 40(5), 791–801, 1995
3. L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks, *Neural Networks*, 16, 683–689, 2003
4. S. S. Ge, J. Zhang, and T. H. Lee, Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4), 1630–1645, 2004
5. R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, USA, 1992
6. S. Haykin, *Kalman Filtering and Neural Networks*, Wiley, New York, USA, 2001
7. S. Jagannathan, Control of a class of nonlinear discrete-time systems using multilayer neural networks, *IEEE Transactions on Neural Networks*, 12(5), 1113–1120, 2001
8. F. L. Lewis, J. Campos, and R. Selmic, *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society of Industrial and Applied Mathematics Press, Philadelphia, 2002
9. F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis, London, 1999
10. K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, 1, 4–27, 1990
11. A. S. Poznyak, E. N. Sanchez, and W. Yu, *Differential Neural Networks for Robust Nonlinear Control*, World Scientific, Singapore, 2001
12. G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High Order Neural Networks*, Springer, Berlin Hiedelberg New York, 2000

13. E. N. Sanchez, A. Y. Alanis, and G. Chen, Recurrent neural networks trained with Kalman filtering for discrete chaos reconstruction, *Dynamics of Continuous, Discrete and Impulsive Systems Series B*, 13, 1–18, 2006
14. E. N. Sanchez and L. J. Ricalde, Trajectory tracking via adaptive recurrent neural control with input saturation, *Proceedings of International Joint Conference on Neural Networks'03*, Portland, Oregon, USA, July 2003
15. S. Singhal and L. Wu, Training multilayer perceptrons with the extended Kalman algorithm, in *Advances in Neural Information Processing Systems*, vol. 1, ed. by D. S. Touretzky, Morgan Kaufmann, San Mateo, CA, USA, 1989, pp. 133–140

Chapter 2

1. A. Y. Alanis, *Neural network training using Kalman filtering*, Master Dissertation, Cinvestav, Unidad Guadalajara, Guadalajara Jalisco Mexico, 2004 (in Spanish).
2. N. Cotter, The Stone–Weiertrass theorem and its application to neural networks, *IEEE Transactions on Neural Networks*, 1(4), 290–295, 1990
3. L. A. Feldkamp, D. V. Prokhorov, and T. M. Feldkamp, Simple and conditioned adaptive behavior from Kalman filter trained recurrent networks, *Neural Networks*, 16, 683–689, 2003
4. R. A. Felix, *Variable Structure Neural Control*, Ph.D. Dissertation, Cinvestav, Unidad Guadalajara, Guadalajara Jalisco Mexico, 2004
5. S. S. Ge, J. Zhang, and T. H. Lee, Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4), 1630–1645, 2004
6. S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control for Robotic Manipulators*, World Scientific, Singapore, 1998
7. R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, USA, 1992
8. S. Haykin, *Kalman Filtering and Neural Networks*, Wiley, New York, USA, 2001
9. H. Khalil, *Nonlinear Systems*, 2nd ed., Prentice Hall, Upper Saddle River, N.J., USA, 1996
10. C. Leunga and L. Chan, Dual extended Kalman filtering in recurrent neural networks, *Neural Networks*, 16, 223–239, 2003
11. A. U. Levin and K. S. Narendra, Control of nonlinear dynamical systems using neural networks, Part 2: observability, identification and control, *IEEE Transactions on Neural Networks*, 7(1), 30–42, 1996
12. W. Lin and C. I. Byrnes, Design of discrete-time nonlinear control systems via smooth feedback, *IEEE Transactions on Automatic Control*, 39(11), 2340–2346, 1994
13. K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Transactions on Neural Networks*, 1, 4–27, 1990
14. M. Norgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer, Berlin Hiedelberg New York, USA, 2000

15. A. S. Poznyak, E. N. Sanchez, and W. Yu, *Differential Neural Networks for Robust Nonlinear Control*, World Scientific, Singapore, 2001
16. G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High Order Neural Networks*, Springer, Berlin Hiedelberg New York, 2000
17. W. J. Rugh, *Linear System Theory*, 2nd ed., Prentice Hall, Upper Saddle River, N.J., USA, 1996
18. E. N. Sanchez, A. Y. Alanis, and G. Chen, Recurrent neural networks trained with Kalman filtering for discrete chaos reconstruction, *Dynamics of Continuous, Discrete and Impulsive Systems Series B*, 13, 1–18, 2006
19. E. N. Sanchez and L. J. Ricalde, Trajectory tracking via adaptive recurrent neural control with input saturation, *Proceedings of International Joint Conference on Neural Networks'03*, Portland, Oregon, USA, July 2003
20. Y. Song and J. W. Grizzle, The extended Kalman filter as local asymptotic observer for discrete-time nonlinear systems, *Journal of Mathematical Systems, Estimation and Control*, 5(1), 59–78, 1995
21. R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, 1, 270–280, 1989

Chapter 3

1. J. Campos, F. L. Lewis, and R. Selmic, Backlash compensation in discrete-time nonlinear systems using dynamic inversion by neural networks, *Proceedings IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000, pp. 1289–1295
2. G. Cybenko, Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, 2, 304–314, 1989
3. F. Chen and H. Khalil, Adaptive control of a class of nonlinear discrete-time systems using neural networks, *IEEE Transactions on Automatic Control*, 40(5), 791–801, 1995
4. S. S. Ge, J. Zhang, and T. H. Lee, Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4), 1630–1645, 2004
5. S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control for Robotic Manipulators*, World Scientific, Singapore, 1998
6. S. Jagannathan, Control of a class of nonlinear discrete-time systems using multilayer neural networks, *IEEE Transactions on Neural Networks*, 12(5), 1113–1120, 2001
7. F. Khorrami, P. Krishnamurthy, and H. Melkote, *Modeling and Adaptive Nonlinear Control of Electric Motors*, Springer, Berlin Hiedelberg New York, 2003
8. Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks,* World Scientific, Singapore, 1998
9. M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*, Wiley, New York, USA, 1995
10. F. L. Lewis, J. Campos, and R. Selmic, *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society of Industrial and Applied Mathematics Press, Philadelphia, 2002

11. F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor and Francis, London, 1999
12. Y. Li, S. Quiang, X. Zhuang, and O. Kanak, Robust and adaptive backstepping control for nonlinear systems using RBF neural networks, *IEEE Transactions on Neural Networks*, 15, 693–701, 2004
13. A. G. Loukianov, J. Rivera, and J. M. Cañedo, Discrete-time sliding mode control of an induction motor, *Proceedings IFAC'02*, Barcelone, Spain, July 2002
14. Y. Song and J. W. Grizzle, The extended Kalman filter as local asymptotic observer for discrete-time nonlinear systems, *Journal of Mathematical Systems, Estimation and Control*, 5(1), 59–78, 1995
15. V. Utkin, J. Guldner, and J. Shi, *Sliding Mode Control in Electromechanical Systems*, Taylor and Francis, Philadelphia, USA, 1999

Chapter 4

1. R. A. Felix, E. N. Sanchez and A. G. Loukianov, Avoiding controller singularities in adaptive recurrent neural control, *Proceedings IFAC'05*, Prague, Czech Republic, July 2005
2. R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, USA, 1992
3. Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks,* World Scientific, Singapore, 1998
4. A. G. Loukianov, J. Rivera, and J. M. Cañedo, Discrete-time sliding mode control of an induction motor, *Proceedings IFAC'02*, Barcelone, Spain, July 2002
5. G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High-Order Neural Networks*, Springer, Berlin Hieldelberg New York, Germany, 2000
6. Y. Song and J. W. Grizzle, The extended Kalman filter as local asymptotic observer for discrete-time nonlinear systems, *Journal of Mathematical Systems, Estimation and Control*, 5(1), 59–78, 1995
7. V. Utkin, J. Guldner and J. Shi, *Sliding Mode Control in Electromechanical Systems*, Taylor and Francis, Philadelphia, USA, 1999
8. W. Yu and X. Li, Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms, *Information Sciences,* 158, 131–147, 2004

Chapter 5

1. R. Grover and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 2nd ed., Wiley, New York, USA, 1992
2. F. Khorrami, P. Krishnamurthy, and H. Melkote, *Modeling and Adaptive Nonlinear Control of Electric Motors*, Springer, Berlin Hiedelberg New York, 2003
3. Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks,* World Scientific, Singapore, 1998
4. A. J. Krener and A. Isidori, Linearization by output injection and nonlinear observers, *System and Control Letters*, 3, 47–52, 1983

5. A. U. Levin and K. S. Narendra, Control of nonlinear dynamical systems using neural networks, Part 2: observability, identification and control, *IEEE Transactions on Neural Networks*, 7(1), 30–42, 1996
6. R. Marino, Observers for single output nonlinear systems, *IEEE Transactions on Automatic Control,* 35, 1054–1058, 1990
7. S. Nicosia and A. Tornambe, High-gain observers in the state and parameter estimation of robots having elastic joins, *System and Control Letters*, 13, 331–337, 1989
8. G. A. Rovithakis and M. A. Chistodoulou, *Adaptive Control with Recurrent High-Order Neural Networks*, Springer, Berlin Hiedelberg New York, 2000
9. E. N. Sanchez, A. Y. Alanis, and G. Chen, Recurrent neural networks trained with Kalman filtering for discrete chaos reconstruction, *Dynamics of Continuous, Discrete and Impulsive Systems Series B*, 13, 1–18, 2006
10. E. N. Sanchez and L. J. Ricalde, Trajectory tracking via adaptive recurrent neural control with input saturation, *Proceedings of International Joint Conference on Neural Networks'03*, Portland, Oregon, USA, July 2003
11. B. L. Walcott and S. H. Zak, State observation of nonlinear uncertain dynamical system, *IEEE Transactions on Automatic Control*, 32, 166–170, 1987
12. Q. Zhu and L. Guo, Stable adaptive neurocontrol for nonlinear discrete-time systems, *IEEE Transactions on Neural Networks*, 15(3), 653–662, 2004

Chapter 6

1. G. Chen and J. L. Moiola, An overview of bifurcation, chaos and nonlinear dynamics in control systems, *The Franklin Institute Journal*, 331B, 819–858, 1994
2. R. A. Felix, E. N. Sanchez, and A. G. Loukianov, Avoiding controller singularities in adaptive recurrent neural control, *Proceedings IFAC'05*, Prague, Czech Republic, July 2005
3. S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control for Robotic Manipulators*, World Scientific, Singapore, 1998
4. H. Khalil, *Nonlinear Systems*, 2nd ed., Prentice Hall, Upper Saddle River, NJ, USA, 1996
5. F. Khorrami, P. Krishnamurthy, and H. Melkote, *Modeling and Adaptive Nonlinear Control of Electric Motors*, Springer, Berlin Hiedelberg New York, 2003
6. M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*, Wiley, New York, USA, 1995
7. W. Lin and C. I. Byrnes, Design of discrete-time nonlinear control systems via smooth feedback, *IEEE Transactions on Automatic Control*, 39(11), 2340–2346, 1994
8. A. G. Loukianov, J. Rivera, and J. M. Cañedo, Discrete-time sliding mode control of an induction motor, *Proceedings IFAC'02*, Barcelone, Spain, July 2002
9. E. N. Sanchez, A. Y. Alanis, and G. Chen, Recurrent neural networks trained with Kalman filtering for discrete chaos reconstruction, *Dynamics of Continuous, Discrete and Impulsive Systems Series B*, 13, 1–18, 2006
10. E. N. Sanchez and L. J. Ricalde, Trajectory tracking via adaptive recurrent neural control with input saturation, *Proceedings of International Joint Conference on Neural Networks'03*, Portland, Oregon, USA, July 2003

Chapter 7

1. J. Quiñones, *Real time implementation of a three phase induction motor control*, Master Dissertation, Cinvestav, Unidad Guadalajara, Guadalajara Jalisco Mexico, 2006 (in Spanish)

Appendix

1. J. Campos, F. L. Lewis, and R. Selmic, Backlash compensation in discrete time nonlinear systems using dynamic inversion by neural networks, *Proceedings IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000, 1289–1295
2. J. Campos and F. L. Lewis, Backlash compensation with filtered prediction in discrete time nonlinear systems by dynamic inversion using neural networks, *Proceedings IEEE Conference on Decision and Control*, Sydney, Australia, December, 2000
3. J. Campos and F. L. Lewis, Backlash compensation with filtered prediction in discrete time nonlinear systems by dynamic inversion using neural networks, *Asian Journal of Control*, 6(3), 362–375, 2004
4. F. Chen and H. Khalil, Adaptive control of a class of nonlinear discrete-time systems using neural networks, *IEEE Transactions on Automatic Control*, 40(5), 791–801, 1995
5. S. S. Ge, J. Zhang, and T. H. Lee, Adaptive neural network control for a class of MIMO nonlinear systems with disturbances in discrete-time, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4), 1630–1645, 2004
6. S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control for Robotic Manipulators*, World Scientific, Singapore, 1998
7. Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks,* World Scientific, Singapore, 1998
8. M. Krstic, I. Kanellakopoulos, and P. Kokotovic, *Nonlinear and Adaptive Control Design*, Wiley, New York, USA, 1995
9. A. J. Krener and A. Isidori, Linearization by output injection and nonlinear observers, *System and Control Letters*, 3, 47–52, 1983
10. C. Leunga and L. Chan, Dual extended Kalman filtering in recurrent neural networks, *Neural Networks*, 16, 223–239, 2003
11. A. U. Levin and K. S. Narendra, Control of nonlinear dynamical systems using neural networks, Part 2: observability, identification and control, *IEEE Transactions on Neural Networks*, 7(1), 30–42, 1996
12. F. L. Lewis, J. Campos, and R. Selmic, *Neuro-Fuzzy Control of Industrial Systems with Actuator Nonlinearities*, Society of Industrial and Applied Mathematics Press, Philadelphia, 2002
13. A. G. Loukianov, B. Castillo-Toledo, and S. Dodds, Robust stabilization of a class of uncertain system via block decomposition and VSC, *International Journal of Robust Nonlinear Control*, 12, 1317–1338, 2002

Index