World Scientific Series in Robotics and Intelligent Systems – Vol. 18

SOFT COMPUTING IN SYSTEMS AND CONTROL TECHNOLOGY

Editor

S G TZAFESTAS

National Technical University of Athens

World Scientific

Singapore • New Jersey • London • Hong Kong

Published by

World Scientific Publishing Co. Pte. Ltd.

P O Box 128, Farrer Road, Singapore 912805

USA office: Suite 1B, 1060 Main Street, River Edge, NJ 07661

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

SOFT COMPUTING IN SYSTEMS AND CONTROL TECHNOLOGY
World Scientific Series in Robotics and Intelligent Systems — Vol. 18

ISBN 981-02-3381-7

Printed in Singapore by Uto-Print

PREFACE

Soft computing is a new branch of scientific computing, which, unlike hard computing, can deal with uncertain, imprecise and inexact data. This field is currently expanding and finds increasing application in modern-life problems and systems. The three constituents of soft computing are fuzzy-logic-based computing, neurocomputing and genetic algorithms. Fuzzy logic provides the capability of approximate and commonsense reasoning, neurocomputing offers learning and function approximation capabilities, and genetic algorithms (GAs) provide a methodology for systematic random search optimization which is based on the mechanics of natural selection and natural genetics. In hybrid soft-computing systems these three capabilities are combined in various complementary and synergetic ways.

This book provides a set of selected contributions dealing with important issues, techniques and applications of soft-computing in systems and control technology. It is divided in the following four parts:

Part I: Neural networks in system identification and control
Part II: Fuzzy and neurofuzzy systems in modeling, control and robot path planning
Part III: Genetic – Evolutionary algorithms
Part IV: Soft computing applications

PART I involves five chapters dealing with advanced neural-network controllers. In particular, chapter 1 provides a comprehensive study on supervised learning and the back propagation (BP) algorithm, and chapter 2 deals with the neural-network based identification of 2-dimensional systems in state space representation. The neural network models employed are: multilayer perceptrons, radial basis function networks, recurrent networks, weightless networks and adaptive resonance theory (ART) network structures. The neurocontrollers derived and implemented include adaptive regulators and generalized predictive controllers via local model networks.

PART II contains three chapters that present state-of-the-art material on the fuzzy and neurofuzzy approach to system modeling, control and robot path planning. Particular topics covered in this part are: functional reasoning (Takagi-Sugeno-Kang model), adaptive-network-based fuzzy inference system (ANFIS), fuzzy adaptive resonance theory (fuzzy ART), combined TSK-ART system modeling, fuzzy nonlinear autoregressive modeling with exogenous variables (FNARX), fuzzy nonlinear output error (FNOE) model, neurofuzzy model-based cancellation control, neurofuzzy model-based predictive control, fuzzy and neurofuzzy system architectures, manipulator fuzzy obstacle avoidance, and mobile robot fuzzy and neurofuzzy motion planning and navigating control.

PART III involves three chapters devoted to genetic and evolutionary algorithms and their implementation in practical problems. The first chapter provides a tutorial overview of genetic algorithms (GAs) starting with the fundamental concepts and a simple genetic algorithm, and showing how GAs work. Then, several modifications of the simple GA are presented and some applications of GAs are outlined. The other two chapters provide a set of experimental results obtained when applying GAs to multiple fault diagnosis, network configuration, edge detection in computer vision, fish distribution determination, route finding and computer aided design of integrated circuits.

Finally, **PART IV** presents comprehensive investigations of the application of soft computing techniques to adaptive and intelligent data fusion, computer gaming, control, signal processing, and environmental systems. Particular topics studied here are: fuzzy low-level and high-level fusion, neural fusing techniques, the COMMONS GAME via neural learning, coherent neural networks in time – sequential signal processing and control, evolutionary reinforcement neural fuzzy systems, industrial air emissions, modeling of kraft recovery boiler via neural networks, and neural applications in ambient air quality, honey bees for air biomonitoring, and aqueous contaminants.

All contributions have been prepared by leading experts in the field and include state-of-the-art material, fresh results and most of them practical how-to-do issues. Taken together they provide a long-standing and well balanced source on soft computing applications in systems and control. The book can serve as a reference volume for the researcher and practitioner in the field.

The editor is grateful to all contributors for their timely and high-quality contributions, and to the World Scientific's editorial staff members for their encouragement and support throughout the lengthy editorial process.

Athens, October 1998 *Spyros G. Tzafestas*

CONTRIBUTORS

ANTHOPOULOS Y.	Natl. Tech. Univ. of Athens, Athens, Greece
BABA N.	Osaka Kyoiku Univ., Osaka Prefecture, Japan
BECKER B.	Albert-Ludwigs Univ., Freiburg, Germany
BHANDARKAR S.	Univ. of Georgia, Athens, Georgia, U.S.A.
D'ANGELO D.	Univ. of Georgia, Athens, Georgia, U.S.A.
DRECHSLER N.	Albert-Ludwigs Univ., Freiburg, Germany
DRECHSLER R.	Albert-Ludwigs Univ., Freiburg, Germany
ESBENSEN H.	Avant Corporation, Fremont, CA, U.S.A.
GAWTHROP P.	Univ. of Glasgow, Glasgow, U.K.
HIROSE A.	Resarch Center for Adv. Sci. & Tech., Tokyo, Japan
KANEKO M.	Hiroshima Univ., Higashi-Hiroshima, Japan
KAVŠEK-BIASIZZO K.	Univ. of Ljubljana, Ljubljana, Slovenia
KOCIJAN J.	Univ. of Ljubljana, Ljubljana, Slovenia
LINKENS D.	Univ. of Sheffield, Sheffield, U.K.
LOONEY C.	Univ. of Nevada, Reno, NV, U.S.A.
MARKAKI M.	Natl. Tech. Univ. of Athens, Athens, Greece
MATKO D.	Univ. of Ljubljana, Ljubljana, Slovenia
MITCHELL R.	University of Reading, Reading, Berks., U.K.
NYONGESA H.	Brunel Univ., Uxbridge, U.K.
POTTER W.	Univ. of Georgia, Athens, Georgia, U.S.A.
RONCO E.	Univ. of Glasgow, Glasgow, U.K.
SALTOUROS M.-P.	Natl. Tech. Univ. of Athens, Athens, Greece
SMITH G.	Univ. of Montana-Missoula, Missoula, MT, U.S.A.
TSUJI T.	Hiroshima Univ., Higashi-Hiroshima, Japan
TZAFESTAS C.	Natl. Tech. Univ. of Athens, Athens, Greece
TZAFESTAS S.	Natl. Tech. Univ. of Athens, Athens, Greece
VAROL Y.	Univ. of Nevada, Reno, NV, U.S.A.
WANG D.	Florida Atlantic Univ., Boca Raton, FL, U.S.A.
WROBEL C.	Univ. of Montana-Missoula, Missoula, MT, U.S.A.
XU B.	Yamamoto Electric Corp., Fukushima, Japan
ZIKIDIS K.	Natl. Tech. Univ. of Athens, Athens, Greece
ZILOUCHIAN A.	Florida Atlantic Univ., Boca Raton, FL, U.S.A.

Chapter 4

Neuro-Based Adaptive Regulator

T.Tsuji, B.H.Xu and M.Kaneko

[†]Industrial and Systems Engineering, Hiroshima University, Kagamiyama, Higashi-Hiroshima, 739 8527 JAPAN
[‡]Yamamoto Electric Corporation, 116 Wadamichi, Sukagawa, Fukushima, 962 0818 JAPAN

1 Introduction

The optimal regulator is usually designed based on a mathematical model of a controlled system. The exact mathematical model, however, is not always known in practical applications. Although some robust optimal regulators have been proposed [1, 2, 3] for the controlled systems with linear uncertainties, these robust optimal regulators cannot work well if nonlinear uncertainties exist.

For this problem, various regulators using neural networks have been exploited in recent years [4, 5, 6, 7, 8, 9]. Yamada and Yabuta [4] proposed a method based on the direct use of a neural network as a feedback controller, and discussed the stability of linear discrete-time single-input single-output controlled plants [5]. Although this type of controller is very simple and can be applied to various feedback control systems, the uncertainty included in the controlled plant cannot be identified and parameters of the neural network such as a learning rate and initial values of the weights are quite difficult to set.

On the other hand, Takahashi [6] used an adaptive neural identifier and a direct neural feedback controller for controlling a flexible arm. The neural identifier estimates the unknown parameters of the arm and the neural controller works on the basis of these identified parameters. Polycarpou and Helmicki [7] presented a learning approach for automated fault detection and accommodation. Their system can detect faults of a plant and adjust the plant behavior using multiple neural networks. Rovithakis and Christodoulou [8] used three neural networks in order to realize an adaptive regulator. Iiguni et al. [9] designed a nonlinear regulator for a controlled system with nonlinear modeling errors. The method uses two neural networks for identification of the controlled object and for modification of a feedback control input according to the identified model. In these proposed methods, the unknown part of the controlled system is identified by one neural network, and other neural networks are used as a kind of compensator. Since the multiple neural networks must be trained, it requires a long time for computation and learning, and stability analysis becomes difficult.

Generally, the controlled system includes a known part and an unknown part, and the unknown part is composed of linear and/or nonlinear uncertainties. Therefore, it is necessary to establish an efficient regulator design utilizing such information about the controlled system.

In this chapter, the Neuro-Based Adaptive Regulator (NBAR) for a class of dynamic systems with linear and nonlinear uncertainties is proposed. The NBAR includes not only an optimal regulator designed based on the linear known part but also a neural network to identify the unknown part included in the controlled system. The neural network also works as an adaptive compensator for the unknown part of the controlled system. First, we show how the neural network's output compensates the control input based on the Riccati equation, and how the compensatory solution of the Riccati equation is estimated by using the least-squares method. Then, in order to illustrate the effectiveness of the NBAR, it is applied to a control problem of a double cart-spring system with linear and nonlinear uncertainties.

2 Neuro-Based Adaptive Regulator

2.1 System Formulation

As a controlled system, we consider the following system that consists of linear and nonlinear parts:

$$\begin{aligned} \dot{x}(t) &= A_L x(t) + \tilde{A}(x(t)) + Bu(t), & (1)\\ y(t) &= Cx(t), & (2) \end{aligned}$$

where $x(t) \in \Re^{n\times 1}$, $u(t) \in \Re^{m\times 1}$ and $y(t) \in \Re^{l\times 1}$ are the state, the input and the output variables, respectively; $A_L \in \Re^{n\times n}$, $B \in \Re^{n\times m}$, $C \in \Re^{l\times n}$ are the parameter matrices; and $\tilde{A}(x(t))$ is the nonlinear function of the state $x(t)$.

The goal of the control is to find the optimal input that minimizes the quadratic performance index of the form

$$J = \int_0^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt, \tag{3}$$

where $R(\in \Re^{m\times m}) > 0$, $Q(\in \Re^{n\times n}) \geq 0$ are the weight matrices specified by the designer. The nonlinear regulator problem for the system (1) is very difficult to solve, and one of the linearized techniques is generally utilized in order to synthesize an observer and a linear optimal regulator. However, serious control problems may arise when (1) cannot be linearized appropriately, so that nonlinear compensation is more strongly required. Therefore we propose an adaptive regulator using a single neural network with capabilities of nonlinear mapping, learning ability and parallel computation in this chapter.

2.2 Quadratic Optimal Regulator for Linearized System

First, we divide the matrix A_L of (1) into a known parameter matrix $A_{Ln} \in \Re^{n\times n}$ and an uncertain matrix $\Delta_{A_L} \in \Re^{n\times n}$, that is

$$A_L = A_{L_n} + \Delta_{A_L}, \tag{4}$$

and assume that the nonlinear function $\tilde{A}(x(t))$ is approximately described as

$$\tilde{A}(x(t)) \approx A^* x(t) + \Delta_{A^*} x(t) \tag{5}$$

near the operating point of the controlled system. The matrix $A^* \in \Re^{n\times n}$ represents the linearized parameter and $\Delta_{A^*} \in \Re^{n\times n}$ represents the unknown linearized modeling error.

Then the system (1) becomes

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), & (6)\\ A &= A_n + \Delta_A, & (7)\\ A_n &= A_{L_n} + A^*, & (8)\\ \Delta_A &= \Delta_{A_L} + \Delta_{A^*}, & (9) \end{aligned}$$

where $\Delta_A \in \Re^{n\times n}$ and $A_n \in \Re^{n\times n}$ denote the system uncertainty and the nominal parameter, respectively. The system (6) is assumed to be controllable and observable.

The optimal control input $u^*(t)$ that minimizes the performance index J of (3) is given as

$$u^*(t) = -R^{-1}B^T Px(t), \tag{10}$$

where $P \in \Re^{n\times n}$ is the unique solution of the following Riccati equation [10]:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0. \tag{11}$$

In this chapter, the solution P is split into two components:

$$P = P_n + \Delta_P, \tag{12}$$

where $P_n \in \Re^{n\times n}$ and $\Delta_P \in \Re^{n\times n}$ are the solution for the known linear part of the system (6) and the compensatory solution of the Riccati equation, respectively.

Substituting (7), (12) into (11), we have

$$\begin{aligned}(P_n + \Delta_P)(A_n + \Delta_A) + (A_n + \Delta_A)^T(P_n + \Delta_P)\\ -(P_n + \Delta_P)BR^{-1}B^T(P_n + \Delta_P) + Q = 0.\end{aligned} \tag{13}$$

If the quadratic terms $\Delta_P\Delta_A$, $\Delta_A^T\Delta_P$, $\Delta_P\Delta_P$ are sufficiently small, (13) can be divided into the following two equations:

$$P_nA_n + A_n^T P_n - P_nBR^{-1}B^T P_n + Q = 0, \tag{14}$$

$$\Delta_P A_n + P_n\Delta_A + \Delta_A^T P_n + A_n^T\Delta_P - \Delta_P BR^{-1}B^T P_n - P_nBR^{-1}B^T\Delta_P = 0. \tag{15}$$

In order to compute the optimal control input $u^*(t)$ of (10), the solution P of the Riccati equation (11) has to be known. Although P_n can be obtained by solving the Riccati equation (14), the compensatory solution Δ_P must be computed from the matrix equation (15).

2.3 Derivation of Compensatory Input

In this subsection, we use the least-squares estimation for the compensatory solution Δ_P of the Riccati equation (15). From (15), we can have the following form:

$$\begin{aligned}\Delta_P\mathcal{D} + \mathcal{F}\Delta_P &= \Delta_A^T P_n + P_n\Delta_A, & (16)\\ \mathcal{D} &= BR^{-1}B^T P_n - A_n, & (17)\\ \mathcal{F} &= P_nBR^{-1}B^T - A_n^T. & (18)\end{aligned}$$

Deriving the quadratic form of the state $x(t)$ for both sides of (16), we have

$$\begin{aligned} x^T(t)\mathcal{K}x(t) &= \Delta_x^T(t)P_n x(t) + x^T(t)P_n\Delta_x(t), & (19)\\ \Delta_x(t) &= \Delta_A x(t), & (20)\\ \mathcal{K} &= \Delta_P\mathcal{D} + \mathcal{F}\Delta_P, & (21)\end{aligned}$$

where $\Delta_x(t)$ is the uncertain state caused by the uncertainty Δ_A.

Developing (19) into a linear equation and arranging the linear equation corresponding to the unknown compensatory solution Δ_P, we can obtain

$$\omega(t)\theta = \lambda(t), \tag{22}$$

$$\theta(t) = [\vartheta_{11},\cdots,\vartheta_{1n},\cdots,\vartheta_{n1},\cdots,\vartheta_{nn}]^T \in \Re^{n^2\times 1}, \tag{23}$$

$$\omega(t) = [\omega_{11}(t),\cdots,\omega_{1n}(t),\cdots,\omega_{n1}(t),\cdots,\omega_{nn}(t)] \in \Re^{1\times n^2}, \tag{24}$$

$$\omega_{kh}(t) = \sum_{i=1}^{n} x_i(t)d_{hi}x_k(t) + \sum_{j=1}^{n} x_h(t)f_{jk}x_j(t) \ \ (k,h=1,2,\cdots,n), \tag{25}$$

$$\lambda(t) = \sum_{j=1}^{n}\sum_{i=1}^{n}\Delta_{x_j}(t)p_{ji}x_i(t) + \sum_{j=1}^{n}\sum_{i=1}^{n}x_j(t)p_{ji}\Delta_{x_i}(t), \tag{26}$$

where p_{ij}, d_{hi}, f_{jk} are the elements of the known matrices P_n, $\mathcal{D}$, $\mathcal{F}$, respectively; ϑ_{ij} is the element of the unknown vector $\theta(t)$; and $\theta(t) = \mathrm{cs}\Delta_P$ is the expanded form of the column of the matrix Δ_P [11].

When we can get $s(\geq n^2)$ sets of the sequential data with the sampling time Δt, the following matrix equation is obtained from (22):

$$\Omega(t)\theta(t) = \Lambda(t), \tag{27}$$

where

$$\Omega(t) = \begin{bmatrix} \omega(t-\Delta t)\\ \vdots\\ \omega(t-l\Delta t)\\ \vdots\\ \omega(t-s\Delta t)\end{bmatrix} \in \Re^{s\times n^2}, \tag{28}$$

$$\Lambda(t) = \begin{bmatrix} \lambda(t-\Delta t)\\ \vdots\\ \lambda(t-l\Delta t)\\ \vdots\\ \lambda(t-s\Delta t)\end{bmatrix} \in \Re^{s\times 1} \tag{29}$$

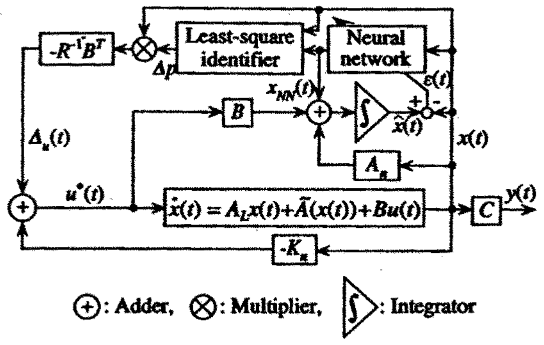

Figure 1: Block diagram of the neuro-based adaptive regulator

If $\lambda(t)$ of (26) is computed, then $\theta(t)$ can be estimated by the least-squares method, that is,

$$\theta(t) = \Omega^+(t)\Lambda(t), \tag{30}$$

where $\Omega^+(t)$ represents the pseudo inverse matrix. For the real time control, $\theta(t)$ can be generally estimated in on-line procedure with the sequential least-squares algorithm [12].

Also it should be noted that, for the compensatory solution Δ_P of the Riccati equation, the following Lemma exists which can be used to reduce the number of estimated parameters of the compensatory solution Δ_P.

Lemma 1 *If the weights R, Q are chosen as positive definite matrices, the compensatory solution Δ_P of the Riccati equation (12) becomes a real and symmetric matrix (Proof: see Appendix).*

Consequently, when $\lambda(t)$ of (26) is obtained, the compensatory solution Δ_P can be estimated. However, $\Delta_x(t)$ included in (26) cannot be computed from (20) since Δ_A is unknown. Therefore the neural network is introduced to solve this problem.

2.4 Neuro-Based Adaptive Regulator

Figure 1 shows the block diagram of the NBAR. The identification system shown in Fig. 1 is described as

$$\hat{\dot{x}}(t) = A_n x(t) + Bu(t) + x_{NN}(t), \tag{31}$$

where $\widehat{x}(t) \in \Re^{n\times 1}$ and $x_{NN}(t) \in \Re^{n\times 1}$ are the predicted state variables of the identification system and the output of the neural network, respectively.

Substituting (12) into (10), we have the optimal control input $u^*(t)$ as

$$\begin{aligned} u^*(t) &= u_n(t) + \Delta_u(t), && (32)\\ u_n(t) &= -K_n x(t), && (33)\\ K_n &= R^{-1}B^T P_n, && (34)\\ \Delta_u(t) &= -R^{-1}B^T\Delta_P x(t), && (35) \end{aligned}$$

where $K_n \in \Re^{m\times n}$ is the feedback gain of the linear optimal regulator. The identified state error $\epsilon(t) \in \Re^{n\times 1}$ between $\widehat{x}(t)$ and $x(t)$ is defined as

$$\begin{aligned} \epsilon(t) &= \widehat{x}(t) - x(t), && (36)\\ &= \int_0^t [\dot{\widehat{x}}(\tau) - \dot{x}(\tau)]d\tau \\ &= \int_0^t \zeta(\tau)d\tau, && (37)\\ \zeta(\tau) &= [x_{NN}(\tau) - \Delta_x(\tau)], && (38) \end{aligned}$$

where $\zeta(\tau) \in \Re^{n\times 1}$ represents the error between the neural network output and the uncertain state. However, even if the identified error $\epsilon(t)$ becomes zero, the integrand term $\zeta(\tau)$ of (37) may not be zero. So, we define the energy function $E(t)$ for training of the neural network as

$$\begin{aligned} E(t) &= \frac{1}{2}\dot{\epsilon}^T(t)\dot{\epsilon}(t) + \frac{1}{2}\epsilon^T(t)\epsilon(t) \\ &= \frac{1}{2}\zeta^T(t)\zeta(t) + \frac{1}{2}\epsilon^T(t)\epsilon(t) \\ &= E^{(1)}(t) + E^{(2)}(t), && (39)\\ E^{(1)}(t) &= \frac{1}{2}\zeta^T(t)\zeta(t), && (40)\\ E^{(2)}(t) &= \frac{1}{2}\epsilon^T(t)\epsilon(t). && (41) \end{aligned}$$

When $E(t)$ becomes zero, the output $x_{NN}(t)$ of the neural network agrees with the state uncertainties $\Delta_x(t)$, that is

$$x_{NN}(t) = \Delta_x(t). \qquad (42)$$

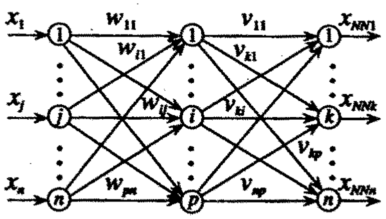

Figure 2: Three-layer neural network

According to the learning of the neural network, we can expect that the control input shown in Fig. 1 will gradually approach the optimal control input $u^*(t)$ of (10).

3 Multi-layer Neural Network and Learning

A multi-layer neural network used in the proposed regulator is shown in Fig. 2. The number of the units of the input layer, the hidden layer and the output layer are n, p and n, respectively. In Fig. 2, w_{ij} represents the weight that connects the unit j of the input layer to the unit i of the hidden layer; v_{ki} represents the weight that connects the unit i of the hidden layer to the unit k of the output layer. The weight matrices are represented as $W(t) \in \Re^{p\times n}$ and $V(t) \in \Re^{n\times p}$, respectively. Also, the input and output vectors of the neural network are represented as $x(t)$ and $x_{NN}(t)$, respectively.

Let the unit j's output of the input layer $(j = 1, \cdots, n)$ be

$$I_j = x_j(t), \tag{43}$$

the unit i's output of the hidden layer $(i = 1, \cdots, p)$ be

$$\begin{aligned} H_i &= \sigma(s_i), & (44) \\ s_i &= \sum\nolimits_{j=1}^{n} w_{ij} I_j, & (45) \end{aligned}$$

where the sigmoid function $\sigma(\cdot)$ is defined as

$$\sigma(\mu) \equiv \frac{1}{\gamma} \tanh(\gamma\mu). \tag{46}$$

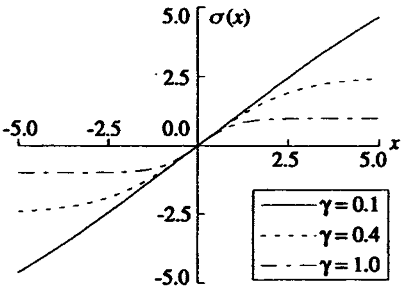

Figure 3: Sigmoid function used in the neural network

The parameter γ is positive, and determines the shape of the sigmoid function. Figure 3 shows the input-output relation of the sigmoid function. When $\gamma < 0.1$, the sigmoid function $\sigma(\mu)$ can be approximated by the linear function, and when $\gamma \geq 1$, $\sigma(\mu)$ takes the form of the tanh function. Also, the unit k's output of the output layer $(k = 1, \cdots, n)$ is given as

$$\begin{aligned} O_k &= \sigma(\kappa_k), && (47) \\ \kappa_k &= \sum_{i=1}^{p} v_{ki} H_i. && (48) \end{aligned}$$

In the training process, the energy function of (39) is minimized by changing the weights w_{ij} and v_{ki}. According to the back-propagation algorithm [13], the weight updating rules can be described as

$$\begin{aligned} v_{kj}(t+\Delta t) &= v_{kj}(t) - \eta \frac{\partial E(t)}{\partial v_{kj}(t)} \\ &= v_{kj}(t) - \eta \left[\frac{\partial E^{(1)}(t)}{\partial v_{kj}(t)} + \frac{\partial E^{(2)}(t)}{\partial v_{kj}(t)} \right], && (49) \\ w_{ij}(t+\Delta t) &= w_{ij}(t) - \eta \frac{\partial E(t)}{\partial w_{ij}(t)} \\ &= w_{ij}(t) - \eta \left[\frac{\partial E^{(1)}(t)}{\partial w_{ij}(t)} + \frac{\partial E^{(2)}(t)}{\partial w_{ij}(t)} \right], && (50) \end{aligned}$$

where $\eta > 0$ is the learning rate; and Δt is the time interval of the neural network learning. The function $E^{(1)}(t)$ of (39) for $\acute{e}(t) = \zeta(t)$ is rewritten as

$$E^{(1)}(t) = \frac{1}{2}\zeta^T(t)\zeta(t)$$

$$= \frac{1}{2}\sum_{q=1}^{n}[x_{NN_q}(t) - \Delta_{x_q}(t)]^2, \tag{51}$$

where $x_{NN_q}(t)$, $\Delta_{x_q}(t)$ are the elements of the neural network output $x_{NN}(t)$ and the uncertain state $\Delta_x(t)$, respectively.

By (37), (51), $\partial E^{(1)}(t)/\partial v_{ki}(t)$ becomes

$$\begin{aligned}\frac{\partial E^{(1)}(t)}{\partial v_{ki}(t)} &= \dot{\epsilon}_k(t)\frac{\partial x_{NN_k}(t)}{\partial v_{ki}(t)} \\ &= \dot{\epsilon}_k(t)\sigma'(\kappa_k)H_i,\end{aligned} \tag{52}$$

where $\dot{\epsilon}_k(t)$ represents the elements of the vector $\dot{\epsilon}(t)$, and $\sigma'(\cdot)$ denotes the derivative of $\sigma(\cdot)$.

Also, $\partial E^{(1)}(t)/\partial w_{ij}(t)$ can be written as

$$\begin{aligned}\frac{\partial E^{(1)}(t)}{\partial w_{ij}(t)} &= \sum_{q=1}^{n}\dot{\epsilon}_q(t)\frac{\partial x_{NN_q}(t)}{\partial w_{ij}(t)} \\ &= \sum_{q=1}^{n}\dot{\epsilon}_q(t)\sigma'(\kappa_q)\frac{\partial}{\partial w_{ij}(t)}[\sum_{i=1}^{p}v_{qi}H_i] \\ &= \sum_{q=1}^{n}\dot{\epsilon}_q(t)\sigma'(\kappa_q)v_{qi}\sigma'(s_i)\frac{\partial}{\partial w_{ij}(t)}[\sum_{j=1}^{n}w_{ij}I_j] \\ &= \sum_{q=1}^{n}\dot{\epsilon}_q(t)\sigma'(\kappa_q)v_{qi}\sigma'(s_i)x_j(t).\end{aligned} \tag{53}$$

On the other hand, the function $E^{(2)}(t)$ of (39) for the identified error $\epsilon(t)$ is given as

$$\begin{aligned}E^{(2)}(t) &= \frac{1}{2}\epsilon^T(t)\epsilon(t) \\ &= \frac{1}{2}\sum_{k=1}^{n}[\hat{x}_k(t) - x_k(t)]^2,\end{aligned} \tag{54}$$

and $\partial E^{(2)}(t)/\partial v_{ki}(t)$ is described by

$$\frac{\partial E^{(2)}(t)}{\partial v_{ki}(t)} = \epsilon_k(t)\frac{\partial \epsilon_k(t)}{\partial x_{NN_k}(t)}\frac{\partial x_{NN_k}(t)}{\partial v_{ki}(t)}, \tag{55}$$

where $\hat{x}_k(t)$ and $x_k(t)$ are the elements of the predicted state $\hat{x}(t)$ and the state $x(t)$, respectively; and $\epsilon_k(t)$ is the element of the vector $\epsilon(t)$. Then, the partial derivative $\partial\epsilon_k(t)/\partial x_{NN_k}(t)$ is approximated as

$$\frac{\partial\epsilon_k(t)}{\partial x_{NN_k}(t)} \approx \frac{\Delta\epsilon_k(t)}{\Delta x_{NN_k}(t)}. \tag{56}$$

The variation $\Delta\epsilon_k(t)$ of $\epsilon_k(t)$ can be approximately computed as

$$\begin{aligned} \Delta\epsilon_k(t) &\approx \{\sum_{j=0}^{N_t}[x_{NN_k}(j\Delta t_s) - \Delta_{x_k}(j\Delta t_s)] + \Delta x_{NN_k}(t)\}\Delta t_s \\ &\quad -\sum_{j=0}^{N_t}[x_{NN_k}(j\Delta t_s) - \Delta_{x_k}(j\Delta t_s)]\Delta t_s \\ &= \Delta x_{NN_k}(t)\Delta t_s, \end{aligned} \tag{57}$$

where Δt_s is the small sampling time and $t = N_t\Delta t_s$.

As a result, we can approximate $\partial\epsilon_k(t)/\partial x_{NN_k}(t)$ as

$$\frac{\partial\epsilon_k(t)}{\partial x_{NN_k}(t)} \approx \Delta t_s. \tag{58}$$

Substituting (58) into (55) yields

$$\frac{\partial E^{(2)}(t)}{\partial v_{ki}(t)} \approx \epsilon_k(t)\Delta t_s \frac{\partial x_{NN_k}(t)}{\partial v_{ki}(t)}. \tag{59}$$

In the same manner, $\partial E^{(2)}(t)/\partial w_{ij}(t)$ can be computed as

$$\frac{\partial E^{(2)}(t)}{\partial w_{ij}(t)} \approx \sum_{q=1}^{n} \epsilon_q(t)\Delta t_s \frac{\partial x_{NN_q}(t)}{\partial w_{ij}(t)}. \tag{60}$$

Consequently, the updating rules of (49), (50) are reduced to the following form

$$v_{ki}(t+\Delta t) \approx v_{ki}(t) - \eta[\dot{\epsilon}_k(t) + \epsilon_k(t)\Delta t_s]\frac{\partial x_{NN_k}(t)}{\partial v_{ki}(t)}, \tag{61}$$

$$w_{ij}(t+\Delta t) \approx w_{ij}(t) - \eta\sum_{q=1}^{n}[\dot{\epsilon}_q(t) + \epsilon_q(t)\Delta t_s]\frac{\partial x_{NN_q}(t)}{\partial w_{ij}(t)}. \tag{62}$$

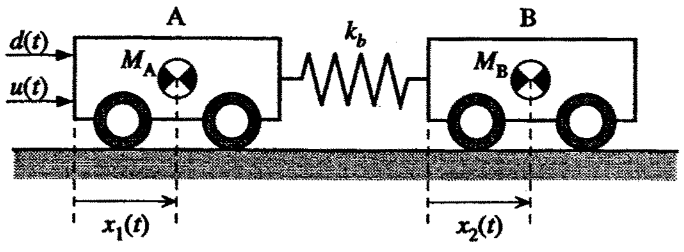

Figure 4: Double cart-spring system

Since (61) and (62) mean that the error signal for learning of the neural network is the weighted sum of $\epsilon(t)$ and $\dot{\epsilon}(t)$, this learning rule corresponds to the PD control of the feedback control system. Therefore, it is termed a PD updating rule in this chapter. It should be noted that, if the energy functions of (39) are modified as

$$E^{(1)}(t) = \frac{1}{2}\dot{\epsilon}^T(t)B_l\dot{\epsilon}(t), \tag{63}$$

$$E^{(2)}(t) = \frac{1}{2}\epsilon^T(t)K_l\epsilon(t), \tag{64}$$

the energy function $E(t)$ can be adjusted by the learning gains $B_l \in \Re^{n\times n}$, $K_l \in \Re^{n\times n}$ defined as the positive definite matrices.

4 Computer Simulation of Double Cart-Spring System

In this section, the NBAR is compared with the conventional Linear Quadratic Regulator (LQR) in order to illustrate the effectiveness of the NBAR. Let us consider a double cart-spring system [14] shown in Fig. 4, where $x_1(t)$, $x_2(t)$ are the displacements of the cart A and the cart B, respectively; $d(t)$ is the disturbance applied to the cart A; $u(t)$ is the control input, k_b is the spring constant; M_A and M_B are the masses of the cart A and the cart B, respectively. Ignoring a frictional force between the floor and the wheel of the cart, we have the following motion equations:

$$M_A\ddot{x}_1(t) + k_b[x_1(t) - x_2(t)] = u_1(t) + d(t), \tag{65}$$

$$M_B\ddot{x}_2(t) - k_b[x_1(t) - x_2(t)] \;=\; 0. \tag{66}$$

By choosing the cart displacements and their velocities as the state variables, we can get the state-space model of the double cart-spring system:

$$\dot{x}(t) = A_{Ln}x(t) + B[u(t) + d(t)], \tag{67}$$

where $x(t) = [x_1(t),\ \dot{x}_1(t),\ x_2(t),\ \dot{x}_2(t)]^T$, and

$$A_L = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -k_b/M_A & 0 & k_b/M_A & 0 \\ 0 & 1 & 0 & 0 \\ k_b/M_B & 0 & -k_b/M_B & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1/M_A \\ 0 \\ 0 \end{bmatrix}.$$

In this section, we use the estimated parameters of the system (67) as $\hat{k}_b = 1$ N/m, $\hat{M}_A = 1$ kg, $\hat{M}_B = 1$ kg. Then, the parameter matrix A_L used as the nominal parameter A_n is given as follows:

$$A_n = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}.$$

In the performance index of (3), the weight matrices are assigned as

$$Q = diag[0.5,\ 0.5,\ 0.5,\ 0.5],\quad R = 0.1,$$

so that we obtain the solution of the Riccati equation (14),

$$P_n = \begin{bmatrix} 1.6508 & 0.4215 & -0.4919 & 1.0599 \\ 0.4215 & 0.3665 & -0.1053 & 0.2605 \\ -0.4919 & -0.1053 & 1.3156 & 0.0893 \\ 1.0599 & 0.2605 & 0.0893 & 1.8504 \end{bmatrix},$$

and the feedback gain,

$$K_n = [4.2152,\ 3.6648,\ -1.0529,\ 2.6049].$$

The elements of the uncertainty Δ_A of (7) are chosen as uniform random numbers in $[-0.25,\ +0.25]$:

$$\Delta_A = \begin{bmatrix} 0 & 0.2173 & 0 & 0 \\ -0.2265 & 0 & -0.2233 & 0 \\ 0 & 0 & 0 & -0.2166 \\ 0.0896 & 0 & 0.0856 & 0 \end{bmatrix}. \tag{68}$$

In this case, the system can be rewritten as

$$\begin{aligned} \dot{x}(t) &= A_L x(t) + B[u(t) + d(t)], & (69) \\ A_L &= A_{Ln} + \Delta_A. & (70) \end{aligned}$$

As a result, the optimal solution P^* for the system (69) is given as

$$P^* = \begin{bmatrix} 1.3744 & 0.3957 & -0.2327 & 0.9343 \\ 0.3957 & 0.3825 & -0.0643 & 0.2841 \\ -0.2327 & -0.0643 & 1.2580 & 0.1962 \\ 0.9343 & 0.2841 & 0.1962 & 1.5189 \end{bmatrix},$$

and the optimal feedback gain K^* is given as

$$K^* = [3.9566,\ 3.8253,\ -0.6429,\ 2.8415].$$

Corresponding to the structure of this double cart-spring system, four units in the input layer, twenty units in the hidden layer and four units in the output layer are used in the three-layer neural network. The initial value of the weight is set as a uniform random number in [-0.6, +0.6], the learning rate is $\eta = 0.025$, the parameter of the sigmoid function is $\gamma = 1$ and the sampling time is $\Delta t = \Delta t_s = 0.1$ s. By Lemma 1, the compensatory solution Δ_P of the Riccati equation is a symmetric matrix with 10 unknown elements and we choose $s = 12$ of (27) in order to keep the rank of the pseudo inverse matrix $\Omega^+(t)$ more than 10. Using the disturbance $d(t)$ of a rectangular signal with the amplitude 20 N and the period $T = 12$ s, the following computer simulations are performed.

4.1 Control Performance

Figure 5 shows the simulation results under the NBAR (shown as the solid line) and the optimal feedback gain K^* (shown as the dashed line: LQR*), where the displacement $x_1(t)$ of the cart A is shown. The responses for short periods at the beginning of the learning and 10 minutes later are shown in Fig. 5 (a) and (b), respectively. Although the large error between two methods is observed in Fig. 5 (a), the result of the NBAR is approaching the optimal solution using the LQR* according to the neural network learning as shown in Fig. 5 (b). Thus, we can see that the NBAR can adaptively realize the optimal control for the system (69) with the uncertainty Δ_A.

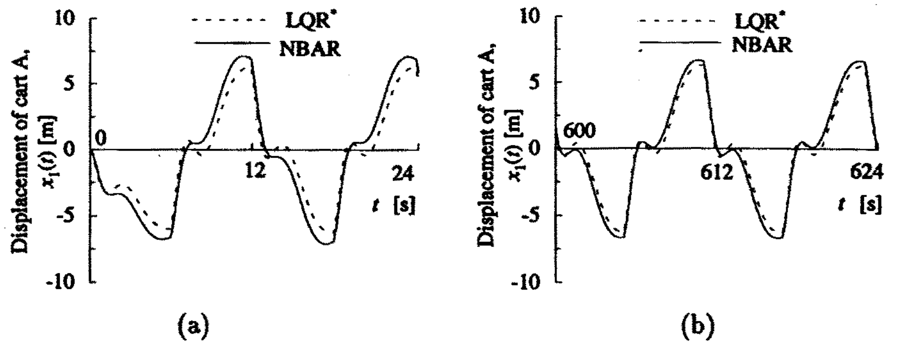

Figure 5: Comparison of the simulation results using the NBAR and the LQR*

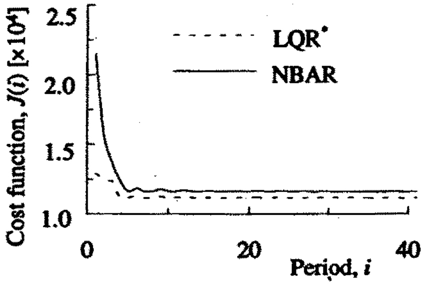

Figure 6: Comparison of the cost functions of LQR* and NBAR

Then, the performance index J of (3) per one period of the disturbance $d(t)$ is computed:

$$J(i) = \int_{(i-1)T}^{iT} [x^T(t)Qx(t) + u^T(t)Ru(t)]dt, \tag{71}$$

where $T = 12$ s.

Figure 6 shows the change of the performance index $J(i)$. It can be seen that the performance index of the NBAR is almost the same as the index value of the LQR* after learning.

Moreover, the matrix norm of the estimation error between the estimated value $\hat{\Delta}_P$ of (30) by the least-square method and the true value Δ_P^* of the

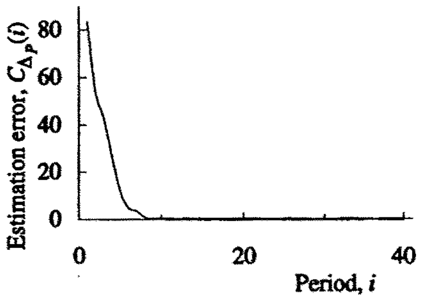

Figure 7: Estimation error of Δ_P

compensatory solution of the Riccati equation is defined as

$$C_{\Delta_P} = \frac{1}{T}\int_{(i-1)T}^{iT} ||\Delta_P^* - \hat{\Delta}_P(t)||dt, \tag{72}$$

and is shown in Fig. 7. In Fig. 6, 7, even if sufficient learning time has passed, a small error between the NBAR and the LQR* exists. This small error may be caused by ignoring the quadratic terms of the uncertainty Δ_A and the compensatory solution Δ_P for deriving (14), (15), and this must be considered as a problem to be dealt with in the future.

4.2 Ability of Learning and Identification

The control input $u(t)$ of the NBAR approaches the optimal control input $u^*(t)$ according to the learning of the uncertainties of the controlled system using the neural network. In order to check the learning ability of the neural network, we compute the energy function $E(i)$ of (39) per one period of the disturbance $d(t)$, which is defined as

$$E(i) = \frac{1}{T}\int_{(i-1)T}^{iT} E(t)dt. \tag{73}$$

Figure 8 indicates the change of the energy function $E(i)$. The energy function $E(i)$ gradually decreases through the neural network learning.

The control performance using the neural network generally depends on the initial values of the neural network's weights. So, we check the control

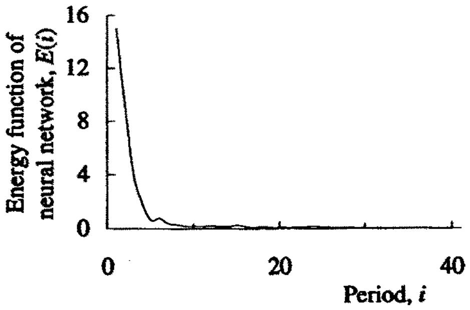

Figure 8: Energy function of the neural network learning

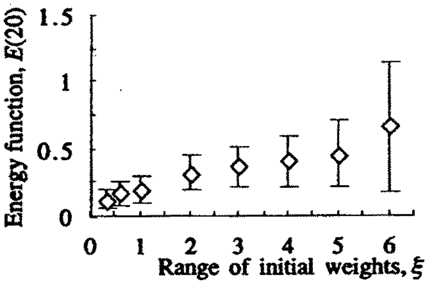

Figure 9: Change of the control performance with various ranges of initial weights of the neural network

performance of the NBAR by varying the initial values of the neural network's weights.

Figure 9 shows the relationship between the range $[-\xi,\ \xi]$ of the uniform random number and the mean square error $E(20)$. Note that the errors are shown by the mean values and their standard deviations for 10 different initial values of the neural network's weights. From Fig. 9, we may reasonably conclude that, as the range of the random number becomes large, the mean square error and its standard deviation increase. During the initial stage of learning, the neural network's output works as an undesirable disturbance to the control system. Therefore, when small initial values are used, the learning

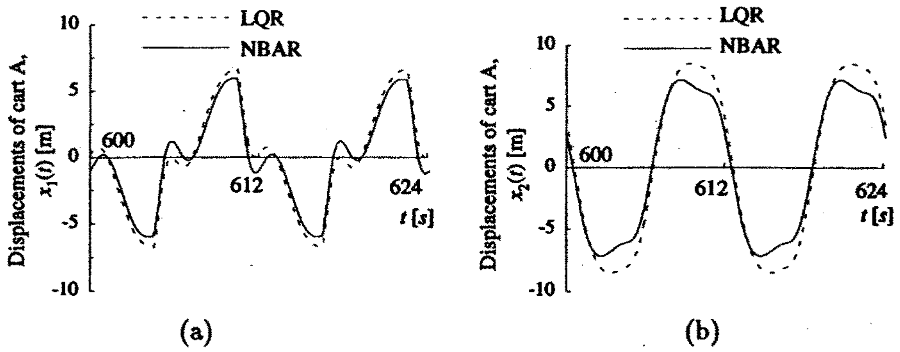

Figure 10: Simulation results for the system including the nonlinear uncertainty

of the neural network can be performed stably, and this gives the small mean square error shown in Fig. 9.

4.3 Nonlinear Uncertainties

Next, to investigate the effectiveness of the NBAR for nonlinear uncertainties included in the controlled system, we add nonlinear elements into the spring property of the motion equations (65), (66) and set the nonlinear function $\tilde{A}(x(t))$ of (1) to be

$$\tilde{A}(x(t)) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\psi x_1^2(t)/M_A & 0 & \psi x_2^2(t)/M_A & 0 \\ 0 & 0 & 0 & 0 \\ \psi x_1^2(t)/M_B & 0 & -\psi x_2^2(t)/M_B & 0 \end{bmatrix},$$

where ψ is the index of the nonlinearity and is set as $\psi = 0.015$.

Figures 10 and 11 are the simulation results using the NBAR (shown as the solid line) and the feedback gain K_n (shown as the dashed line: LQR) designed for (67), where the displacements of the two carts and the values of the performance index (71) are shown in Fig. 10 and 11, respectively. We can see from these figures that even if nonlinear uncertainties exist in the controlled system, the NBAR gives a smaller value of the performance index than the LQR.

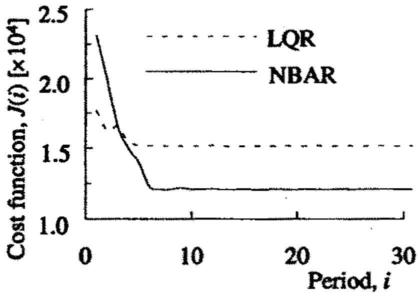

Figure 11: Cost functions for the controlled system including the nonlinear uncertainty

5 Conclusion

In this chapter, the Neuro-Based Adaptive Regulator has been proposed for a class of systems with linear and nonlinear uncertainties. It has been shown that the neural network can be used for identifying the uncertainties and adaptively compensating the control input, simultaneously. The NBAR computes the compensatory control input from the neural network's output based on the least-square algorithm, and it is effective for a controlled system with linear and nonlinear uncertainties.

From a practical point of view, the learning speed largely influences the applicability of the proposed methods. However, this chapter used the standard neural network model and did not consider speeding up the learning. For improving the learning speed as quickly as possible, two problems have to be solved in the future. The first one is the learning rate that should be studied in connection with the stability of the control system. The second is the weight's initial value for the neural network. During the initial stage of the learning, the neural network's output works as an undesirable disturbance to the control system, so that the weight's initial value has a large effect on the stability and the learning speed of the control system. It is necessary to establish a design method for the initial value, which should be stable and have as a small effect as possible on the control system.

Appendix

Proof of Lemma 1. If the weights R, Q are chosen as the positive definite matrices, we can see that the solution P of the Riccati equation (11) and the solution P_n of the Riccati equation (14) are the real symmetric matrices [15]. From (12), Δ_P is given by

$$\Delta_P = P - P_n, \tag{74}$$

then we can get the real symmetric matrix Δ_P.

References

[1] A. Weinmann, *Uncertain Models and Robust Control*, Springer-Verlag, New York, 1991.

[2] J. Douglas and M. Athans, "Robust linear quadratic design with real parameter uncertainty," *IEEE Transactions On Automatic Control*, vol. 39, no. 1, pp. 107–111, 1994.

[3] M. Green and D. J. N. Limebeer, *Linear Robust Control*, Prentice Hall, Englewood Cliffs, 1995.

[4] T. Yamada and T. Yabuta, "Nonlinear neural network controller for dynamic systems," in *Proceedings of 16th Annual Conference of IEEE Industrial Electronics Society*, 1990, pp. 1244–1249.

[5] T. Yamada and T. Yabuta, "Some remarks on characteristics of direct neuro-controller with regard to adaptive control," *Transactions On the Society of Instrument and Control Engineers*, vol. 27, no. 7, pp. 784–791, 1991.

[6] K. Takahashi, "Neural-network-based learning control applied to a single-link flexible arm," in *Proceedings of Second International Conference on Motion and Vibration Control*, 1994, pp. 811–816.

[7] M. M. Polycarpou and A. J. Helmicki, "Automated fault detection and accommodation:a learning systems approach," *IEEE Transactions On Systems, Man, and Cybernetics*, vol. 25, pp. 1447–1458, 1995.

[8] G. A. Rovithakis and M. A. Christodoulou, "Direct adaptive regulation of unknown nonlinear dynamical systems via dynamic neural networks," *IEEE Transactions On Systems, Man, and Cybernetics*, vol. 25, pp. 1578–1594, 1995.

[9] H. Sakai Y. Iiguni and H. Tokumaru, "A nonlinear regulator design in the presence of system uncertainties using multilayered neural network," *IEEE Transactions On Neural Networks*, vol. 2, no. 4, pp. 410–417, 1991.

[10] T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, 1980.

[11] B. P. Molinari, "The time-invariant linear-quadratic optimal control problem," *Automatica*, vol. 13, pp. 347–357, 1977.

[12] M. Jamshidi and M. Malek-zavarei, *Linear Control Systems: a Computer-Aided Approach*, Pergamon Press, 1986.

[13] G. E. Hinton D. E. Rumelhart and R. J. Williams, "Learning representations by error propagation," in *Parallel Distributed Processing*, J. L. McClelland D. E. Rumelhart and PDP Research Group, Eds., vol. 1. MIT Press, 1986.

[14] R. Setola A. Cavallo and F. Vasca, *Using MATLAB, SIMULINK and Control System Toolbox*, Prentice Hall, Englewood Cliffs, 1996.

[15] S. Kodama and N. Suda, *Matrix Theory for System Control*, The Society of Instrument and Control Engineers of Japan, Tokyo, 1978.