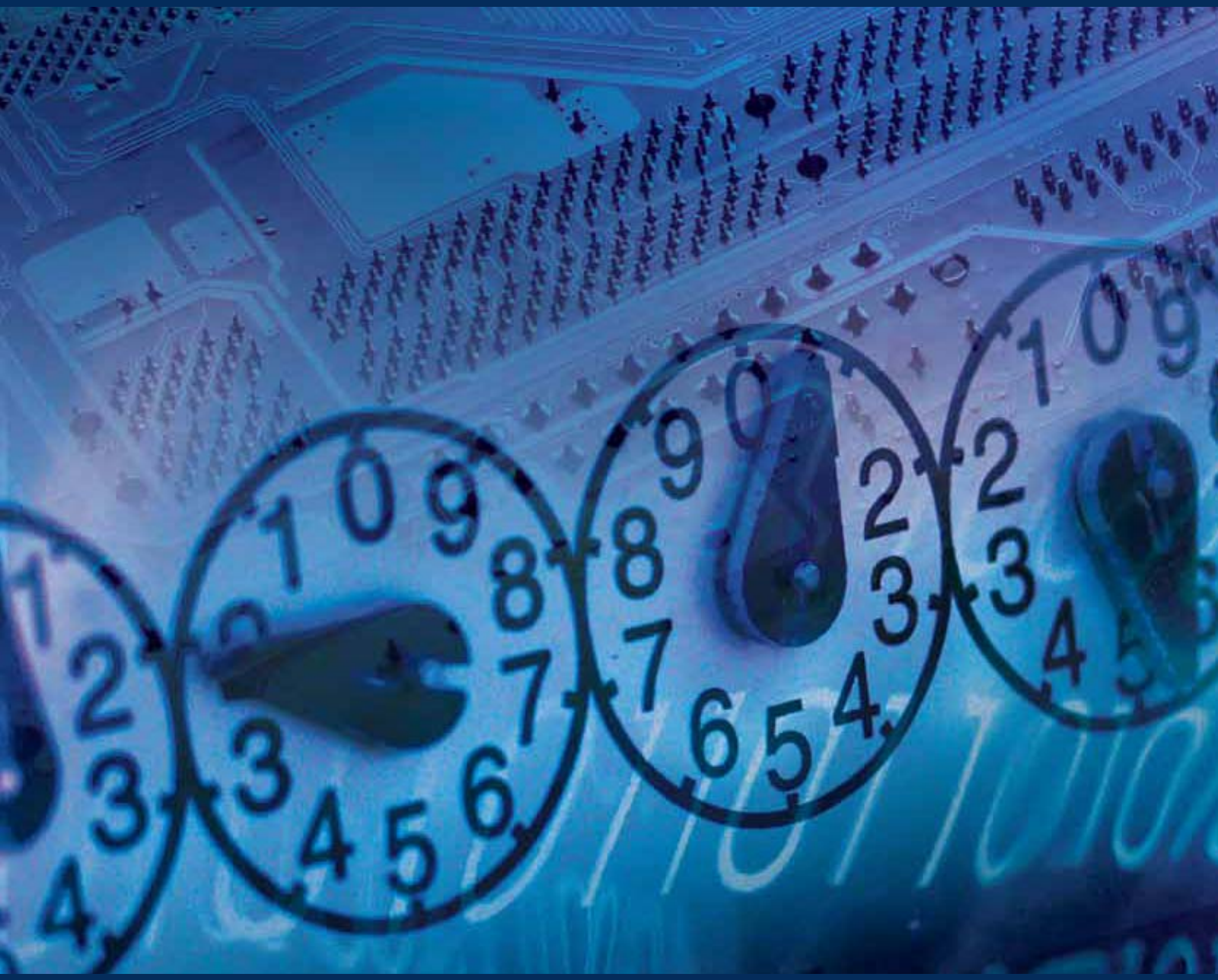
Journal of Control Science and Engineering

Dynamic Neural Networks for Model-Free Control and Identification

Guest Editors: Alex Poznyak, Isaac Chairez, Haibo He, and Wen Yu



Dynamic Neural Networks for Model-Free Control and Identification

Journal of Control Science and Engineering

Dynamic Neural Networks for Model-Free Control and Identification

Guest Editors: Alex Poznyak, Isaac Chairez, Haibo He, and Wen Yu

Copyright © 2012 Hindawi Publishing Corporation. All rights reserved.

This is a special issue published in "Journal of Control Science and Engineering." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board

Edwin K. P. Chong, USA
Ricardo Dunia, USA
Nicola Elia, USA
Peilin Fu, USA
Bijoy K. Ghosh, USA
F. Gordillo, Spain
Shinji Hara, Japan
Seul Jung, Republic of Korea
Vladimir Kharitonov, Russia
James Lam, Hong Kong
Derong Liu, China
Tomas McKelvey, Sweden
Silviu-Iulian Niculescu, France
Yoshito Ohta, Japan
Yang Shi, Canada
Zoltan Szabo, Hungary
Onur Toker, Turkey
Xiaofan Wang, China
Jianliang Wang, Singapore
Wen Yu, Mexico
Mohamed A. Zribi, Kuwait

Contents

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 916340, 2 pages
doi:10.1155/2012/916340

Editorial

Dynamic Neural Networks for Model-Free Control and Identification

Alex Poznyak,[1] Isaac Chairez,[2] Haibo He,[3] and Wen Yu[1]

[1] *Departamento de Control Automático, Centro de Investigación y de Estudios Avanzados, 07360 Mexico City, DF, Mexico*
[2] *UPIBI, Biotechnology Department, National Polytecnical Institute (IPN), 07738 Mexico City, DF, Mexico*
[3] *Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA*

Correspondence should be addressed to Alex Poznyak, apoznyak@ctrl.cinvestav.mx

Received 17 December 2012; Accepted 17 December 2012

Copyright © 2012 Alex Poznyak et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Neural networks have been used to solve a broad diversity of problems on different scientific and technological disciplines. Particularly, control and identification of uncertain systems have received attention since many years ago by the natural interest to solve problem such as automatic regulation or tracking of systems having a high degree of vagueness on their formal mathematical description. On the other hand, artificial modeling of uncertain systems (where the pair output-input is the only available information) has been exploited by many years with remarkable results.

Within automatic control and identification theory, neural networks must be designed using a dynamic structure. Therefore, the so-called dynamic neural network scheme has emerged as a relevant and interesting field. Dynamic neural networks have used recurrent and differential forms to represent the uncertainties of nonlinear models. This couple of representations has permitted to use the well-developed mathematical machinery of control theory within the neural network framework.

The purpose of this special issue is to give an insight on novel results regarding neural networks having either recurrent or differential models. This issue has encouraged application of such type of neural networks on adaptive control designs or/and no parametric modeling of uncertain systems.

The contributions of this issue reflect the well-known fact that neural networks traditionally cover a broad variety of the thoroughness of techniques deployed for new analysis and learning methods of neural networks. Based on the recommendation of the guest editors, a number of authors were invited to submit their most recent and unpublished contributions on the aforementioned topics. Finally, five papers were accepted for publication.

So, the paper of P. K. Kim and S. Jung titled "*Experimental studies of neural network control for one-wheel mobile robot*" presents development and control of a disc-typed one-wheel mobile robot, called GYROBO. Several models of the one-wheel mobile robot are designed, developed, and controlled. The current version of GYROBO is successfully balanced and controlled to follow the straight line. GYROBO has three actuators to balance and move. Two actuators are used for balancing control by virtue of gyroeffect and one actuator for driving movements. Since the space is limited and weight balance is an important factor for the successful balancing control, careful mechanical design is considered. To compensate for uncertainties in robot dynamics, a neural network is added to the nonmodel-based PD-controlled system. The reference compensation technique (RCT) is used for the neural network controller to help GYROBO to improve balancing and tracking performances. The paper of R. C. Rodríguez and W. Yu "*Robust adaptive control via neural linearization and compensation*" proposes a new type of neural adaptive control via dynamic neural networks. For a class of unknown nonlinear systems, a neural identifier-based feedback linearization controller is first used. Dead-zone and projection techniques are applied to assure the stability of neural identification. Then four types of compensator are addressed. The stability of closed-loop system is also proven. "*Dynamics model abstraction scheme using radial basis functions*" is presented in the paper of S. Tolu et al. where a control model for object manipulation is presented. System dynamics depend on an unobserved

external context, for example, work load of a robot manipulator. The dynamics of a robot arm change as it manipulates objects with different physical properties, for example, the mass, shape, or mass distribution. Active sensing strategies to acquire object dynamical models with a radial basis function neural network (RBF) are addressed. The paper "*An output-recurrent-neural-network-based iterative learning control for unknown nonlinear dynamic plants*" presented by Y.-C. Wang and C.-J. Chien deals with a design method for iterative learning control system by using an output recurrent neural network (ORNN). Two ORNNs are employed to design the learning control structure. The first ORNN, which is called the output recurrent neural controller (ORNC), is used as an iterative learning controller to achieve the learning control objective. To guarantee the convergence of learning error, some information of plant sensitivity is required to design a suitable adaptive law for the ORNC. Therefore, a second ORNN, which is called the output recurrent neural identifier (ORNI), is used as an identifier to provide the required information. The problems related to "*3D nonparametric neural identification*" are presented in the paper of R. Q. Fuentes et al., there, the state identification study of 3D partial differential equations (PDEs) using the differential neural networks (DNNs) approximation is given. The adaptive laws for weights ensure the "practical stability" of the DNN trajectories to the parabolic three-dimensional (3D) PDE states. To verify the qualitative behavior of the suggested methodology, a nonparametric modeling problem for a distributed parameter plant is analyzed.

These papers were exploring dissimilar applications of neural networks in control and identification from very different point of view. Despite the number of papers, the spirit of neural networks as a model-independent tool has been emphasized. Moreover, the number of practical examples included in the papers of this issue gives an additional contribution to the theory of dynamic neural network.

Acknowledgments

The editors wish to thank the editorial board for providing the opportunity to edit this special issue on modeling and adaptive control with dynamic neural networks. The guest editors wish also to thank the referees who have critically evaluated the papers within the short stipulated time. Finally we hope the reader will share our joy and find this special issue very useful.

Alex Poznyak
Isaac Chairez
Haibo He
Wen Yu

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 194397, 12 pages
doi:10.1155/2012/194397

Research Article

Experimental Studies of Neural Network Control for One-Wheel Mobile Robot

P. K. Kim and S. Jung

Intelligent Systems and Emotional Engineering (I.S.E.E.) Laboratory, Department of Mechatronics Engineering, Chungnam National University, Daejeon 305-764, Republic of Korea

Correspondence should be addressed to S. Jung, jungs@cnu.ac.kr

Received 18 July 2011; Revised 28 January 2012; Accepted 14 February 2012

Academic Editor: Haibo He

Copyright © 2012 P. K. Kim and S. Jung. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents development and control of a disc-typed one-wheel mobile robot, called GYROBO. Several models of the one-wheel mobile robot are designed, developed, and controlled. The current version of GYROBO is successfully balanced and controlled to follow the straight line. GYROBO has three actuators to balance and move. Two actuators are used for balancing control by virtue of gyro effect and one actuator for driving movements. Since the space is limited and weight balance is an important factor for the successful balancing control, careful mechanical design is considered. To compensate for uncertainties in robot dynamics, a neural network is added to the nonmodel-based PD-controlled system. The reference compensation technique (RCT) is used for the neural network controller to help GYROBO to improve balancing and tracking performances. Experimental studies of a self-balancing task and a line tracking task are conducted to demonstrate the control performances of GYROBO.

1. Introduction

Mobile robots are considered as quite a useful robot system for conducting conveying objects, conducting surveillance, and carrying objects to the desired destination. Service robots must have the mobility to serve human beings in many aspects. Most of mobile robots have a two-actuated wheel structure with three- or four-point contact on the ground to maintain stable pose on the plane.

Recently, the balancing mechanism becomes an important issue in the mobile robot research. Evolving from the inverted pendulum system, the mobile-inverted pendulum system (MIPS) has a combined structure of two systems: an inverted pendulum system and a mobile robot system. Relying on the balancing mechanism of the MIPS, a personal transportation vehicle has been introduced [1]. The MIPS has two-point contact to stabilize itself. Advantages of the MIPS include robust balancing against small obstacles on the ground and possible narrow turns while three- or four-point contact mobile robots are not able to do so.

In research on balancing robots, two-point contact mobile robots are designed and controlled [1–5]. A series of balancing robots has been implemented and demonstrated their performances [3–5]. Currently, successful navigation and balancing control performances with carrying a human operator as a transportation vehicle have been reported [5].

More challengingly, one-wheel mobile robots are developed. A one-wheel robot is a rolling disk that requires balancing while running on the ground [6]. Gyrover is a typical disc-typed mobile robot that has been developed and presented for many years [7–11]. Gyrover is a gyroscopically stabilized system that uses the gyroscopic motion to balance the lean angle against falling as shown in Figure 1. Three actuators are required to make Gyrover stable. Two actuators are used for balancing and one actuator for driving. Experimental results as well as dynamical modeling analysis on Gyrover are well summarized in the literature [12].

In other researches on one-wheel robots, different approaches of modelling dynamics of a one-wheel robot have been presented [13, 14]. Simulation studies of controlling

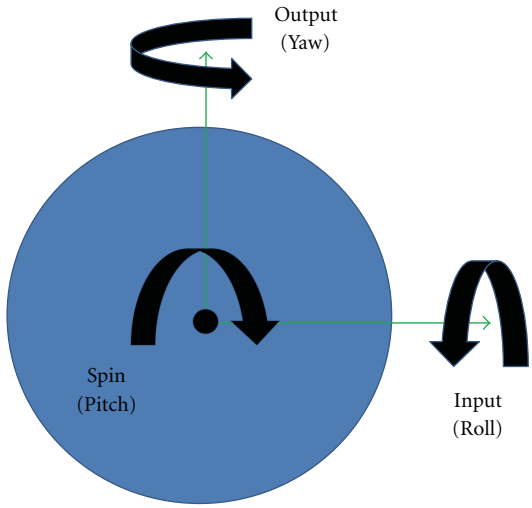


Figure 1: Gyro motion of one-wheel robot.

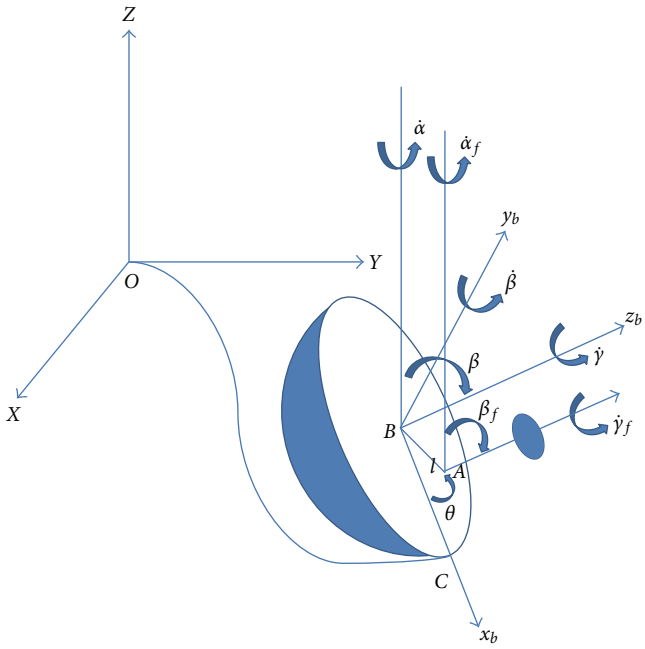


Figure 2: GYROBO Coordinates.

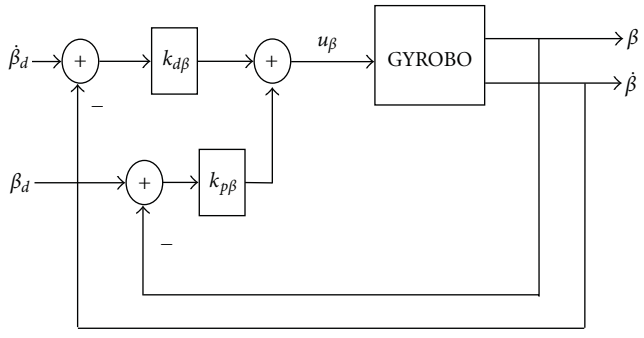


Figure 3: Lean angle control for balancing.

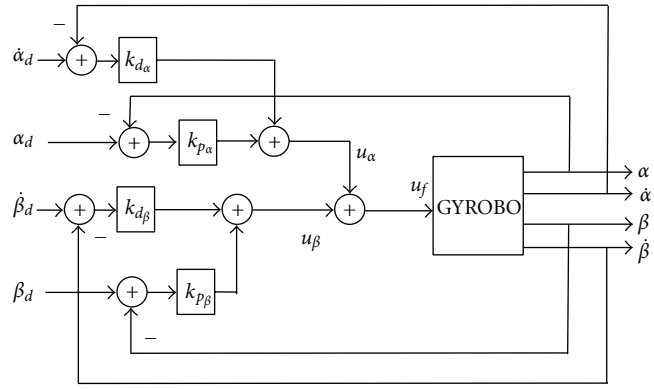


Figure 4: Straight line tracking control structure.

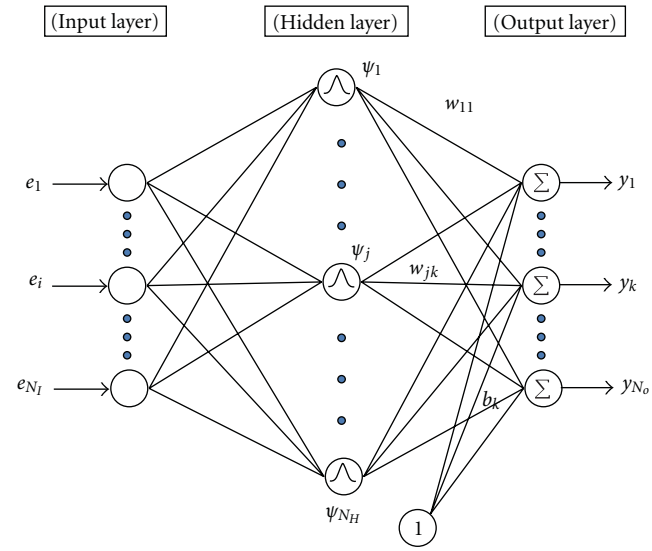


Figure 5: Neural network structure.

a Gyrobot along with design and fabrication are presented [15, 16]. An interesting design of a single spherical-wheel robot has been presented [17–19]. Successful balancing control of the spherical-wheel robot has been demonstrated.

In this paper, a one-wheel robot called GYROBO is designed, implemented, and controlled. GYROBO has three actuators to move forward and backward and to balance itself. Although aforementioned research results provide dynamics models for the one-wheel robot, in real physical system, it is quite difficult to control GYROBO based on dynamic models due to several reasons. First, modeling the system is not accurate. Second, it is hard to describe coupling effects between the body wheel and the flywheel since the system is nonlinearly coupled. Third, it is a nonholonomic system.

Therefore, we focus on optimal mechanical design rather than modeling the system since the dynamic model is quite complicated. Although we have dynamic models, other effects from uncertainties and nonmodel dynamics should be considered as well for the better performance.

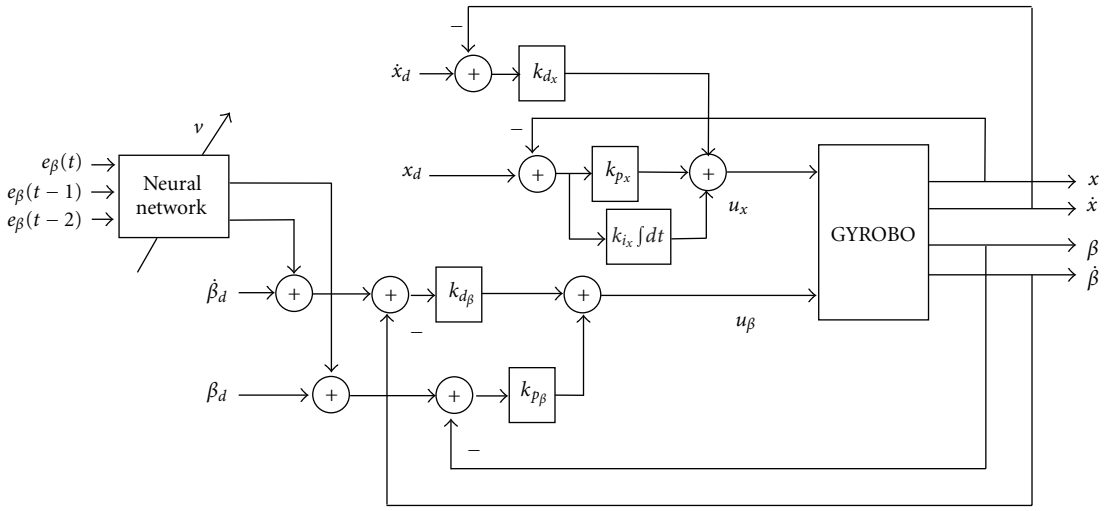


Figure 6: RCT neural network control structure.

Figure 7: Sphere robot with links.

After several modifications of the design, the successful design and control of GYROBO are achieved. Since all of actuators have to be housed inside the one wheel, design and placement of each part become the most difficult problem.

After careful design of the system, a neural network control scheme is applied to the model-free-controlled system to improve balancing performance achieved by linear controllers. First, a simple PD control method is applied to the system and then a neural network controller is added to help the PD controller to improve tracking performance. Neural network has been known for its capabilities of learning and adaptation for model-free dynamical systems in online fashion [2, 20].

Experimental studies of a self-balancing task and a straight line tracking task are conducted. Performances by a PD controller and a neural controller are compared.

2. GYROBO Modelling

GYROBO is described as shown in Figure 2. Three angles such as spin, lean, and tilt angle are generated by three corresponding actuators. The gyro effect is created by the combination of two angle rotations, flywheel spin and tilt motion.

Table 1: Parameter definition.

α, α_f	Precession angle of a wheel and a flywheel
β	Lean angle of a wheel
β_f	Tilt angle of a flywheel
γ, γ_f	Spin angles of a wheel and a flywheel
θ	Angle between l and x_b axis
m_w, m_f	Masses of a wheel and a flywheel
m	Mass of whole body
R	Radius of a wheel
I_{xw}, I_{yw}, I_{zw}	Wheel moment of inertia about X, Y, and Z axes
I_{xf}, I_{yf}, I_{zf}	Flywheel moment of inertia about X, Y, and Z
g	Gravitational velocity
u_1, u_2	Drive torque and tilt torque
l	Distance between A and B

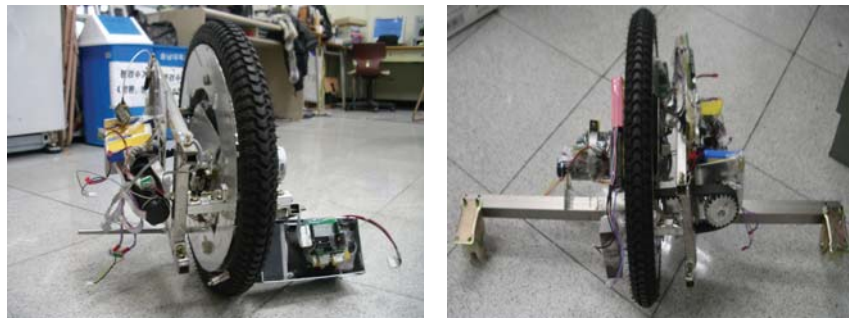
The dynamics of one-wheel robot system has been presented in [7, 11, 13]. Table 1 lists parameters of GYROBO. This paper is focused on the implementation and control of GYROBO rather than analyzing the dynamics model of the system since the modeling has been well presented in the literature [11].

Since the GYROBO is a nonholonomic system, there are kinematic constraints such that the robot cannot move in the lateral direction. GYROBO is an underactuated system that has three actuators to drive more states.

We follow the dynamic equation of the GYROBO described in [11]:

$$M(q)\ddot{q} + F(q,\dot{q}) = A^T\lambda + Bu, \tag{1}$$

(a) First model (b) Second model



(c) Third model (d) Fourth model

Figure 8: Models of GYROBO using gyro effects.

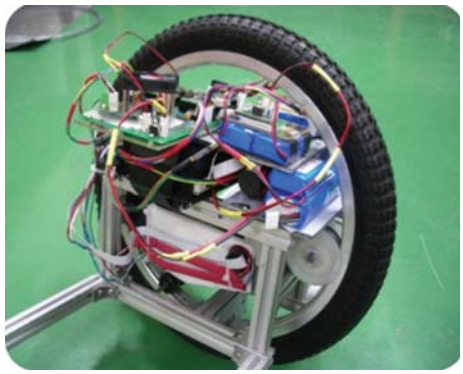


Figure 9: Real design of GYROBO I.

where

$$M(q) = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & M_{33} & 0 & I_{zw}S\beta & 0 \\ 0 & 0 & 0 & I_{xw} + I_{xf} + mR^2S^2\beta & 0 & I_{xf} \\ 0 & 0 & I_{zw}S\beta & 0 & I_{zw} & 0 \\ 0 & 0 & 0 & I_{xf} & 0 & I_{xf} \end{bmatrix},$$

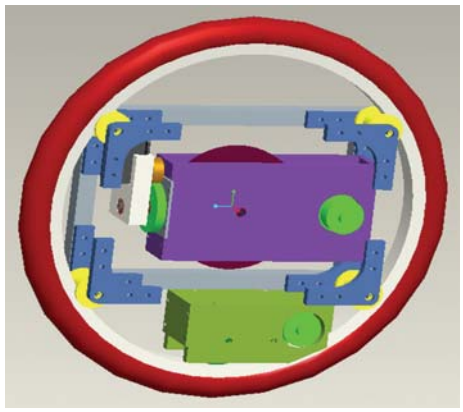
$$M_{33} = I_{yw}C^2\beta + I_{zw}S^2\beta + I_{yf}C^2\left(\beta + \beta_f\right) + I_{zf}S^2\left(\beta + \beta_f\right),$$

$$F(q,\dot{q}) = \begin{bmatrix} 0 \\ 0 \\ F_3 \\ F_4 \\ I_{zw}C\beta\dot{\alpha}\dot{\beta} \\ F_6 \end{bmatrix}, \tag{2}$$

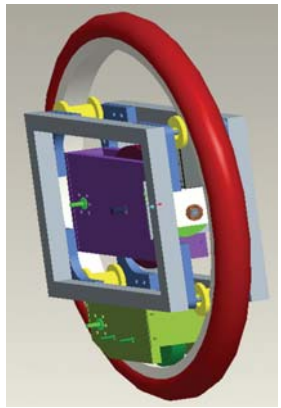
where

$$\begin{aligned} F_3 &= 2\left(I_{zw} - I_{yw}\right)C\beta S\beta\dot{\alpha}\dot{\beta} + I_{zw}C\beta\dot{\beta}\dot{\gamma} \\ &\quad + 2\left(I_{zf} - I_{yf}\right)C\left(\beta + \beta_f\right)S\left(\beta + \beta_f\right)\left(\dot{\beta} + \dot{\beta}_f\right)\dot{\alpha} \\ &\quad + I_{zf}C\left(\beta + \beta_f\right)\left(\dot{\beta} + \dot{\beta}_f\right)\gamma_f, \\ F_4 &= mR^2S\beta C\beta\dot{\beta}^2 + \left(I_{yw} - I_{zw}\right)C\beta S\beta\dot{\alpha}^2 - I_{zw}C\beta\dot{\gamma}\dot{\alpha} \\ &\quad + \left(I_{yf} - I_{zf}\right)C\left(\beta + \beta_f\right)S\left(\beta + \beta_f\right)\dot{\alpha}^2 \\ &\quad - I_{zf}C\left(\beta + \beta_f\right)\dot{\gamma}_f\dot{\alpha} - mgRS\beta, \\ F_6 &= \left(I_{yf} - I_{zf}\right)C\left(\beta + \beta_f\right)S\left(\beta + \beta_f\right)\dot{\alpha}^2 - I_{zf}C\left(\beta + \beta_f\right)\dot{\gamma}_f\dot{\alpha}, \end{aligned} \tag{3}$$

$$A = \begin{bmatrix} 1 & 0 & -RC\alpha C\beta & RS\alpha S\beta & -RC\alpha & 0 \\ 0 & 1 & -RC\beta C\alpha & -RC\alpha C\beta & -RS\alpha & 0 \end{bmatrix}, \tag{4}$$



(a)



(b)

Figure 10: Design of GYROBO II.

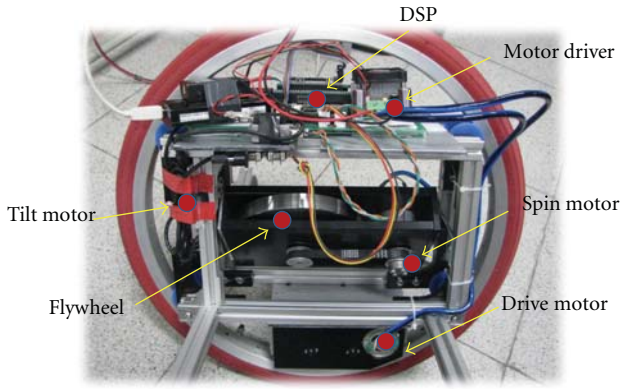


Figure 11: Real GYROBO II.

$$q = \begin{bmatrix} X \\ Y \\ \alpha \\ \beta \\ \gamma \\ \beta_f \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ k_1 & 0 \\ 0 & 1 \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{5}$$

where $C\alpha = \cos(\alpha)$, and $S\alpha = \sin(\alpha)$.

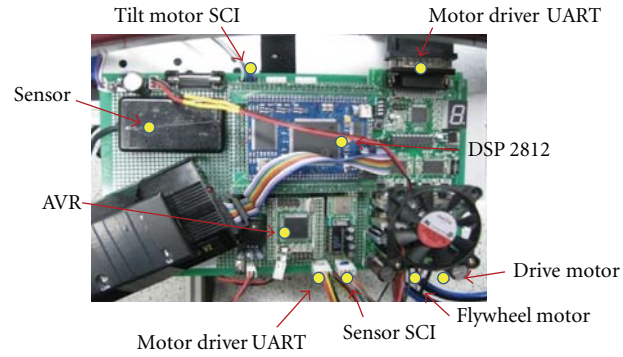


Figure 12: Control hardware.

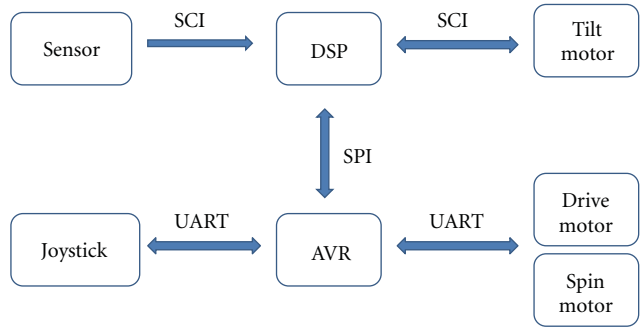


Figure 13: Control hardware block diagram.

3. Linear Control Schemes

The most important aspect of controlling GYROBO is the stabilization that prevents from falling in the lateral direction. Stabilization can be achieved by controlling the lean angle β. The lean angle in the y-axis can be separately controlled by controlling the flywheel while the flywheel rotates at a high constant speed. Spinning and tilting velocities of the flywheel induce the precession angle rate that enables GYROBO to stand up. Therefore, a linear controller is designed for the lean angle separately to generate suitable flywheel tilting motions.

3.1. PD Control for Balancing. The PD control method is used for balancing of GYROBO. The lean angle error is defined as

$$e_\beta = \beta_d - \beta, \tag{6}$$

where β_d is the desired lean angle value which is 0 for the balancing purpose and β is the actual lean angle. The angle error passes through the PD controller and generates the tilt torque u_β for the flywheel:

$$u_\beta = k_{p\beta} e_\beta + k_{d\beta} \dot{e}_\beta, \tag{7}$$

where $k_{p\beta}$ and $k_{d\beta}$ are controller gains. Figure 3 shows the PD control block diagram.

3.2. Straight Line Tracking Control. For the GYROBO to follow the straight line, a torque for the body wheel and a torque

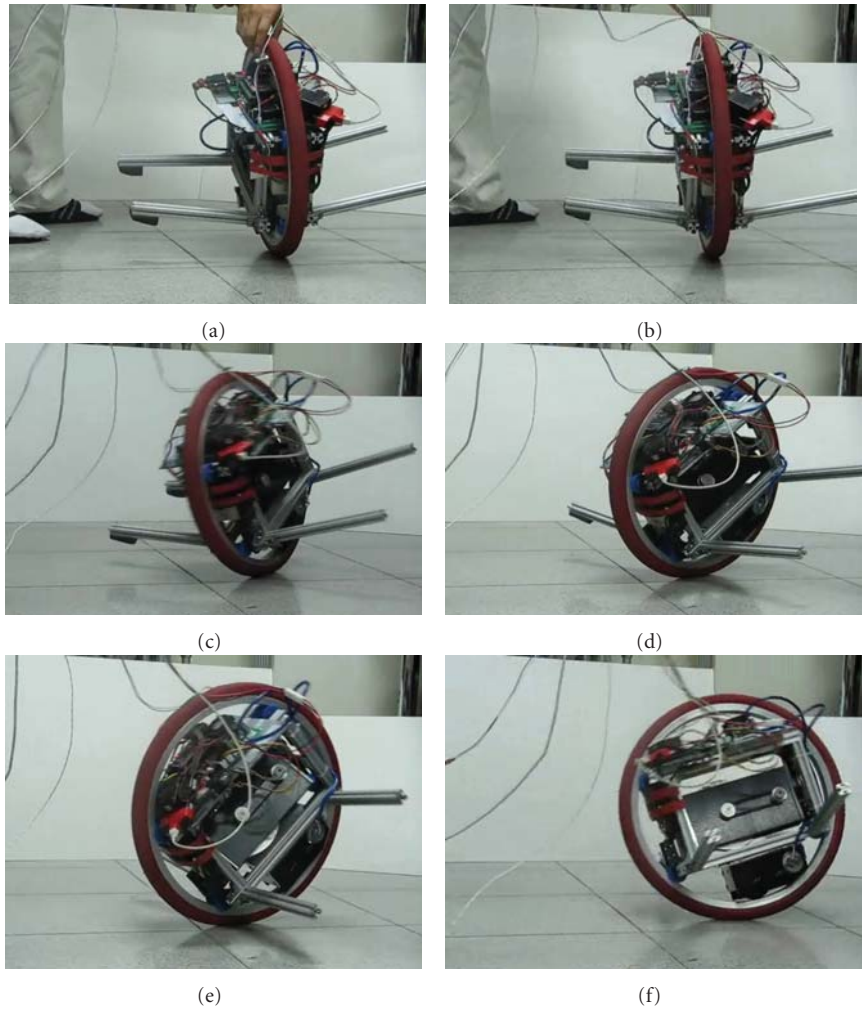


Figure 14: PD control: balancing task (a-b-c-d-e-f in order).

for the flywheel should be separately controlled. The body wheel rotates to follow the straight line while the lean angle is controlled to maintain balancing.

The position error detected by an encoder is defined by

$$e_p = x_d - x, \tag{8}$$

where x_d is the desired position value and x is the actual position. The detailed PID controller output becomes

$$u_x = k_{px}e_x + k_{dx}\dot{e}_x + k_{ix}\int e_x dt, \tag{9}$$

where k_{px}, k_{dx}, and k_{ix} are controller gains.

Thus line tracking control requires both position control and angle control. The detailed control block diagram for the straight line following is shown in Figure 4.

4. Neural Control Schemes

4.1. RBF Neural Network Structure. The purpose of using a neural network is to improve the performance controlled by linear controllers. Linear controllers for controlling GYROBO may have limited performances since GYROBO is a highly nonlinear and coupled system.

Neural networks have been known for their capabilities of learning and adaptation of nonlinear functions and used for nonlinear system control [20]. Successful balancing control performances of a two-wheel mobile robot have been presented [2].

One of the advantages of using a neural network as an auxiliary controller is that the dynamic model of the system is not required. The neural network can take care of nonlinear uncertainties in the system by an iterative adaptation process of internal weights.

Here the radial basis function (RBF) network is used for an auxiliary controller to compensate for uncertainties caused by nonlinear dynamics of GYROBO. Figure 5 shows the structure of the radial basis function.

The Gaussian function used in the hidden layer is

$$\psi_j(e) = \exp\left(-\frac{\left|e-\mu_j\right|^2}{2\sigma_j^2}\right), \tag{10}$$

where e is the input vector, $e = [e_1 e_2 \cdots e_{N_I}]^T$, μ_j is

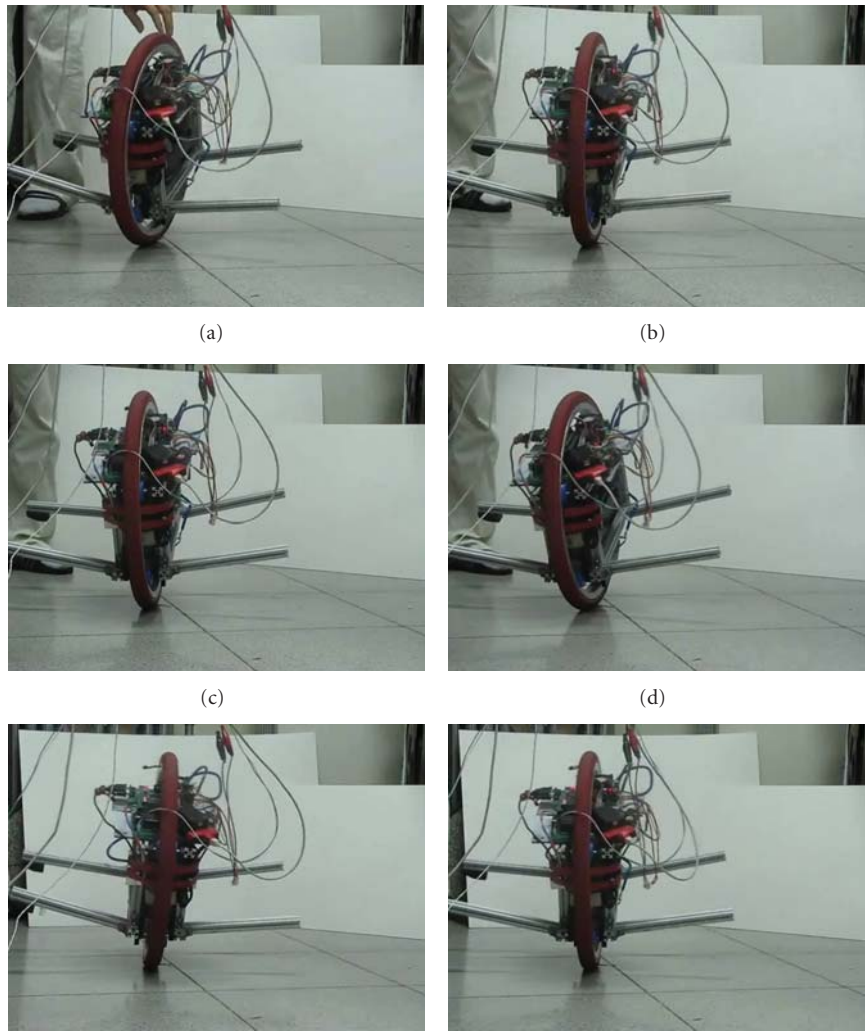


Figure 15: Neural network control: balancing task.

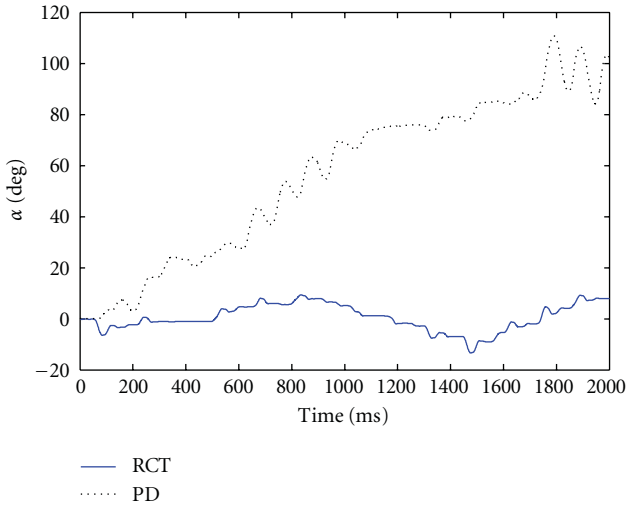


Figure 16: Heading angle comparison of PD (black dotted line) and RCT control (blue solid line).

the center value vector of the jth hidden unit, and σ_j is the width of the jth hidden unit.

The forward kth output in the output layer can be calculated as a sum of outputs from the hidden layer:

$$y_k = \sum_{j=1}^{N_H} \psi_j w_{jk} + b_k, \tag{11}$$

where ψ_j is jth output of the hidden layer in (11), w_{jk} is the weight between the jth hidden unit and kth output, and b_k is the bias weight.

4.2. Neural Network Control. Neural network is utilized to generate compensating signals to help linear controllers by minimizing the output errors. Initial stability can be achieved by linear controllers and further tracking improvement can be done by neural network in online fashion.

Different compensation location of neural network yields different neural network control schemes, but they eventually perform the same goal to minimize the output error.

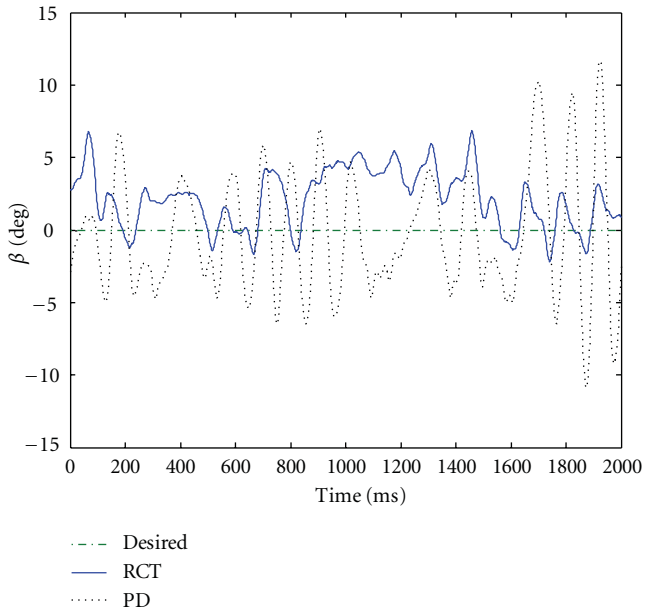


Figure 17: Lean angle comparison of PD (black dotted line) and RCT control (blue solid line).

The reference compensation technique (RCT) is known as one of neural network control schemes that provides a structural advantage of not interrupting the predesigned control structure [3, 4]. Since the output of neural network is added to the input trajectories of PD controllers as in Figure 4, tracking performance can be improved. This forms the totally separable control structure as an auxiliary controller shown in Figure 6:

$$u_\beta = k_{p\beta}\left(e_\beta + \phi_1\right) + k_{d\beta}\left(\dot{e}_\beta + \phi_2\right), \tag{12}$$

where ϕ_1 and ϕ_2 are neural network outputs.

Here, the training signal v is selected as a function of tracking errors such as a form of PD controller outputs:

$$v = k_{p\beta}e_\beta + k_{d\beta}\dot{e}_\beta. \tag{13}$$

Then (12) becomes

$$v = u_\beta - \left(k_{p\beta}\phi_1 + k_{d\beta}\phi_2\right). \tag{14}$$

To achieve the inverse dynamic control in (14) such as satisfying the relationship $u_\beta = \tau$ where τ is the dynamics of GYROBO, we need to drive the training signal of neural network to satisfy that $v \to 0$. Then the neural network outputs become equal to the dynamics of the GYROBO in the ideal condition. This is known as an inverse dynamics control scheme. Therefore learning algorithm is developed for the neural network to minimize output errors in the next section.

4.3. Neural Network Learning. Here neural network does not require any offline learning process. As an adaptive controller, internal weights in neural network are updated at each sampling time to minimize errors. Therefore, the selection of an appropriate training signal for the neural network becomes an important issue in the neural network control, even it determines the ultimate control performance.
The objective function is defined as

$$E = \frac{1}{2}v^2. \tag{15}$$

Differentiating (15) and combining with (14) yield the gradient function required in the back-propagation algorithm:

$$\frac{\partial E}{\partial w} = v\frac{\partial v}{\partial w} = -v\left(k_{p\beta}\frac{\partial \phi_1}{\partial w} + k_{d\beta}\frac{\partial \phi_2}{\partial w}\right). \tag{16}$$

The weights are updated as

$$w(t+1) = w(t) + \eta v\left(k_{p\beta}\frac{\partial \phi_1}{\partial w} + k_{d\beta}\frac{\partial \phi_2}{\partial w}\right), \tag{17}$$

where η is the learning rate.

5. Design of One-Wheel Robot

Design of the one-wheel robot becomes the most important issue due to the limitation of space. Our first model is shown in Figure 7. The original idea was to balance and control the sphere robot by differing rotating speeds of two links. Thus, two rotational links inside the sphere are supposed to balance itself but failed due to the irregular momentum induced by two links.

After that, the balancing concept has been changed to use a flywheel instead of links to generate gyro effect as shown in Figure 8. Controlling spin and tilt angles of the flywheel yields gyro effects to make the robot keep upright position.

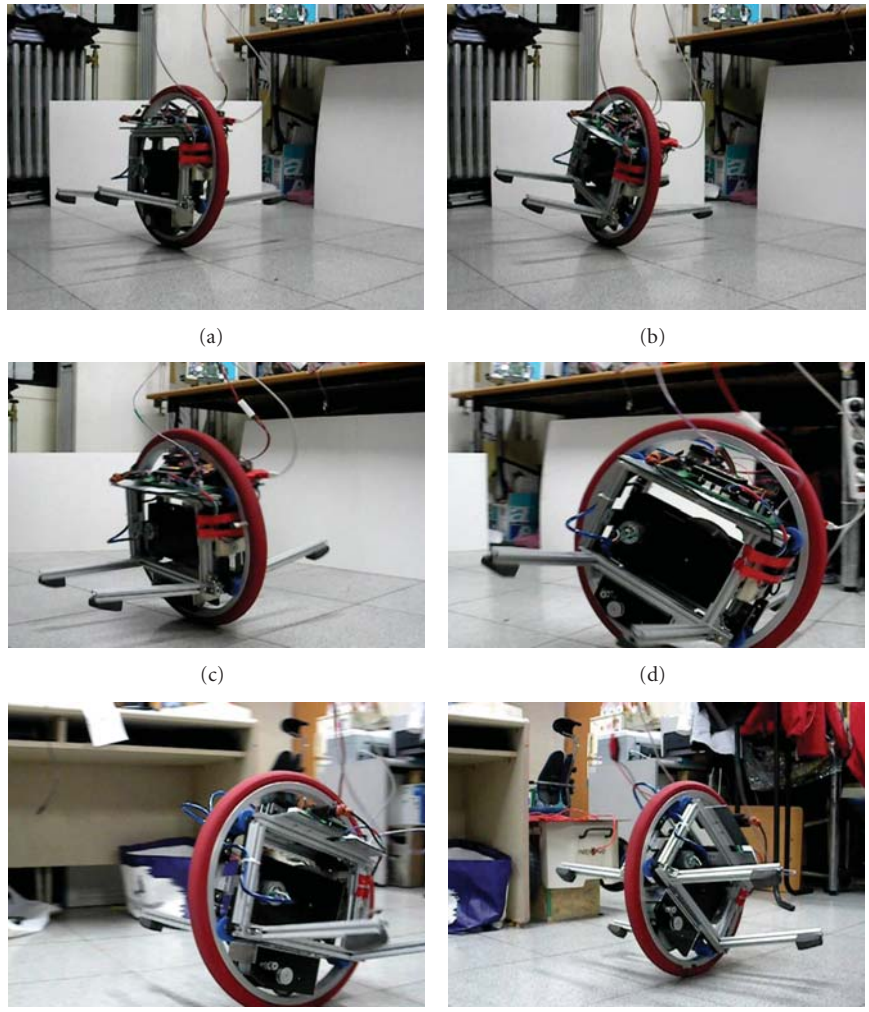
We have designed and built several models as shown in Figure 8. However, all of models are not able to balance itself long enough. Through trial and error processes of designing models, we have a current version of GYROBO I and II as shown in Figures 9 and 10. Since the GYROBO I in Figure 9 has a limited space for housing, the second model of GYROBO has been redesigned as shown in Figure 11.

6. GYROBO Design

Figure 10 shows the CAD design of the GYROBO II. The real design consists of motors, a flywheel, and necessary hardware as shown in Figure 11.

GYROBO II is redesigned with several criteria. Locating motors appropriately in the limited space becomes quite an important problem to satisfy the mass balance. Thus the size of the body wheel is increased.

The placement of a flywheel effects the location of the center of gravity as well. The size of the flywheel is also critical in the design to generate enough force to make the whole body upright position. The size of the flywheel is increased. The frame of the flywheel system is redesigned and located at the center of the body wheel.



(a) (b) (c) (d) (e) (f)

Figure 18: PD control: line following task.

The drive motor is attached to the wheel so that it directly controls the movement of the wheel. A tilt motor is mounted on the center line to change the precession angle of the gyro effect. A high-speed spin motor is located to rotate the flywheel through a timing belt.

Control hardware includes a DSP2812 and an AVR as main processors. The AVR controls drive and spin motors while DSP controls a tilt motor. The detailed layout is shown in Figure 12. To detect the lean angle, a gyro sensor is used. Interface between hardware is shown in Figure 13.

7. Experimental Studies

7.1. Balancing Test at One Point

7.1.1. Scheme 1: PD Control. The speed of the flywheel is set to 7,000 rpm. The control frequency is 10 ms. When the PD controller is used, the GYROBO is able to balance itself as shown in Figure 14. However, the heading angle also rotates as it balances. Figures 14(d), 14(e), and 14(f) show the deviation of the heading angle controlled by a PD controller whose gains are selected as $k_{p\beta} = 10$, and $k_{d\beta} = 15$. PD gains are selected by empirical studies.

7.1.2. Scheme 2: Neural Network Compensation. The same balancing test is conducted with the help of the neural network controller. The learning rate is set to 0.0005, and 3 inputs, 4 hidden units, and 2 outputs are used for the neural network structure. Neural network parameters are found by empirical studies. The GYROBO balances itself and does not rotate much comparing with Figure 14. It stays still as shown in Figure 15. The heading angle is not changed much when neural network control is applied while it deviates for PD control. The larger heading angle error means rotation of GYROBO while balancing which is not desired.

Further clear comparisons of performances between PD control and neural network control are shown in Figures 16 and 17, which are the corresponding plots of Figures 14 and 15. We clearly see that the heading angle of GYROBO is

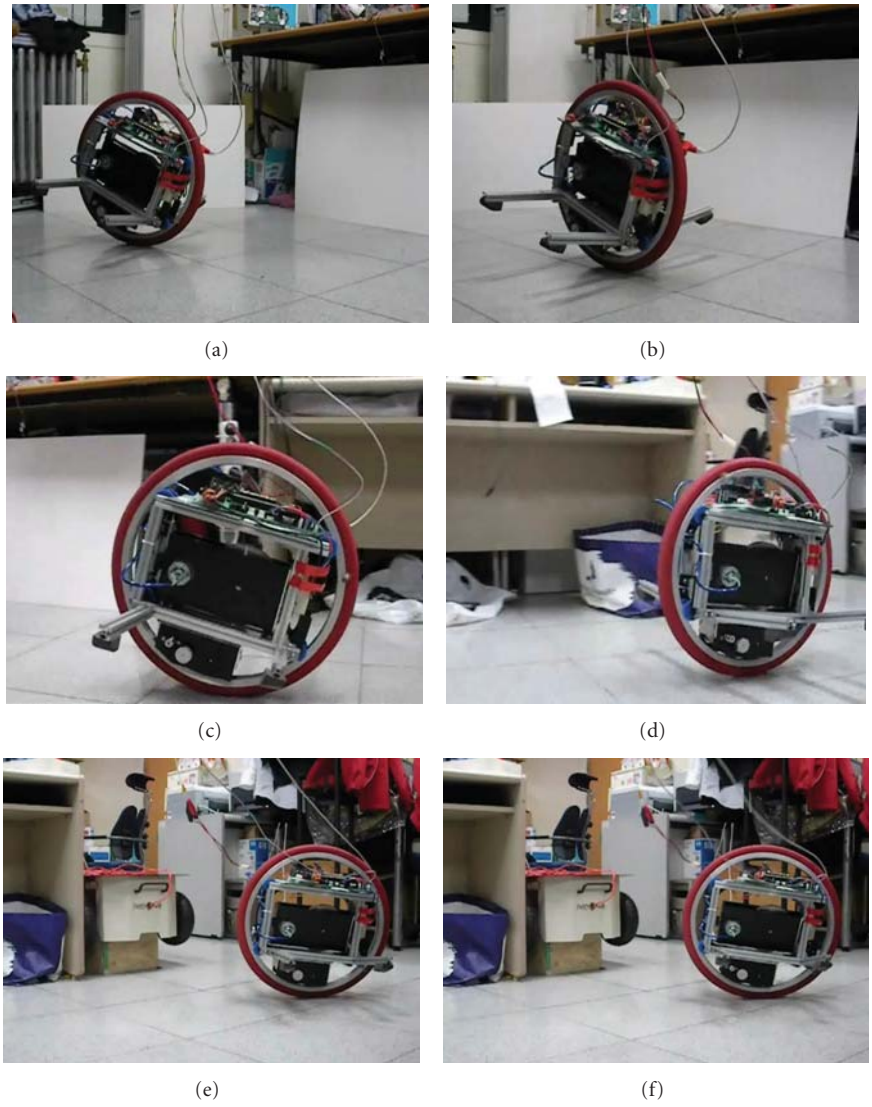


Figure 19: Neural network control: line following.

deviating in the PD control case. We clearly see the larger oscillation of PD control than that of RCT control.

7.2. Line Tracking Test

7.2.1. Scheme 1: PD Control. Next test is to follow the desired straight line. Figure 18 shows the line tracking performance by the PD controller. GYROBO moves forward about 2 m and then stops.

7.2.2. Scheme 2: Neural Network Compensation. Figure 19 shows the line tracking performance by the neural network controller.

We see that the GYROBO moves forward while balancing. Since the GYROBO has to carry a power line, navigation is stopped within a short distance about 2 m.

Clear distinctions between PD control and neural network control are shown in Figures 20 and 21. Both controllers maintain balance successfully. GYROBO controlled by a PD control method deviates from the desired straight line trajectory further as described in Figure 20. In addition, the oscillatory behaviour is reduced much by the neural network controller as shown in Figure 21.

8. Conclusion

A one-wheel robot GYROBO is designed and implemented. After several trial errors of body design, successful design is presented. An important key issue is the design to package all materials in one wheel. One important tip in controlling GYROBO is to reduce the weight so that the flywheel can generate enough gyro effect because it is not easy to find suitable motors. Although balancing and line tracking tasks are successful in this paper, one major problem has to be solved in the future. The power supply for the GYROBO should be independent. Since the stand-alone type of GYROBO is preferred, a battery should be mounted inside

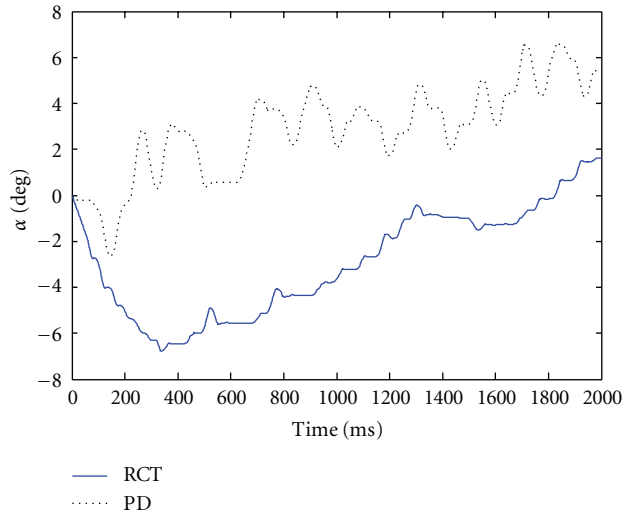


Figure 20: Heading angles of PD (black dotted line) and RCT control (blue solid line).

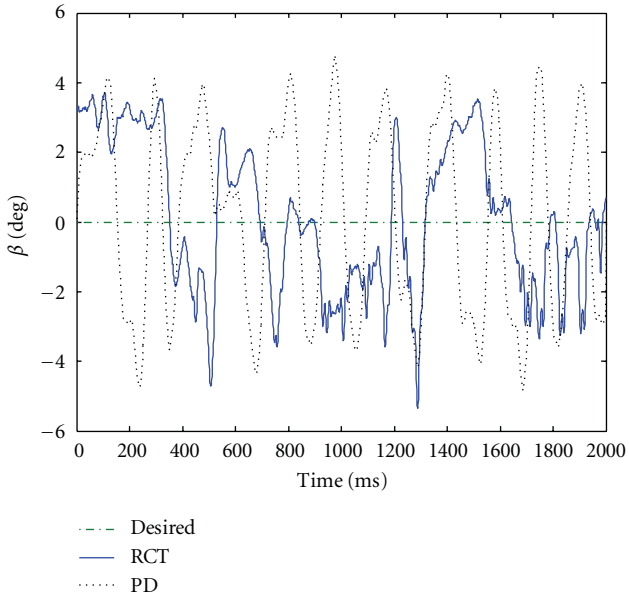


Figure 21: Lean angles of PD (black dotted line) and RCT control (blue solid line).

the wheel. Then a room for a battery requires modification of the body design again.

Acknowledgments

This work was financially supported by the basic research program (R01-2008-000-10992-0) of the Ministry of Education, Science and Technology (MEST) and by Center for Autonomous Intelligent Manipulation (AIM) under Human Resources Development Program for Convergence Robot Specialists (Ministry of Knowledge Economy), Republic of Korea.

References

[1] "Segway," http://www.segway.com/.

[2] S. H. Jeong and T. Takayuki, "Wheeled inverted pendulum type assistant robot: design concept and mobile control," in *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS '07)*, p. 1937, 1932, 2007.

[3] S. S. Kim and S. Jung, "Control experiment of a wheel-driven mobile inverted pendulum using neural network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 297–303, 2008.

[4] J. S. Noh, G. H. Lee, H. J. Choi, and S. Jung, "Robust control of a mobile inverted pendulum robot using a RBF neural network controller," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics, (ROBIO '08)*, pp. 1932–1937, February 2009.

[5] H. J. Lee and S. Jung, "Development of car like mobile inverted pendulum system : BalBOT VI," *The Korean Robotics Society*, vol. 4, no. 4, pp. 289–297, 2009.

[6] C. Rui and N. H. McClamroch, "Stabilization and asymptotic path tracking of a rolling disk," in *Proceedings of the 34th IEEE Conference on Decision and Control*, pp. 4294–4299, December 1995.

[7] G. C. Nandy and Y. Xu, "Dynamic model of a gyroscopic wheel," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2683–2688, May 1998.

[8] Y. Xu, K. W. Au, G. C. Nandy, and H. B. Brown, "Analysis of actuation and dynamic balancing for a single wheel robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1789–1794, October 1998.

[9] Y. Xu, H. B. Brown, and K. W. Au, "Dynamic mobility with single-wheel configuration," *International Journal of Robotics Research*, vol. 18, no. 7, pp. 728–738, 1999.

[10] S. J. Tsai, E. D. Ferreira, and C. J. Raredis, "Control of the gyrover: a single-wheel gyroscopically stabilized robot," in *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS '99)*, pp. 179–184, 1999.

[11] Y. Xu and S. K. W. Au, "Stabilization and path following of a single wheel robot," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 2, pp. 407–419, 2004.

[12] Y. S. Xu and Y. S. Ou, *Control of One-Wheel Robots*, Springer, 2005.

[13] W. Nukulwuthiopas, S. Laowattana, and T. Maneewarn, "Dynamic modeling of a one-wheel robot by using Kane's method," in *Proceedings of the IEEE International Conference on Industrial Technology, (IEEE ICIT '02)*, pp. 524–529, 2002.

[14] A. Alasty and H. Pendar, "Equations of motion of a single-wheel robot in a rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 879–884, April 2005.

[15] Z. Zhu, A. Al Mamun, P. Vadakkepat, and T. H. Lee, "Line tracking of the Gyrobot—a gyroscopically stabilized single-wheeled robot," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics, (ROBIO '06)*, pp. 293–298, December 2006.

[16] Z. Zhu, M. P. Naing, and A. Al-Mamun, "Integrated ADAMS+MATLAB environment for design of an autonomous single wheel robot," in *Proceedings of the 35th Annual Conference of the IEEE Industrial Electronics Society, (IECON '09)*, pp. 2253–2258, November 2009.

[17] T. B. Lauwers, G. A. Kantor, and R. L. Hollis, "A dynamically stable single-wheeled mobile robot with inverse mouse-ball

drive," in *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA '06)*, pp. 2884–2889, May 2006.

[18] U. Nagarajan, A. Mampetta, G. A. Kantor, and R. L. Hollis, "State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'10)*, pp. 998–1003, 2009.

[19] U. Nagarajan, G. Kantor, and R. L. Hollis, "Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot," in *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA '09)*, pp. 3743–3748, May 2009.

[20] P. K. Kim, J. H. Park, and S. Jung, "Experimental studies of balancing control for a disc-typed mobile robot using a neural controller: GYROBO," in *Proceedings of the IEEE International Symposium on Intelligent Control, (ISIC '10)*, pp. 1499–1503, September 2010.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 867178, 9 pages
doi:10.1155/2012/867178

Research Article

Robust Adaptive Control via Neural Linearization and Compensation

Roberto Carmona Rodríguez and Wen Yu

Departamento de Control Automatico, CINVESTAV-IPN, Avenue.IPN 2508, 07360 Mexico City, DF, Mexico

Correspondence should be addressed to Wen Yu, yuw@ctrl.cinvestav.mx

Received 6 October 2011; Revised 4 January 2012; Accepted 5 January 2012

Academic Editor: Isaac Chairez

Copyright © 2012 R. C. Rodríguez and W. Yu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose a new type of neural adaptive control via dynamic neural networks. For a class of unknown nonlinear systems, a neural identifier-based feedback linearization controller is first used. Dead-zone and projection techniques are applied to assure the stability of neural identification. Then four types of compensator are addressed. The stability of closed-loop system is also proven.

1. Introduction

Feedback control of the nonlinear systems is a big challenge for engineer, especially when we have no complete model information. A reasonable solution is to identify the nonlinear, then a adaptive feedback controller can be designed based on the identifier. Neural network technique seems to be a very effective tool to identify complex nonlinear systems when we have no complete model information or, even, consider controlled plants as "black box".

Neuroidentifier could be classified as static (feed forward) or as dynamic (recurrent) ones [1]. Most of publications in nonlinear system identification use static networks, for example multilayer perceptrons, which are implemented for the approximation of nonlinear function in the right-side hand of dynamic model equations [2]. The main drawback of these networks is that the weight updating utilize information on the local data structures (local optima) and the function approximation is sensitive to the training dates [3]. Dynamic neural networks can successfully overcome this disadvantage as well as present adequate behavior in presence of unmodeled dynamics because their structure incorporate feedback [4–6].

Neurocontrol seems to be a very useful tool for unknown systems, because it is model-free control, that is, this controller does not depend on the plant. Many kinds of neurocontrol were proposed in recent years, for example, supervised neuro control [7] is able to clone the human actions. The neural network inputs correspond to sensory information perceived by the human, and the outputs correspond to the human control actions. Direct inverse control [1] uses an inverse model of the plant cascaded with the plant, so the composed system results in an identity map between the desired response and the plant one, but the absence of feedback dismisses its robustness; internal model neurocontrol [8] that used forward and inverse model is within the feedback loop. Adaptive neurocontrol has two kinds of structure: indirect and direct adaptive control. Direct neuroadaptive may realize the neurocontrol by neural network directly [1]. The indirect method is the combination of the neural network identifier and adaptive control, the controller is derived from the on-line identification [5].

In this paper we extend our previous results in [9, 10]. In [9], the neurocontrol was derived by gradient principal, so the neural control is local optimal. No any restriction is needed, because the controller did not include the inverse of the weights. In [10], we assume the inverse of the weights exists, so the learning law was normal. The main contributions of this paper are (1) a special weights updating law is proposed to assure the existence of neurocontrol. (2) Four different robust compensators are proposed. By means of a Lyapunov-like analysis, we derive stability conditions for

the neuroidentifier and the adaptive controller. We show that the neuroidentifier-based adaptive control is effective for a large classes of unknown nonlinear systems.

2. Neuroidentifier

The controlled nonlinear plant is given as

$$\dot{x}_t = f(x_t, u_t, t), \qquad x_t \in \Re^n,\ u_t \in \Re^n, \tag{1}$$

where $f(x_t)$ is unknown vector function. In order to realize indirect neural control, a parallel neural identifier is used as in [9, 10] (in [5] the *series-parallel* structure is used):

$$\dot{\hat{x}}_t = A\hat{x}_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\phi(\hat{x}_t)\gamma(u_t), \tag{2}$$

where $\hat{x}_t \in \Re^n$ is the state of the neural network, $W_{1,t}, W_{2,t} \in \Re^{n\times n}$ are the weight matrices, $A \in \Re^{n\times n}$ is a stable matrix. The vector functions $\sigma(\cdot) \in \Re^n$, $\phi(\cdot) \in \Re^{n\times n}$ is a diagonal matrix. Function $\gamma(\cdot)$ is selected as $\|\gamma(u_t)\|^2 \le \overline{u}.$, for example $\gamma(\cdot)$ may be linear saturation function,

$$\gamma(u_t) = \begin{cases} u_t, & \text{if } |u_t| < b, \\ \overline{u}, & \text{if } |u_t| \ge b. \end{cases} \tag{3}$$

The elements of the weight matrices are selected as monotone increasing functions, a typical presentation is sigmoid function:

$$\sigma_i(\hat{x}_t) = \frac{a_i}{1 + e^{-b_i\hat{x}_t}} - c_i, \tag{4}$$

where $a_i, b_i, c_i > 0$. In order to avoid $\phi(\hat{x}_t) = 0$, we select

$$\phi_i(\hat{x}_t) = \frac{a_i}{1 + e^{-b_i\hat{x}_t}} + c_i. \tag{5}$$

Remark 1. The dynamic neural network (2) has been discussed by many authors, for example [4, 5, 9, 10]. It can be seen that Hopfield model is the special case of this networks with $A = \mathrm{diag}\{a_i\}$, $a_i := -1/R_iC_i$, $R_i > 0$ and $C_i > 0$. R_i and C_i are the resistance and capacitance at the ith node of the network, respectively.

Let us define identification error as

$$\Delta_t = \hat{x}_t - x_t. \tag{6}$$

Generally, dynamic neural network (2) cannot follow the nonlinear system (1) exactly. The nonlinear system may be written as

$$\dot{x}_t = Ax_t + W_1^0\sigma(x_t) + W_2^0\phi(x_t)\gamma(u_t) - \widetilde{f}_t, \tag{7}$$

where W_1^0 and W_2^0 are initial matrices of $W_{1,t}$ and $W_{2,t}$

$$W_1^0\Lambda_1^{-1}W_1^{0T} \le \overline{W}_1, \qquad W_2^0\Lambda_2^{-1}W_2^{0T} \le \overline{W}_2. \tag{8}$$

$\overline{W}_1$ and $\overline{W}_2$ are prior known matrices, vector function $\widetilde{f}_t$ can be regarded as modelling error and disturbances. Because $\sigma(\cdot)$ and $\phi(\cdot)$ are chosen as sigmoid functions, clearly they satisfy the following *Lipschitz* property:

$$\begin{aligned} \widetilde{\sigma}^T\Lambda_1\widetilde{\sigma} &\le \Delta_t^T D_\sigma \Delta_t, \\ \left(\widetilde{\phi}_t\gamma(u_t)\right)^T\Lambda_2\left(\widetilde{\phi}_t\gamma(u_t)\right) &\le \overline{u}\Delta_t^T D_\phi \Delta_t, \end{aligned} \tag{9}$$

where $\widetilde{\sigma} = \sigma(\hat{x}_t) - \sigma(x_t)$, $\widetilde{\phi} = \phi(\hat{x}_t) - \phi(x_t)$, Λ_1, Λ_2, D_σ, and D_ϕ are known positive constants matrices. The error dynamic is obtained from (2) and (7):

$$\dot{\Delta}_t = A\Delta_t + \widetilde{W}_{1,t}\sigma(\hat{x}_t) + \widetilde{W}_{2,t}\phi(\hat{x}_t)\gamma(u_t) + W_1^0\widetilde{\sigma} + W_2^0\widetilde{\phi}\gamma(u_t) + \widetilde{f}_t, \tag{10}$$

where $\widetilde{W}_{1,t} = W_{1,t} - W_1^0$, $\widetilde{W}_{2,t} = W_{2,t} - W_2^0$. As in [4, 5, 9, 10], we assume modeling error is bounded.

(A1) the unmodeled dynamic $\widetilde{f}$ satisfies

$$\widetilde{f}_t^T\Lambda_f^{-1}\widetilde{f}_t \le \overline{\eta}. \tag{11}$$

Λ_f is a known positive constants matrix.

If we define

$$R = \overline{W}_1 + \overline{W}_2 + \Lambda_f, \qquad Q = D_\sigma + \overline{u}D_\phi + Q_0, \tag{12}$$

and the matrices A and Q_0 are selected to fulfill the following conditions:

(1) the pair $(A, R^{1/2})$ is controllable, the pair $(Q^{1/2}, A)$ is observable,

(2) local frequency condition [9] satisfies frequency condition:

$$A^TR^{-1}A - Q \ge \frac{1}{4}\left[A^TR^{-1} - R^{-1}A\right]R\left[A^TR^{-1} - R^{-1}A\right]^T, \tag{13}$$

then the following assumption can be established.

(A2) There exist a stable matrix A and a strictly positive definite matrix Q_0 such that the matrix Riccati equation:

$$A^TP + PA + PRP + Q = 0 \tag{14}$$

has a positive solution $P = P^T > 0$.

This condition is easily fulfilled if we select A as stable diagonal matrix. Next Theorem states the learning procedure of neuroidentifier.

Theorem 2. *Subject to assumptions* **A1** *and* **A2** *being satisfied, if the weights $W_{1,t}$ and $W_{2,t}$ are updated as*

$$\begin{aligned} \dot{W}_{1,t} &= s_t\left[-K_1P\Delta_t\sigma^T(\hat{x}_t)\right], \\ \dot{W}_{2,t} &= s_t\mathrm{Pr}\left[-K_2P\phi(\hat{x}_t)\gamma(u_t)\Delta_t^T\right], \end{aligned} \tag{15}$$

where $K_1,\ K_2 > 0$, P is the solution of Riccati equation (14), $\mathrm{Pr}_i[\omega]$ $(i = 1, 2)$ *are projection functions which are defined as* $\omega = K_2 P\phi(\hat{x}_t)\gamma(u_t)\Delta_t^T$

$$\mathrm{Pr}[-\omega] = \begin{cases} -\omega, & \text{condition}, \\ -\omega + \dfrac{\left\|\widetilde{W}_{2,t}\right\|^2}{tr\left(\widetilde{W}_{2,t}^T(K_2P)\widetilde{W}_{2,t}\right)}\omega & \text{otherwise}, \end{cases} \tag{16}$$

where the "condition" is $\|\widetilde{W}_{2,t}\| < r$ *or* $[\|\widetilde{W}_{2,t}\| = r$ *and* $\mathrm{tr}(-\omega\widetilde{W}_{2,t}) \le 0]$, $r < \|W_2^0\|$ *is a positive constant.* s_t *is a dead-zone function*

$$s_t = \begin{cases} 1, & \text{if } \|\Delta_t\|^2 > \lambda_{\min}^{-1}(Q_0)\overline{\eta}, \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

then the weight matrices and identification error remain bounded, that is,

$$\Delta_t \in L_\infty, \qquad W_{1,t} \in L_\infty, \qquad W_{2,t} \in L_\infty, \tag{18}$$

for any $T > 0$ *the identification error fulfills the following tracking performance:*

$$\frac{1}{T}\int_0^T \|\Delta_t\|_{Q_0}^2 dt \le \kappa\overline{\eta} + \frac{\Delta_0^T P\Delta_0}{T}, \tag{19}$$

where κ *is the condition number of* Q_0 *defined as* $\kappa = \lambda_{\max}(Q_0)/\lambda_{\min}(Q_0)$.

Proof. Select a Lyapunov function as

$$V_t = \Delta_t^T P\Delta_t + \mathrm{tr}\left\{\widetilde{W}_{1,t}^T K_1^{-1}\widetilde{W}_{1,t}\right\} + \mathrm{tr}\left\{\widetilde{W}_{2,t}^T K_2^{-1}\widetilde{W}_{2,t}\right\}, \tag{20}$$

where $P \in \Re^{n\times n}$ is positive definite matrix. According to (10), the derivative is

$$\begin{aligned} \dot{V}_t = {} & \Delta_t^T\left(PA + A^TP\right)\Delta_t + 2\Delta_t^T P\widetilde{W}_{1,t}\sigma(\hat{x}_t) \\ & + 2\Delta_t^T P\widetilde{W}_{2,t}\phi(\hat{x}_t)\gamma(u_t) + 2\Delta_t^T P\widetilde{f}_t \\ & + 2\Delta_t^T P\left[W_1^*\widetilde{\sigma} + W_1^*\widetilde{\phi}\gamma(u_t)\right] + 2\,\mathrm{tr}\left\{\dot{\widetilde{W}}_{1,t}^T K_1^{-1}\widetilde{W}_{1,t}\right\} \\ & + 2\,\mathrm{tr}\left\{\dot{\widetilde{W}}_{2,t}^T K_2^{-1}\widetilde{W}_{2,t}\right\}. \end{aligned} \tag{21}$$

Since $\Delta_t^T P W_1^*\widetilde{\sigma}_t$ is scalar, using (9) and matrix inequality

$$X^TY + \left(X^TY\right)^T \le X^T\Lambda^{-1}X + Y^T\Lambda Y, \tag{22}$$

where $X, Y, \Lambda \in \Re^{n\times k}$ are any matrices, Λ is any positive definite matrix, we obtain

$$\begin{aligned} 2\Delta_t^T P W_1^*\widetilde{\sigma}_t &\le \Delta_t^T P W_1^*\Lambda_1^{-1}W_1^{*T}P\Delta_t + \widetilde{\sigma}_t^T\Lambda_1\widetilde{\sigma}_t \\ &\le \Delta_t^T\left(P\overline{W}_1P + D_\sigma\right)\Delta_t, \\ 2\Delta_t^T P W_2^*\widetilde{\phi}_t\gamma(u_t) &\le \Delta_t^T\left(P\overline{W}_2P + \overline{u}D_\phi\right)\Delta_t. \end{aligned} \tag{23}$$

In view of the matrix inequality (22) and (**A1**),

$$2\Delta_t^T P\widetilde{f}_t \le \Delta_t^T P\Lambda_f P\Delta_t + \overline{\eta}. \tag{24}$$

So we have

$$\begin{aligned} \dot{V}_t \le {} & \Delta_t^T\Big[PA + A^TP + P\left(\overline{W}_1 + \overline{W}_2 + \Lambda_f\right)P \\ & + \left(D_\sigma + \overline{u}D_\phi + Q_0\right)\Big]\Delta_t \\ & + 2\,\mathrm{tr}\left\{\dot{\widetilde{W}}_{1,t}^T K_1^{-1}\widetilde{W}_{1,t}\right\} + 2\Delta_t^T P\widetilde{W}_{1,t}\sigma(\hat{x}_t) + \overline{\eta} - \Delta_t^T Q_0\Delta_t \\ & + 2\,\mathrm{tr}\left\{\dot{\widetilde{W}}_{2,t}^T K_2^{-1}\widetilde{W}_{2,t}\right\} + 2\Delta_t^T P\widetilde{W}_{2,t}\phi(\hat{x}_t)\gamma(u_t). \end{aligned} \tag{25}$$

Since $\dot{\widetilde{W}}_{1,t} = \dot{W}_{1,t}$ and $\dot{\widetilde{W}}_{2,t} = \dot{W}_{2,t}$, if we use (**A2**), we have

$$\begin{aligned} \dot{V}_t \le {} & 2\,\mathrm{tr}\left\{\left[K_1^{-1}\dot{W}_{1,t}^T + K_1P\Delta_t\sigma^T(\hat{x}_t)\right]\widetilde{W}_{1,t}\right\} + \overline{\eta} - \Delta_t^T Q_0\Delta_t \\ & + 2\,\mathrm{tr}\left\{\left[K_2^{-1}\dot{W}_{2,t} + P\phi(\hat{x}_t)\gamma(u_t)\Delta_t^T\right]\widetilde{W}_{2,t}\right\}. \end{aligned} \tag{26}$$

(I) if $\|\Delta_t\|^2 > \lambda_{\min}^{-1}(Q_0)\overline{\eta}$, using the updating law as (15) we can conclude that

$$\begin{aligned} \dot{V}_t \le {} & 2\,\mathrm{tr}\left\{\left[\mathrm{Pr}\left[P\phi(\hat{x}_t)\gamma(u_t)\Delta_t^T\right] + P\phi(\hat{x}_t)\gamma(u_t)\Delta_t^T\right]\widetilde{W}_{2,t}\right\} \\ & - \Delta_t^T Q_0\Delta_t + \overline{\eta}, \end{aligned} \tag{27}$$

(a) if $\|\widetilde{W}_{2,t}\| < r$ or $[\|\widetilde{W}_{2,t}\| = r$ and $\mathrm{tr}(-\omega\widetilde{W}_{2,t}) \le 0]$, $\dot{V}_t \le -\lambda_{\min}(Q_0)\|\Delta_t\|^2 + \overline{\eta} < 0$,

(b) if $\|\widetilde{W}_{2,t}\| = r$ and $\mathrm{tr}(-\omega\widetilde{W}_{2,t}) > 0$

$$\begin{aligned} \dot{V}_t &\le 2\,\mathrm{tr}\left\{K_2P\frac{\left\|\widetilde{W}_{2,t}\right\|^2}{\mathrm{tr}\left(\widetilde{W}_{2,t}^T(K_2P)\widetilde{W}_{2,t}\right)}\omega\widetilde{W}_{2,t}\right\} - \Delta_t^T Q_0\Delta_t + \overline{\eta} \\ &\le -\Delta_t^T Q_0\Delta_t + \overline{\eta} < 0. \end{aligned} \tag{28}$$

V_t is bounded. Integrating (27) from 0 up to T yields

$$V_T - V_0 \le -\int_0^T \Delta_t^T Q_0\Delta_t dt + \overline{\eta}T. \tag{29}$$

Because $\kappa \ge 1$, we have

$$\begin{aligned} \int_0^T \Delta_t^T Q_0\Delta_t dt &\le V_0 - V_T + \int_0^T \Delta_t^T Q_0\Delta_t dt \le V_0 + \overline{\eta}T, \\ &\le V_0 + \kappa\overline{\eta}T, \end{aligned} \tag{30}$$

where κ is condition number of Q_0

(II) If $\|\Delta_t\|^2 \le \lambda_{\min}^{-1}(Q_0)\overline{\eta}$, the weights become constants, V_t remains bounded. And

$$\begin{aligned} \int_0^T \Delta_t^T Q_0\Delta_t dt &\le \int_0^T \lambda_{\max}(Q_0)\|\Delta_t\|^2 dt \\ &\le \frac{\lambda_{\max}(Q_0)}{\lambda_{\min}(Q_0)}\overline{\eta}T \le V_0 + \kappa\overline{\eta}T. \end{aligned} \tag{31}$$

From (I) and (II), V_t is bounded, (18) is realized. From (20) and $\widetilde{W}_{1,t} = W_{1,t} - W_1^0, \widetilde{W}_{2,t} = W_{2,t} - W_2^0$ we know $V_0 = \Delta_0^T P\Delta_0$. Using (30) and (31), (19) is obtained. The theorem is proved. □

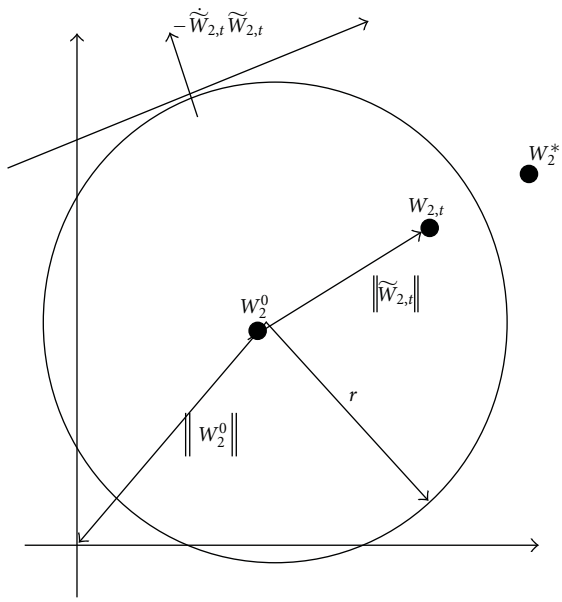


Figure 1: Projection algorithm.

Remark 3. The weight update law (15) uses two techniques. The dead-zone s_t is applied to overcome the robust problem caused by unmodeled dynamic $\widetilde{f}_t$. In presence of disturbance or unmodeled dynamics, adaptive procedures may easily go unstable. The lack of robustness of parameters identification was demonstrated in [11] and became a hot issue in 1980s. Dead-zone method is one of simple and effective tool. The second technique is projection approach which may guarantee that the parameters remain within a constrained region and do not alter the properties of the adaptive law established without projection [12]. The projection approach proposed in this paper is explained in Figure 1. We hope to force $W_{2,t}$ inside the ball of center W_2^0 and radius r. If $\|\widetilde{W}_{2,t}\| < r$, we use the normal gradient algorithm. When $W_{2,t} - W_2^0$ is on the ball, and the vector $W_{2,t}$ points either inside or along the ball, that is, $(d/dt)\|\widetilde{W}_{2,t}\|^2 = 2\,\mathrm{tr}(-\omega\widetilde{W}_{2,t}) \le 0$, we also keep this algorithm. If $\mathrm{tr}(-\omega\widetilde{W}_{2,t}) > 0$, $\mathrm{tr}[(-\omega + (\|\widetilde{W}_{2,t}\|^2/\mathrm{tr}(\widetilde{W}_{2,t}^T (K_2P)\widetilde{W}_{2,t}))\omega)\widetilde{W}_{2,t}] < 0$, so $(d/dt)\|\widetilde{W}_{2,t}\|^2 < 0, W_{2,t}$ are directed toward the inside or the ball, that is, $W_{2,t}$ will never leave the ball. Since $r < \|W_2^0\|, W_{2,t} \neq 0$.

Remark 4. Figure 1 and (7) show that the initial conditions of the weights influence identification accuracy. In order to find good initial weights, we design an offline method. From above theorem, we know the weights will convergence to a zone. We use any initial weights, W_1^0 and W_2^0, after T_0, the identification error should become smaller, that is, W_{1,T_0} and W_{2,T_0} are better than W_1^0 and W_2^0. We use following steps to find the initial weights.

(1) Start from any initial value for $W_1^0 = W_{1,0}$, $W_2^0 = W_{2,0}$.

(2) Do identification until training time arrives T_0.

(3) If the $\|\Delta(T_0)\| < \|\Delta(0)\|$, let W_{1,T_0}, W_{2,T_0} as a new W_1^0 and W_2^0, go to 2 to repeat the identification process.

(4) If the $\|\Delta(T_0)\| \ge \|\Delta(0)\|$, stop this offline identification, now W_{1,T_0}, W_{2,T_0} are the final initial weights.

Remark 5. Since the updating rate is K_iP $(i = 1, 2)$, and K_i can be selected as any positive matrix, the learning process of the dynamic neural network (15) is free of the solution of Riccati equation (14).

Remark 6. Let us notice that the upper bound (19) turns out to be "sharp", that is, in the case of not having any uncertainties (exactly matching case: $\widetilde{f} = 0$) we obtain $\overline{\eta} = 0$ and, hence,

$$\limsup_{T\to\infty} \frac{1}{T}\int_0^T \|\Delta_t\|_{Q_0}^2 dt = 0 \tag{32}$$

from which, for this special situation, the asymptotic stability property ($\|\Delta_t\| \underset{t\to\infty}{\to} 0$) follows. In general, only the asymptotic stability "in average" is guaranteed, because the dead-zone parameter $\overline{\eta}$ can be never set zero.

3. Robust Adaptive Controller Based on Neuro Identifier

From (7) we know that the nonlinear system (1) may be modeled as

$$\begin{aligned}\dot{x}_t &= Ax_t + W_1^*\sigma(x_t) + W_2^*\phi(x_t)\gamma(u_t) + \widetilde{f}\\ &= Ax_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\phi(x_t)\gamma(u_t)\\ &\quad + \widetilde{f} + \widetilde{W}_{1,t}\sigma(\hat{x}_t) + \widetilde{W}_{2,t}\phi(x_t)\gamma(u_t) + W_{1,t}^*\tilde{\sigma}_t + W_1^*\tilde{\phi}\gamma(u_t).\end{aligned} \tag{33}$$

Equation (33) can be rewritten as

$$\dot{x}_t = Ax_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\phi(x_t)\gamma(u_t) + d_t, \tag{34}$$

where

$$d_t = \widetilde{f} + \widetilde{W}_{1,t}\sigma(\hat{x}_t) + \widetilde{W}_{2,t}\phi(x_t)\gamma(u_t) + W_{1,t}^*\tilde{\sigma}_t + W_1^*\tilde{\phi}\gamma(u_t). \tag{35}$$

If updated law of $W_{1,t}$ and $W_{2,t}$ is (15), $W_{1,t}$ and $W_{2,t}$ are bounded. Using the assumption **(A1)**, d_t is bounded as $\overline{d} = \sup_t\|d_t\|$.

The object of adaptive control is to force the nonlinear system (1) following a optimal trajectory $x_t^* \in \Re^r$ which is assumed to be smooth enough. This trajectory is regarded as a solution of a nonlinear reference model:

$$x_t^* = \varphi(x_t^*, t), \tag{36}$$

with a fixed initial condition. If the trajectory has points of discontinuity in some fixed moments, we can use any approximating trajectory which is smooth. In the case of regulation problem $\varphi(x_t^*, t) = 0$, $x^*(0) = c$, c is constant. Let us define the sate trajectory error as

$$\Delta_t^* = x_t - x_t^*. \tag{37}$$

From (34) and (36) we have

$$\dot{\Delta}_t^* = Ax_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\phi(x_t)\gamma(u_t) + d_t - \varphi(x_t^*, t). \tag{38}$$

Let us select the control action $\gamma(u_t)$ as linear form

$$\gamma(u_t) = U_{1,t} + \left[W_{2,t}\phi(\hat{x}_t)\right]^{-1} U_{2,t}, \tag{39}$$

where $U_{1,t} \in \Re^n$ is direct control part and $U_{2,t} \in \Re^n$ is a compensation of unmodeled dynamic d_t. As $\varphi(x_t^*, t)$, x_t^*, $W_{1,t}\sigma(\hat{x}_t)$ and $W_{2,t}\phi(\hat{x}_t)$ are available, we can select $U_{1,t}$ as

$$U_{1,t} = \left[W_{2,t}\phi(\hat{x}_t)\right]^{-1}\left[\varphi(x_t^*, t) - Ax_t^* - W_{1,t}\sigma(\hat{x}_t)\right]. \tag{40}$$

Because $\phi(\hat{x}_t)$ in (5) is different from zero, and $W_{2,t} \neq 0$ by the projection approach in Theorem 2. Substitute (39) and (40) into (38), we have So the error equation is

$$\dot{\Delta}_t^* = A\Delta_t^* + U_{2,t} + d_t. \tag{41}$$

Four robust algorithms may be applied to compensate d_t.

(A) Exactly Compensation. From (7) and (2) we have

$$d_t = \left(\dot{x}_t - \dot{\hat{x}}_t\right) - A(x_t - \hat{x}_t). \tag{42}$$

If $\dot{x}_t$ is available, we can select $U_{2,t}$ as $U_{2,t}^a = -d_t$, that is,

$$U_{2,t}^a = A(x_t - \hat{x}_t) - \left(\dot{x}_t - \dot{\hat{x}}_t\right). \tag{43}$$

So, the ODE which describes the state trajectory error is

$$\dot{\Delta}_t^* = A\Delta_t^*. \tag{44}$$

Because A is stable, Δ_t^* is globally asymptotically stable.

$$\lim_{t\to\infty} \Delta_t^* = 0. \tag{45}$$

(B) An Approximate Method. If $\dot{x}_t$ is not available, an approximate method may be used as

$$\dot{x}_t = \frac{x_t - x_{t-\tau}}{\tau} + \delta_t, \tag{46}$$

where $\delta_t > 0$, is the differential approximation error. Let us select the compensator as

$$U_{2,t}^b = A(x_t - \hat{x}_t) - \left(\frac{x_t - x_{t-\tau}}{\tau} - \dot{\hat{x}}_t\right). \tag{47}$$

So $U_{2,t}^b = U_{2,t}^a + \delta_t$, (44) become

$$\dot{\Delta}_t^* = A\Delta_t^* + \delta_t. \tag{48}$$

Define Lyapunov-like function as

$$V_t = \Delta_t^{*T} P_2 \Delta_t^*, \qquad P_2 = P_2^T > 0. \tag{49}$$

The time derivative of (49) is

$$\dot{V}_t = \Delta_t^*\left(A^T P_2 + P_2 A\right)\Delta_t^* + 2\Delta_t^{*T} P_2 \delta_t, \tag{50}$$

$2\Delta_t^T P_2 \delta_t$ can be estimated as

$$2\Delta_t^{*T} P_2 \delta_t \leq \Delta_t^{*T} P_2 \Lambda P_2 \Delta_t^* + \delta_t^T \Lambda^{-1} \delta_t \tag{51}$$

where Λ is any positive define matrix. So (50) becomes

$$\dot{V}_t \leq \Delta_t^*\left(A^T P_2 + P_2 A + P_2 \Lambda P_2 + Q_2\right)\Delta_t^* + \delta_t^T \Lambda^{-1}\delta_t - \Delta_t^{*T} Q_2 \Delta_t^*, \tag{52}$$

where Q is any positive define matrix. Because A is stable, there exit Λ and Q_2 such that the matrix Riccati equation:

$$A^T P_2 + P_2 A + P_2 \Lambda P_2 + Q_2 = 0 \tag{53}$$

has positive solution $P_2 = P_2^T > 0$. Defining the following seminorms:

$$\left|\left|\Delta_t^*\right|\right|_{Q_2}^2 = \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T \Delta_t^* Q_2 \Delta_t^* dt, \tag{54}$$

where $Q_2 = Q_2 > 0$ is the given weighting matrix, the state trajectory tracking can be formulated as the following optimization problem:

$$J_{\min} = \min_{u_t} J, \qquad J = \left|\left|x_t - x_t^*\right|\right|_{Q_2}^2. \tag{55}$$

Note that

$$\overline{\lim_{T\to\infty}} \frac{1}{T}\left(\Delta_0^{*T} P_2 \Delta_0^*\right) = 0 \tag{56}$$

based on the dynamic neural network (2), the control law (47) can make the trajectory tracking error satisfies the following property:

$$\left|\left|\Delta_t^*\right|\right|_{Q_2}^2 \leq \|\delta_t\|_{\Lambda^{-1}}^2. \tag{57}$$

A suitable selection of Λ and Q_2 can make the Riccati equation (53) has positive solution and make $\|\Delta_t^*\|_{Q_2}^2$ small enough if τ is small enough.

(C) Sliding Mode Compensation. If $\dot{x}_t$ is not available, the sliding mode technique may be applied. Let us define Lyapunov-like function as

$$V_t = \Delta_t^{*T} P_3 \Delta_t^*, \tag{58}$$

where P_3 is a solution of the Lyapunov equation:

$$A^T P_3 + P_3 A = -I. \tag{59}$$

Using (41) whose time derivative is

$$\dot{V}_t = \Delta_t^*\left(A^T P_3 + P_3 A\right)\Delta_t^* + 2\Delta_t^{*T} P_3 U_{2,t} + 2\Delta_t^{*T} P_3 d_t. \tag{60}$$

According to sliding mode technique, we may select $u_{2,t}$ as

$$U_{2,t}^c = -kP_3^{-1}\,\mathrm{sgn}(\Delta_t^*), \qquad k > 0, \tag{61}$$

where k is positive constant,

$$\mathrm{sgn}(\Delta_t^*) = \begin{cases} 1 & \Delta_t^* > 0 \\ 0 & \Delta_t^* = 0 \\ -1 & \Delta_t^* < 0 \end{cases} \tag{62}$$

$$\mathrm{sgn}(\Delta_t^*) = \left[\mathrm{sgn}\left(\Delta_{1,t}^*\right), \ldots \mathrm{sgn}\left(\Delta_{n,t}^*\right)\right]^T \in \Re^n.$$

Substitute (59) and (61) into (60)

$$\begin{aligned}\dot{V}_t &= -||\Delta_t^*||^2 - 2k||\Delta_t^*|| + 2\Delta_t^{*T} P d_t \\ &\le -||\Delta_t^*||^2 - 2k||\Delta_t^*|| + 2\lambda_{\max}(P)||\Delta_t^*||\|d_t\| \\ &= -||\Delta_t^*||^2 - 2||\Delta_t^*||(k - \lambda_{\max}(P)\|d_t\|).\end{aligned} \tag{63}$$

If we select

$$k > \lambda_{\max}(P_3)\overline{d}, \tag{64}$$

where $\overline{d}$ is define as (35), then $\dot{V}_t < 0$. So,

$$\lim_{t\to\infty} \Delta_t^* = 0. \tag{65}$$

(D) Local Optimal Control. If $\dot{x}_t$ is not available and $\dot{x}_t$ is not approximated as (B). In order to analyze the tracking error stability, we introduce the following Lyapunov function:

$$V_t(\Delta_t^*) = \Delta_t^* P_4 \Delta_t^*, \qquad P_4 = P_4^T > 0. \tag{66}$$

Using (41), whose time derivative is

$$V_t = \Delta_t^* \left(A^T P_4 + P_4 A\right)\Delta_t^* + 2\Delta_t^{*T} P_4 U_{2,t} + 2\Delta_t^{*T} P_4 d_t, \tag{67}$$

$2\Delta_t^{*T} P_4 d_t$ can be estimated as

$$2\Delta_t^{*T} P_4 d_t \le \Delta_t^* P_4 \Lambda_4^{-1} P_4 \Delta_t^* + d_t^T \Lambda_4 d_t. \tag{68}$$

Substituting (68) in (67), adding and subtracting the term $\Delta_t^{*T} Q_4 \Delta_t^*$ and $U_{2,t}^{dT} R_4 U_{2,t}^d$ with $Q_4 = Q_4^T > 0$ and $R_4 = R_4^T > 0$, we formulate

$$\begin{aligned}V_t \le{}& \Delta_t^* \left(A^T P_4 + P_4 A + P_4 \Lambda_4 P_4 + Q_4\right)\Delta_t^* \\ &+ 2\Delta_t^{*T} P_4 U_{2,t}^d + U_{2,t}^{dT} R_4 U_{2,t}^d + d_t^T \Lambda_4^{-1} d_t - \Delta_t^* Q \Delta_t^* \\ &- U_{2,t}^{dT} R_4 U_{2,t}^d.\end{aligned} \tag{69}$$

Because A is stable, there exit Λ_4 and Q_4 such that the matrix Riccati equation:

$$A^T P_4 + P_4 A + P_4 \Lambda_4 P_4 + Q_4 = 0. \tag{70}$$

So (69) is

$$V_t \le -\left(||\Delta_t^*||_{Q_4}^2 + \left\|U_{2,t}^d\right\|_{R_4}^2\right) + \Psi\left(U_{2,t}^d\right) + d_t^T \Lambda_4^{-1} d_t, \tag{71}$$

where

$$\Psi\left(U_{2,t}^d\right) = 2\Delta_t^{*T} P_4 U_{2,t}^d + U_{2,t}^{dT} R_4 U_{2,t}^d. \tag{72}$$

We reformulate (71) as

$$||\Delta_t^*||_{Q_4}^2 + \left\|U_{2,t}^d\right\|_{R_4}^2 \le \Psi\left(U_{2,t}^d\right) + d_t^T \Lambda_4^{-1} d_t - V_t. \tag{73}$$

Then, integrating each term from 0 to τ, dividing each term by τ, and taking the limit, for $\tau \to \infty$ of these integrals' supreme, we obtain

$$\begin{aligned}&\overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T \Delta_t^{*T} Q_4 \Delta_t^* dt + \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T U_{2,t}^{dT} R_4 U_{2,t}^d dt \\ &\quad\le \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T d_t^T \Lambda_4^{-1} d_t dt + \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T \Psi\left(U_{2,t}^d\right) dt \\ &\qquad + \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T \left[-V_t\right] dt.\end{aligned} \tag{74}$$

In the view of definitions of the seminorms (55), we have

$$||\Delta_t^*||_{Q_4}^2 + \left\|U_{2,t}^d\right\|_{R_4}^2 \le \|d_t\|_{\Lambda_4^{-1}}^2 + \overline{\lim_{T\to\infty}} \frac{1}{T}\int_0^T \Psi\left(U_{2,t}^d\right) dt. \tag{75}$$

It fixes a *tolerance level* for the trajectory-tracking error. So, the control goal now is to minimize $\Psi(U_{2,t}^d)$ and $\|d_t\|_{\Lambda_4^{-1}}^2$. To minimize $\|d_t\|_{\Lambda_4^{-1}}^2$, we should minimize Λ_4^{-1}. From (13), if select Q_4 to make (70) have solution, we can choose the minimal Λ_4^{-1} as

$$\Lambda_4^{-1} = A^{-T} Q_4 A^{-1}. \tag{76}$$

To minimizing $\Psi(U_{2,t}^d)$, we assume that, at the given t (positive), $x^*(t)$ and $\widehat{x}(t)$ are already realized and do not depend on $U_{2,t}^d$. We name the $U_{2,t}^{d*}(t)$ as *the locally optimal* control, because it is calculated based only on "local" information. The solution of this optimization problem is given by

$$\begin{aligned}&\min \Psi\left(u_{2,t}^d\right) = 2\Delta_t^{*T} P_4 u_{2,t}^d + U_{2,t}^{dT} R_4 U_{2,t}^d. \\ &\text{subject: } A_0\left(U_{1,t} + U_{2,t}^d\right) \le B_0.\end{aligned} \tag{77}$$

It is typical quadratic programming problem. Without restriction U^* is selected according to the linear squares optimal control law:

$$u_{2,t}^d = -2R_4^{-1} P_4 \Delta_t^*. \tag{78}$$

Remark 7. Approaches (A) and (C) are exactly compensations of d_t, Approach (A) needs the information of $\dot{x}_t$. Because Approach (C) uses the sliding mode control $U_{2,t}^c$ that is inserted in the closed-loop system, chattering occurs in the control input which may excite unmodeled high-frequency dynamics. To eliminate chattering, the boundary layer compensator can be used, it offers a continuous approximation to the discontinuous sliding mode control law inside the boundary layer and guarantees the output tracking error within any neighborhood of the origin [13].

Finally, we give following design steps for the robust neurocontrollers proposed in this paper.

(1) According to the dimension of the plant (1), design a neural networks identifier (2) which has the same dimension as the plant. In (2), A can be selected a stable matrix. A will influence the dynamic response of the neural network. The bigger eigenvalues of A will make the neural network slower. The initial conditions for $W_{1,t}$ and $W_{2,t}$ are obtained as in Remark 4.

(2) Do online identification. The learning algorithm is (15) with the dead zone in Theorem 2. We assume we know the upper bound of modeling error, we can give a value for $\overline{\eta}$. Q_0 is chosen such that Riccati equation (14) has positive defined solution, R can be selected as any positive defined matrix because Λ_1^{-1} is arbitrary positive defined matrix. The updating rate in the learning algorithm (15) is $K_1 P$, and K_1 can be selected as any positive defined matrix, so the learning process

is free of the solution P of the Riccati equations (14). The larger K_1P is selected, the faster convergence the neuroidentifier has.

(3) Use robust control (39) and one of compensation of (43), (47), (61), and (78).

4. Simulation

In this section, a two-link robot manipulator is used to illustrate the proposed approach. Its dynamics of can be expressed as follows [14]:

$$M(\theta)\,\ddot{\theta} + V\left(\theta,\dot{\theta}\right)\dot{\theta} + G(\theta) + F_d\left(\dot{\theta}\right) = \tau, \tag{79}$$

where $\theta \in \Re^2$ consists of the joint variables, $\dot{\theta} \in \Re^2$ denotes the links velocity, τ is the generalized forces, $M(\theta)$ is the intertie matrix, $V(\theta,\dot{\theta})$ is centripetal-Coriolis matrix, and $G(\theta)$ is gravity vector, $F_d(\dot{\theta})$ is the friction vector. $M(\theta)$ represents the positive defined inertia matrix. If we define $x_1 = \theta = [\theta_1, \theta_2]$ is joint position, $x_2 = \dot{\theta}$ is joint velocity of the link, $x_t = [x_1, x_2]^T$, (79) can be rewritten as state space form [15]:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= H(x_t, u_t), \end{aligned} \tag{80}$$

where $u_t = \tau$ is control input,

$$H(x_t, u_t) = -M(x_1)^{-1}[C(x_1, x_2)\dot{x}_1 + G(x_1) + F\dot{x}_1 + u_t]. \tag{81}$$

Equation (80) can also be rewritten as

$$\dot{x}_1 = \int_0^t H(x_\tau, u_\tau)d\tau + H(x_0, u_0). \tag{82}$$

So the dynamic of the two-link robot (79) is in form of (1) with

$$f(x_t, u_t, t) = \int_0^t H(x_\tau, u_\tau)d\tau + H(x_0, u_0). \tag{83}$$

The values of the parameters are listed below: $m_1 = m_2 = 1.53\,\text{kg}$, $l_1 = l_2 = 0.365\,\text{m}$, $r1 = r2 = 0.1$, $v1 = v2 = 0.4$, $k1 = k2 = 0.8$. Let define $\hat{x} = [\hat{\theta}_1, \hat{\theta}_2]^T$, and $u = [\tau_1, \tau_2]^T$, the neural network for control is represented as

$$\dot{\hat{x}} = A\hat{x} + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\phi(\hat{x})u. \tag{84}$$

We select $A = \left[\begin{smallmatrix} -1.5 & 0 \\ 0 & -1 \end{smallmatrix}\right]$, $\phi(\hat{x}_t) = \text{diag}(\phi_1(\hat{x}_1), \phi_2(\hat{x}_2))$, $\sigma(\hat{x}_t) = [\sigma_2(\hat{x}_2), \sigma_2(\hat{x}_2)]^T$

$$\begin{aligned} \sigma_i(\hat{x}_i) &= \frac{2}{(1 + e^{-2\hat{x}_i})} - \frac{1}{2}, \\ \phi_i(\hat{x}_i) &= \frac{2}{(1 + e^{-2\hat{x}_i})} + \frac{1}{2}, \end{aligned} \tag{85}$$

where $i = 1, 2$. We used Remark 4 to obtain a suitable W_1^0 and W_2^0, start from random values, $T_0 = 100$. After 2 loops, $\|\Delta(T_0)\|$ does not decrease, we let the $W_{1,300}$ and $W_{2,300}$ as the new $W_1^0 = \left[\begin{smallmatrix} 0.51 & 3.8 \\ -2.3 & 1.51 \end{smallmatrix}\right]$ and $W_2^0 = \left[\begin{smallmatrix} 3.12 & -2.78 \\ 5.52 & -4.021 \end{smallmatrix}\right]$. For the update laws (15), we select $\overline{\eta} = 0.1$, $r = 5$, $K_1P = K_1P = \left[\begin{smallmatrix} 5 & 0 \\ 0 & 2 \end{smallmatrix}\right]$. If we select the generalized forces as

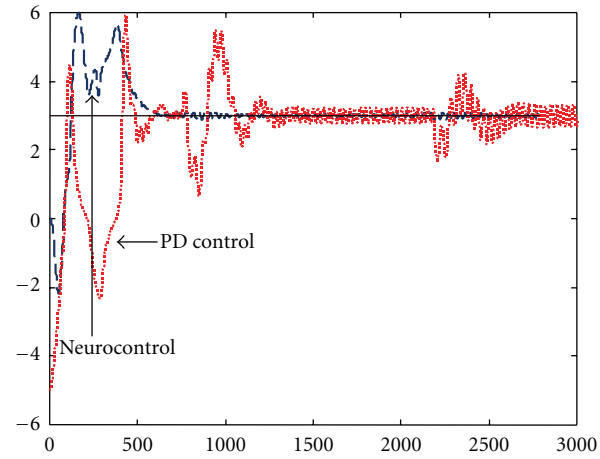


FIGURE 2: Tracking control of θ_1 (method B).

$$\tau_1 = 7\sin t, \qquad \tau_2 = 0. \tag{86}$$

Now we check the neurocontrol. We assume the robot is changed at $t = 480$, after that $m_1 = m_2 = 3.5\,\text{kg}$, $l_1 = l_2 = 0.5\,\text{m}$, and the friction becomes disturbance as $D\sin((\pi/3)t)$, D is a positive constant. We compare neurocontrol with a PD control as

$$\tau_{\text{PD}} = -10(\theta - \theta^*) - 5\left(\dot{\theta} - \dot{\theta}^*\right), \tag{87}$$

where $\theta_1^* = 3$; θ_2^* is square wave. So $\varphi(\theta^*) = \dot{\theta}^* = 0$.

The neurocontrol is (39)

$$\begin{aligned} \tau_{\text{neuro}} = {} & [W_{2,t}\phi(\hat{x})]^+[\varphi(x_t^*, t) - Ax_t^* - W_{1,t}\sigma(\hat{x})] \\ & + [W_{2,t}\phi(\hat{x})]^+ U_{2,t}. \end{aligned} \tag{88}$$

$U_{2,t}$ is selected to compensate the unmodeled dynamics. Sine f is unknown method. (A) exactly compensation, cannot be used.

(B) $D = 1$. The link velocity $\dot{\theta}$ is measurable, as in (43),

$$U_{2,t} = A\left(\theta - \hat{\theta}\right) - \left(\dot{\theta} - \dot{\hat{\theta}}\right). \tag{89}$$

The results are shown in Figures 2 and 3.

(C) $D = 0.3$. $\dot{\theta}$ is not available, the sliding mode technique may be applied. we select $u_{2,t}$ as (61).

$$u_{2,t} = -10 \times \text{sgn}(\theta - \theta^*). \tag{90}$$

The results are shown in Figures 4 and 5.

(D) $D = 3$. We select $Q = 1/2$, $R = 1/20$, $\Lambda = 4.5$, the solution of following Riccati equation:

$$A^TP + PA + P\Lambda P_t + Q = -\dot{P} \tag{91}$$

is $P = \left[\begin{smallmatrix} 0.33 & 0 \\ 0 & 0.33 \end{smallmatrix}\right]$. If without restriction τ, the linear squares optimal control law:

$$u_{2,t} = -2R^{-1}P(\theta - \theta^*) = \begin{bmatrix} -20 & 0 \\ 0 & -20 \end{bmatrix}(\theta - \theta^*). \tag{92}$$

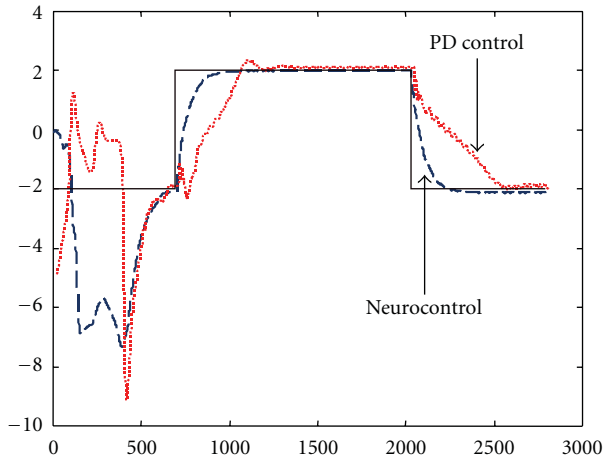


Figure 3: Tracking control of θ_2 (method B).

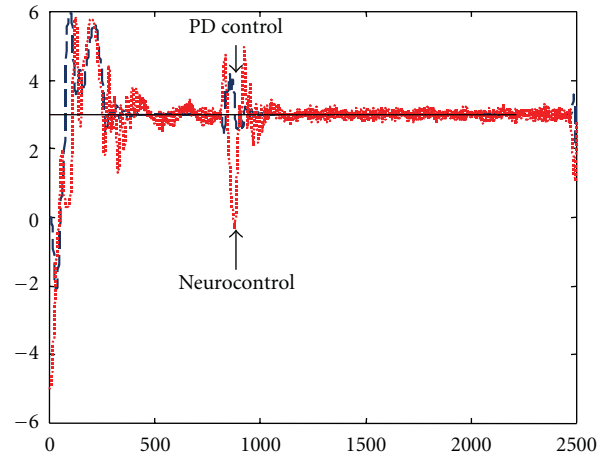


Figure 4: Tracking control of θ_1 (method C).

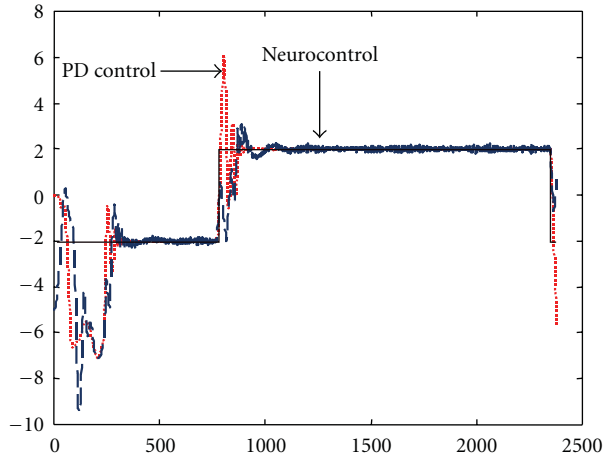


Figure 5: Tracking control of θ_2 (method C).

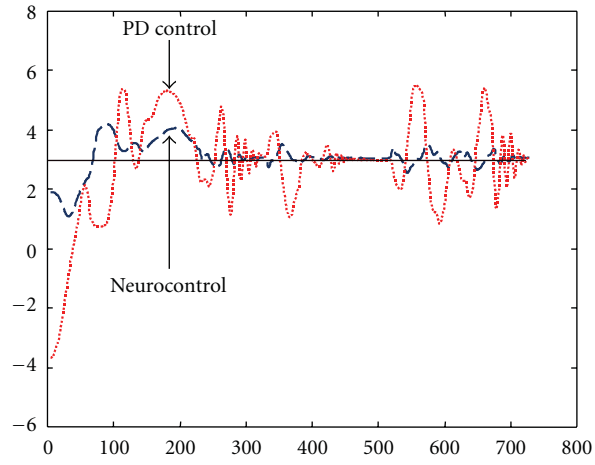


Figure 6: Tracking control of θ_1 (method D).

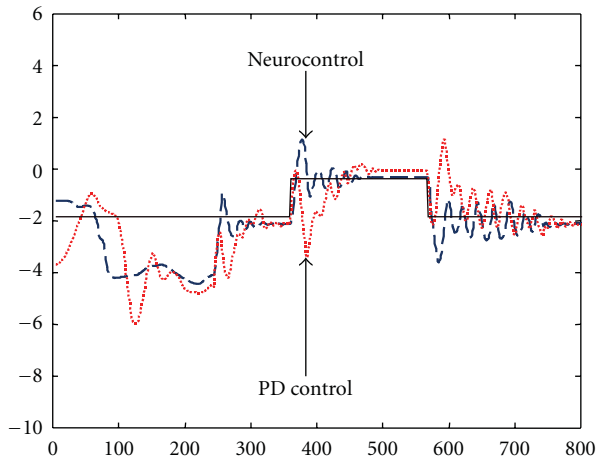


Figure 7: Tracking control of θ_2 (method D).

The results of local optimal compensation are shown in Figures 6 and 7.

We may find that the neurocontrol is robust and effective when the robot is changed.

5. Conclusion

By means of Lyapunov analysis, we establish bounds for both the identifier and adaptive controller. The main contributions of our paper is that we give four different compensation methods and prove the stability of the neural controllers.

References

[1] K. S. Narendra and K. Parthasarathy, "Identi cation and control for dynamic systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, pp. 4–27, 1990.

[2] S. Jagannathan and F. L. Lewis, "Identi cation of nonlinear dynamical systems using multilayered neural networks," *Automatica*, vol. 32, no. 12, pp. 1707–1712, 1996.

[3] S. Haykin, *Neural Networks-A comprehensive Foundation*, Macmillan College, New York, NY, USA, 1994.

[4] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 422–431, 1995.
[5] G. A. Rovithakis and M. A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 3, pp. 400–412, 1994.
[6] W. Yu and X. Li, "Some new results on system identi cation with dynamic neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 412–417, 2001.
[7] E. Grant and B. Zhang, "A neural net approach to supervised learning of pole placement," in *Proceedings of the IEEE Symposium on Intelligent Control*, 1989.
[8] K. J. Hunt and D. Sbarbaro, "Neural networks for nonlinear internal model control," *IEE Proceedings D—Control Theory and Applications*, vol. 138, no. 5, pp. 431–438, 1991.
[9] A. S. Poznyak, W. Yu, E. N. Sanchez, and J. P. Perez, "Nonlinear adaptive trajectory tracking using dynamic neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1402–1411, 1999.
[10] W. Yu and A. S. Poznyak, "Indirect adaptive control via parallel dynamic neural networks," *IEE Proceedings Control Theory and Applications*, vol. 146, no. 1, pp. 25–30, 1999.
[11] B. Egardt, *Stability of Adaptive Controllers*, vol. 20 of *Lecture Notes in Control and Information Sciences*, Springer, Berlin, Germany, 1979.
[12] P. A. Ioannou and J. Sun, *Robust Adaptive Control*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.
[13] M. J. Corless and G. Leitmann, "Countinuous state feedback guaranteeing uniform ultimate boundness for uncertain dynamic systems," *IEEE Transactions on Automatic Control*, vol. 26, pp. 1139–1144, 1981.
[14] F. L. Lewis, A. Yeşildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 388–399, 1996.
[15] S. Nicosia and A. Tornambe, "High-gain observers in the state and parameter estimation of robots having elastic joins," *System & Control Letter*, vol. 13, pp. 331–337, 1989.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 761019, 11 pages
doi:10.1155/2012/761019

Research Article

Dynamics Model Abstraction Scheme Using Radial Basis Functions

Silvia Tolu,[1] Mauricio Vanegas,[2] Rodrigo Agís,[1] Richard Carrillo,[1] and Antonio Cañas[1]

[1] *Department of Computer Architecture and Technology, CITIC ETSI Informática y de Telecomunicación, University of Granada, Spain*
[2] *PSPC Group, Department of Biophysical and Electronic Engineering (DIBE), University of Genoa, Italy*

Correspondence should be addressed to Silvia Tolu, stolu@atc.ugr.es

Received 27 July 2011; Accepted 24 January 2012

Academic Editor: Wen Yu

Copyright © 2012 Silvia Tolu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a control model for object manipulation. Properties of objects and environmental conditions influence the motor control and learning. System dynamics depend on an unobserved external context, for example, work load of a robot manipulator. The dynamics of a robot arm change as it manipulates objects with different physical properties, for example, the mass, shape, or mass distribution. We address active sensing strategies to acquire object dynamical models with a radial basis function neural network (RBF). Experiments are done using a real robot's arm, and trajectory data are gathered during various trials manipulating different objects. Biped robots do not have high force joint servos and the control system hardly compensates all the inertia variation of the adjacent joints and disturbance torque on dynamic gait control. In order to achieve smoother control and lead to more reliable sensorimotor complexes, we evaluate and compare a sparse velocity-driven versus a dense position-driven control scheme.

1. Introduction

Current research of biomorphic robots can highly benefit from simulations of advance learning paradigms [1–5] and also from knowledge acquired from biological systems [6–8]. The basic control of robot actuators is usually implemented by adopting a classic scheme [9–14]: (a) the desired trajectory, in joint coordinates, is obtained using the inverse kinematics [12, 14, 15] and (b) efficient motor commands drive, for example, an arm, making use of the inverse dynamics model or using high power motors to achieve rigid movements. Different controlling schemes using hybrid position/velocity and forces have been introduced mainly for industrial robots [9, 10]. In fact, industrial robots are equipped with digital controllers, generally of PID type with no possibility of modifying the control algorithms to improve their performance. Robust control (control using PID paradigms [12, 14–19]) is very power consuming and highly reduces autonomy of nonrigid robots. Traditionally, the major application of industrial robots is related to tasks that require only position control of the arm. Nevertheless, there are other important robotic tasks that require interaction between the robot's end-effector and the environment. System dynamics depend on an unobserved external context [3, 20], for example, work load of a robot manipulator.

Biped robots do not have high force joint servos and the control system hardly compensates all the inertia variation of the adjacent joints and disturbance torque on dynamic gait control [21]. Motivated by these concerns and the promising preliminary results [22], we evaluate a sparse velocity-driven control scheme. In this case, smooth motion is naturally achieved, cutting down jerky movements and reducing the propagation of vibrations along the whole platform. Including the dynamic model into the control scheme becomes important for an accurate manipulation, therefore, it is crucial to study strategies to acquire it. There are other bioinspired model abstraction approaches that could take advantage of this strategy [23–26]. Conventional artificial neural networks [27, 28] have been also applied to this issue [29–31]. In biological systems [6–8], the cerebellum [32, 33] seems to play a crucial role on model extraction tasks during manipulation [12, 34–36]. The cerebellar cortex has a unique, massively parallel modular architecture [34, 37, 38] that appears to be efficient in the model abstraction [11, 35].

Human sensorimotor recognition [8, 12, 37] continually learns from current and past sensory experiences. We set up an experimental methodology using a biped robot's arm [34, 36] which has been equipped, at its last arm-limb, with three acceleration sensors [39] to capture the dynamics of the different movements along the three spatial dimensions. In human body, there are skin sensors specifically sensitive to acceleration [8, 40]. They represent haptic sensors and provide acceleration signals during the arm motion. In order to be able to abstract a model from object manipulation, accurate data of the movements are required. We acquire the position and the acceleration along desired trajectories when manipulating different objects. We feed these data into the radial basis function network (RBF) [27, 31]. Learning is done off-line through the knowledge acquisition module. The RBF learns the dynamic model, that is, it learns to react by means of the acceleration in response to specific input forces and to reduce the effects of the uncertainty and nonlinearity of the system dynamics [41, 42].

To sum up, we avoid adopting a classic regular point-to-point strategy (see Figure 1(b)) with fine PID control modules [11, 14, 15] that drive the movement along a finely defined trajectory (position-based). This control scheme requires high power motors in order to achieve rigid movements (when manipulating heavy objects) and the whole robot augments vibration artefacts that make difficult accurate control. Furthermore, these platform jerks induce high noise in the embodied accelerometers. Contrary to this scheme, we define the trajectory by a reduced set of target points (Figure 1(c)) and implement a velocity-driven control strategy that highly reduces jerks.

2. Control Scheme

The motor-driver controller and the communication interface are implemented on an FPGA (Spartan XC31500 [43] embedded on the robot) (Figure 1(a)). Robonova-I [44] is a fully articulating mechanical biped robot, 0.3 m, 1.3 kg, controlled with sixteen motors [44]. We have tested two control strategies for a specific "L" trajectory previously defined for the robot's arm controlled with three motors (one in the shoulder and two in the arm, see Figure 2. This whole movement along three joints makes use of three degrees of freedom during the trajectory. The movement that involves the motor at the last arm-limb is the dominant one, as is evidenced from the sensorimotor values in Figure 4(b).

2.1. Control Strategies. The trajectory is defined in different ways for the two control strategies.

2.1.1. Dense Position-Driven. The desired joint trajectory is finely defined by a set of target positions (P_i) that regularly sample the desired movement (see Figure 1(b)).

2.1.2. Sparse Velocity-Driven. In this control strategy, the joint trajectory definition is carried out by specifying only positions related to changes in movement direction.

During all straight intervals, between a starting position P_1 and the next change of direction P_{cd}, we proceed to command the arm by modulating the velocity towards P_{cd} (see Figure 1(c)). Velocities (V_n) are calculated taking into account the last captured position and the time step in which the position has been acquired. Each target velocity for period T is calculated with the following expression, $V_n = (P_{\text{cd}} - P_n)/T$. The control scheme applies a force proportional to the target velocity. Figure 1(d(B)) illustrates how a smoother movement is achieved using sparse velocity-driven control.

The dense position-driven strategy reveals noisy vibrations and jerks because each motor always tries to get the desired position in the minimum time, whereas it would be better to efficiently adjust the velocity according to the whole target trajectory. The second strategy defines velocities dynamically along the trajectory as described above. Therefore, we need to sample the position regularly to adapt the velocity.

2.2. Modelling Robot Dynamics. The dynamics of a robot arm have a linear relationship to the inertial properties of the manipulator joints [45, 46]. In other words, for a specific context r they can be written in the form (1),

$$\tau = \Upsilon(q, \dot{q}, \ddot{q}) \cdot \pi_r, \tag{1}$$

where q, $\dot{q}$, and $\ddot{q}$ are joint angles, velocities, and accelerations respectively. Equation (1), based on fundamentals of robot dynamics [47], splits the dynamics in two terms. $\Upsilon(q, \dot{q}, \ddot{q})$ is a term that depends on kinematics properties of the arm such as direction of the axis of rotation of joints, and link lengths. Term π_r is a high-dimensional vector containing all inertial parameters of all links of the arm [45]. Now, let us consider that the manipulated objects are attached at the end of the arm, so we model the dynamics of the arm as the manipulated object being the last link of the arm. Then, manipulating different objects is equivalent to changing the physical properties of the last link of the arm. The first term of the equation remains constant in different models. It means that all kinematic quantities of the arm remain the same between different contexts. Under this assumption, we could use a set of models with known inertial parameters to infer a predictive model (RBF) of dynamics [27, 31, 48] for any possible context. From another point of view, the previous dynamic Equation (1) could be written in the form (2):

$$\tau = A(q)(\ddot{q}) + H(q, \dot{q}). \tag{2}$$

Each dynamic term is nonlinear and coupled [49] where $A(q)$ is the matrix of inertia, symmetric, and positive and $H(q, \dot{q})$ includes Coriolis, gravitational, and centrifugal forces. $A(q)$ could be inverted for each robot configuration. Approximating $A(q) = \widehat{A}(q)$ and $H(q, \dot{q}) = \widehat{H}(q, \dot{q})$, we obtain (3):

$$\widehat{\tau} = \widehat{A}(q) \cdot u + \widehat{H}(q, \dot{q}), \tag{3}$$

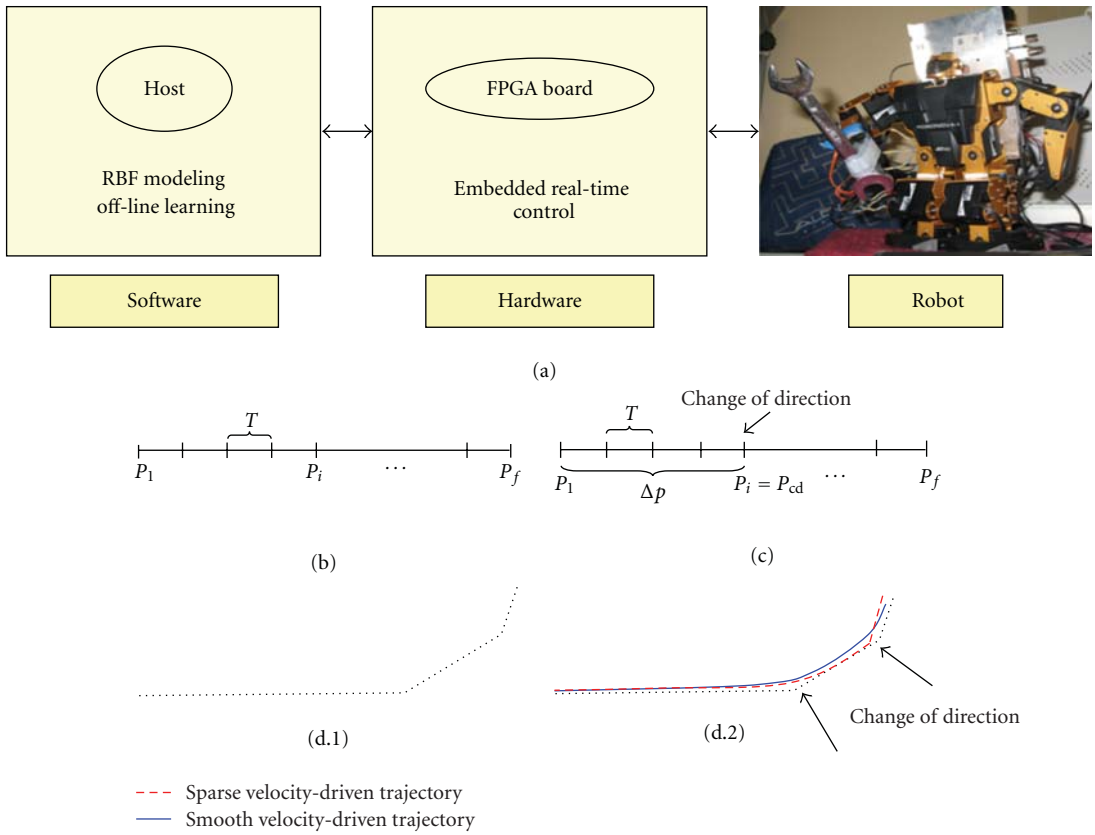


FIGURE 1: (a) Computing scheme. An FPGA processing platform embeds all the real-time control interfaces. The robot is connected to the FPGA which receives commands from the host PC. The robot is equipped with three accelerometers in its arm. The computer captures the sensor data. A RBF is used for learning the objects models from sensorimotor complexes. (b) Dense position-driven scheme. A motor moves from an initial position P_1 to a final position P_f temporally targeting each intermediate point P_i along this trajectory. (c) Sparse velocity-driven scheme. For each position P_i indicated above, a target velocity is calculated for a motor with a time step T of 0.015 s. Using a fixed sampling frequency for obtaining feedback position, we calculate the distance ΔP from this current position to a position P_{cd} corresponding to a change of direction. (d(A)) Dense position-driven trajectory. The trajectory is defined in terms of regularly sampled intermediate points (dashed line). (d(B)) The thinner continuous line represents the real curve executed by the robot's arm in the space under the sparse velocity-driven control. The black arrows indicate points in which changes of direction take place. The thicker continuous line corresponds to the smooth velocity-driven trajectory executed anticipating the points in which changes of direction are applied before the arm reaches them.

where u is a new control vector. $\hat{A}(q)$ and $\hat{H}(q, \dot{q})$ are estimations of the real robot terms. So we have the following equivalence relation (4):

$$u = \ddot{q}. \tag{4}$$

Component u is influenced only by a second-order term, it depends on the joint variables $(q, \dot{q})$, independently from the movement of each joint. To sum up, we replaced the nonlinear and coupled dynamic expressions with a second-order linear system of noncoupled equations. The inputs to the RBF network are the positions and velocities of the robot joints, while the outputs are the accelerations. The input-output relationship is in accordance with the equations of motion. In this work, we propose and show the RBF to be useful in approximating the unknown nonlinearities of the dynamical systems [41, 42, 50–56] through the control signal (4) found for the estimation method for the dynamics of the joints of a robot.

2.3. The RBF-Based Modelling. The RBF function is defined as (5):

$$z(x) = \phi(||x - \mu||), \tag{5}$$

where x is the n-dimensional input vector, μ is an n-dimensional vector called centre of the RBF, $\|\cdot\|$ is the Euclidean distance, and ϕ is the RBF outline.

We built our model as a linear combination of N RBF functions with N centres. The RBF output is given by (6):

$$\hat{y}(x) = \sum_{j=1}^{N} \beta_j z_j(x), \tag{6}$$

where B_j is the weight of the jth radial function, centered at μ and with an activation level z_j.

RBF has seven inputs: two per each of the three motors in the robot arm (joints) and one for the label of the trajectory which is executed (T_r). Three inputs encode the difference between the current joint position and the next

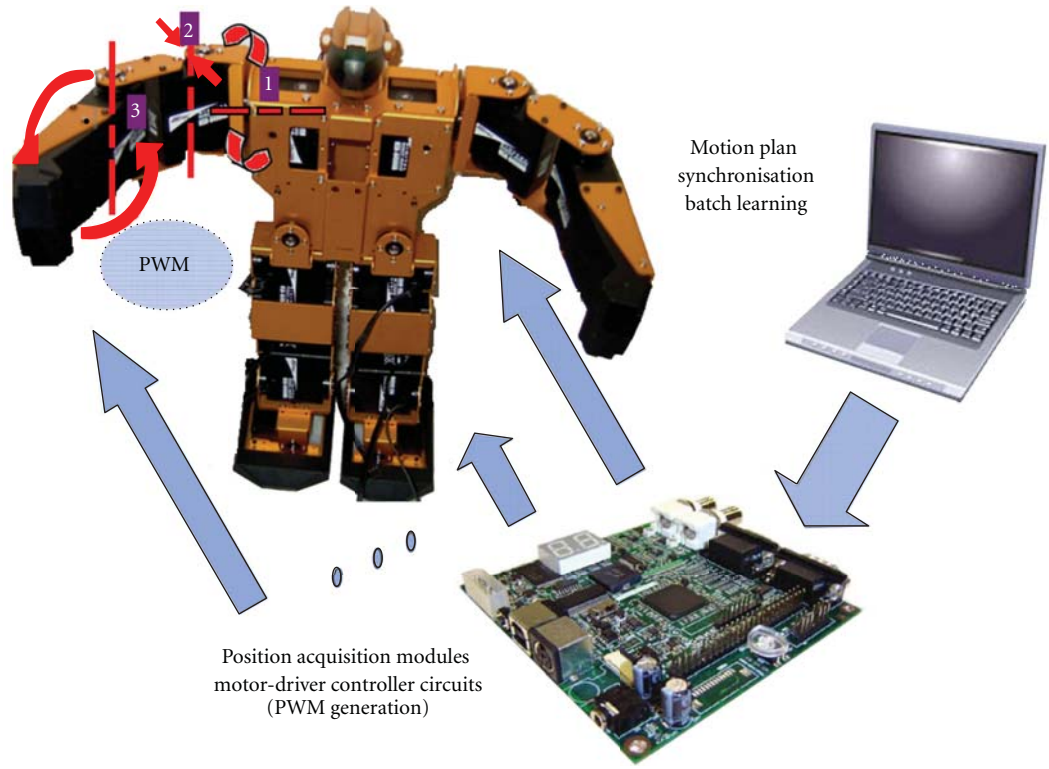


Figure 2: Diagram of the whole system architecture. The robot's picture in the left is showing the different joints of the 3 DOF Robonova arm involved in the movement. The PC applied appropriate motor commands to each joint of the arm at all times during the movement to follow the desired trajectory. To relieve the computer from interface computation and allow real-time communication with the robot, an FPGA board containing position acquisition modules and motor-driver controller circuits with PWM (pulse with modulation) was connected to the robot. The communication task between devices and the operations described above was synchronised by the computer speeding up the process. Finally, the batch learning with RBF of sensorimotor data collected during the movements was performed in the PC. The dataset was split up into training data (75%) and test data (25%).

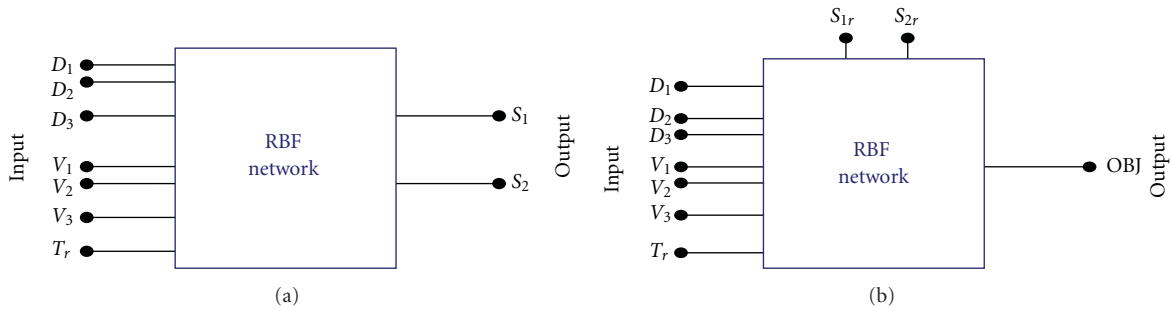


Figure 3: RBF inputs and outputs. (a) The network is used for function approximation of the sensor responses to specific inputs along different trajectories. (b) This network includes an output label (OBJ) for object classification and uses as inputs the actual sensor responses (connected to the upper part of the block).

target position in each of the joints (D_1, D_2, and D_3). The other three inputs (V_1, V_2, and V_3) encode the velocity computed on the basis of the previous three inputs. We aim to model the acceleration sensory outputs (S_1 and S_2) (S_3 does not provide further information than S_2 for the movement trajectory defined) and we also include one output (label) to classify different kinds of objects (OBJ). See the illustrative block in Figure 3(b). Therefore, we use two RBF networks. The first one (see Figure 3(a)) aims to approximate the acceleration sensor responses to the input motor commands. The second one (see Figure 3(b)) aims to discriminate object models (classification task) using as inputs the motor commands and also the actual acceleration estimations given by the sensors [57]. During learning, we feed the inputs and actual outputs of the training set in the network [27, 31]. During the test stage, we evaluate the error obtained from the sensors [39] that indicates how accurate the acquired model is and, therefore, how stable, predictable, and repeatable is

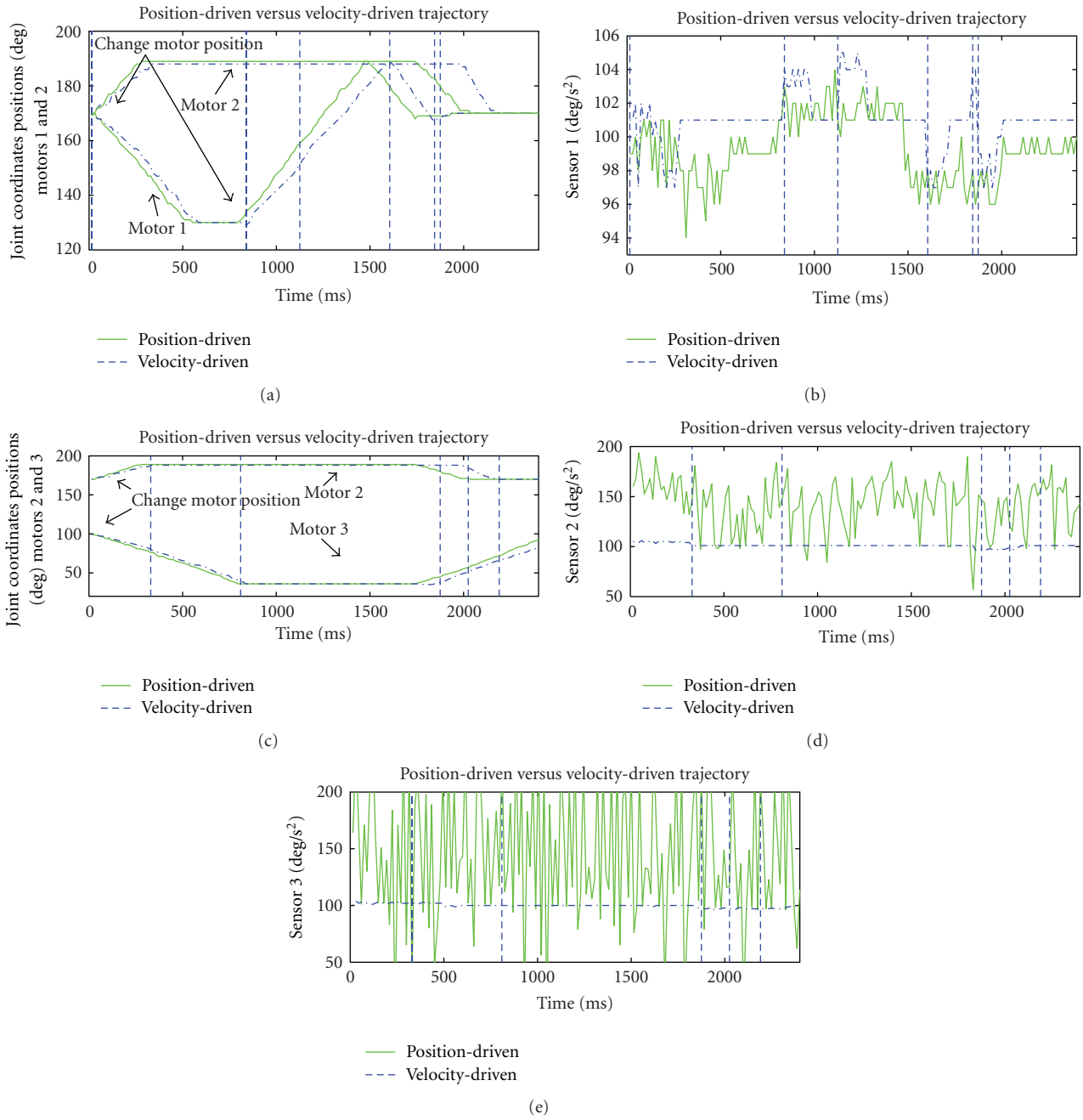


Figure 4: Plots (a) and (c) show the motor commands (in joint coordinates) and plots (b), (d), and (e) show the sensor responses during the movement. We compare the dense position-driven versus the sparse velocity-driven strategies for the case related to object 0 (see the experimental results). In (b) (Sensor 1), it can be clearly seen how the sensor responses are directly related with the motor command changes in plot (a). The sensor responses, obtained using the sparse velocity-driven strategy, are more stable. Motor commands increasing the derivative of joint coordinates cause increments in sensor 1 response during a certain period, while motor commands decreasing the derivative of joint coordinates cause decrements in sensor 1 response, plot (b). The piece of trajectory monitored by these plots is mainly related to sensor 1. It can be seen in the right plots (d) and (e) that the other sensors are not so sensitive to the motions in this example. Nevertheless, sparse velocity-driven control still leads to much more stable sensor responses. Dense position-driven control causes large vibrations that highly affect the capability of the neural network to approach the sensor response function.

the object manipulation. The error metric shown in the result figures (see Figure 5) is the root mean square error (RMSE) in degrees/s^2 of the different sensors along the whole trajectory. Finally, the classification error (related to the label output) indicates how accurately the network can classify a specific object from the positions, velocities, and accelerometer responses along the trajectory with a corresponding target output. A lower error in the predicted sensor responses indicates that the motion is performed in a smoother and more predictable way. Therefore, as can be seen in

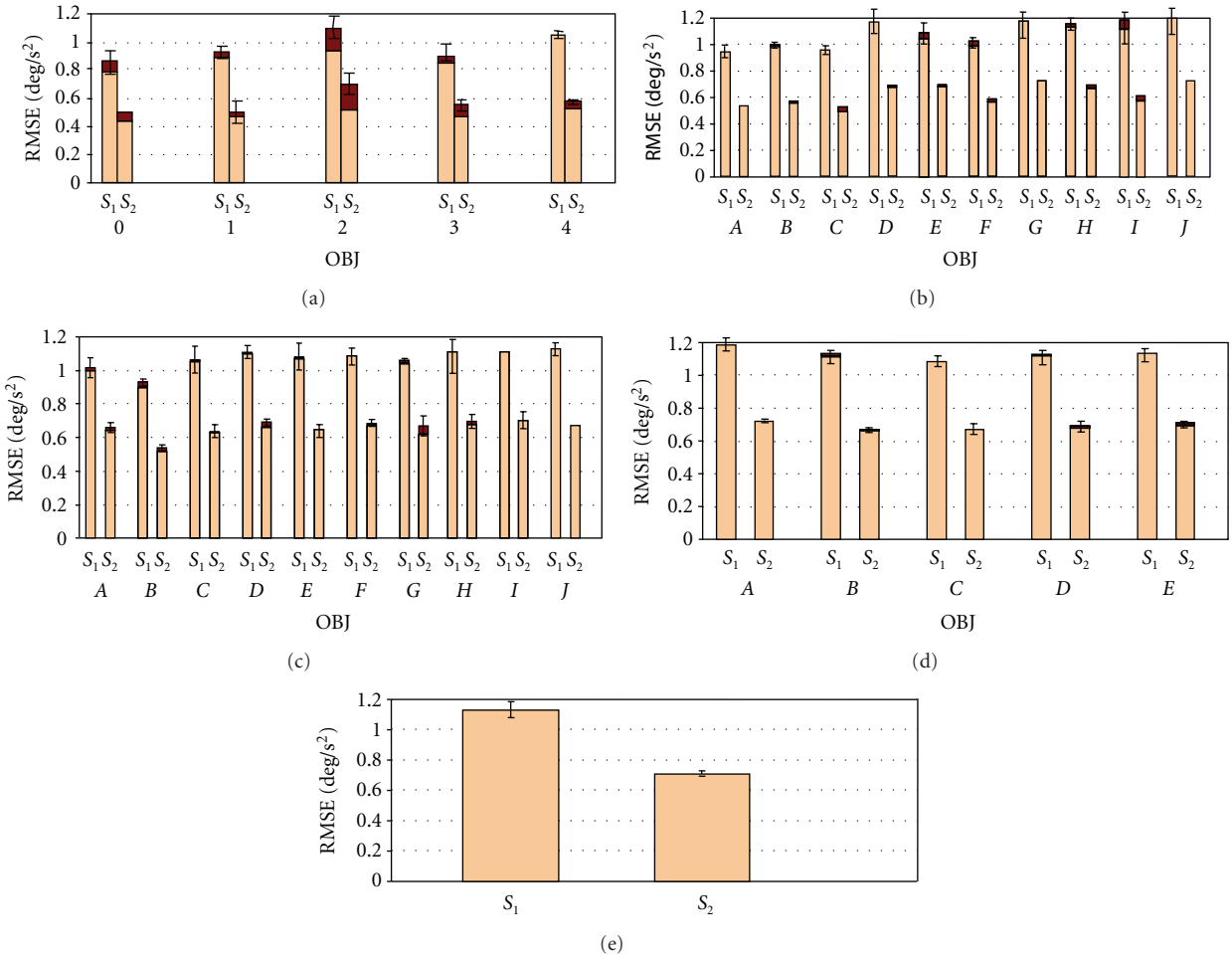


Figure 5: RMSE values for predicted data of sensors S_1 and S_2 for sparse velocity-driven control scheme. The darker colour in the top of the columns indicates the increment of RMSE in the test stage. (a) Using a different class (model) for each single object. (b) Using a different class (model) for each set of two objects. Cases (objects): *A* (0 and 1), *B* (0 and 2), *C* (0 and 3), *D* (0 and 4), *E* (1 and 2), *F* (1 and 3), *G* (1 and 4), *H* (2 and 3), *I* (2 and 4), and *J* (3 and 4). (c) Using a different class (model) for each set of three objects. Cases (objects): *A* (0, 1, and 2), *B* (0, 1, and 3), *C* (0, 1, and 4), *D* (0, 2, and 3), *E* (0, 2, and 4), *F* (0, 3, and 4), *G* (1, 2, and 3), *H* (objects 1, 2, and 4), *I* (1, 3, and 4), and *J* (2, 3, and 4). (d) Using a different class for each set of four objects. Cases (objects): *A* (0, 1, 2, and 4), *B* (0, 1, 2, and 3), *C* (0, 1, 3, and 4), *D* (0, 2, 3, and 4), *E* (1, 2, 3, and 4). (e) Using a single class for the only set of five objects. In this case the increment of RMSE in the test stage is neglectable. Cases (objects): (0, 1, 2, 3, and 4).

Figures 4, 5, and 6, strategies such as sparse velocity-driven control lead to a more reliable movement control scheme. Furthermore, this allows a better dynamic characterisation of different objects. This facilitates the use of adaptation strategies of control gains to achieve more accurate manipulation skills (although this issue is not specifically addressed in this work).

3. Results

Studies have shown high correlations between the inertia tensor and various haptic properties like length, height, orientation of the object, and position of the hand grasp [57–59]. The inertia tensor has the central role to be the perceptual basis for making haptic judgments of object properties.

In order to describe the approach and the pursued movements (several repetitions) and to test different dynamic models of the whole system (robot arm with object) [60, 61], the robot hand manipulated objects with different weights, shapes, and position of grasp. We acquired all the data from the position and accelerator sensors and we used them for the RBF off-line training. We considered the following cases of study:

(0) NO object;

(1) Scissors (0.05 kg, 0.14 m);

(2) Monkey wrench manipulated fixing the centre (0.07 kg, 0.16 m);

(3) Monkey wrench manipulated fixing an extreme (0.07 kg, 0.16 m);

(4) Two Allen keys (0.1 kg, 0.10 m).

We repeated manipulation experiments ten times for each case of study collecting sixty thousand data. We applied

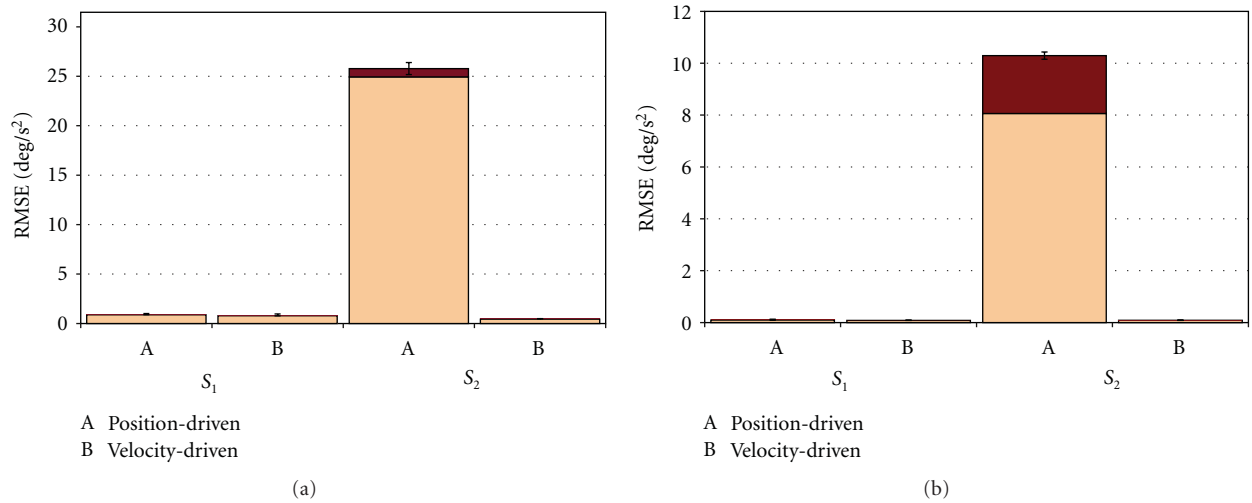


Figure 6: Comparison between dense position-driven control scheme (column A) and the sparse velocity-driven control scheme (column B) for the case related to object 0. RMSE (degrees/s^2) of predicted values of the sensors S_1 and S_2 using a sparse velocity-driven scheme (B) are always lower than the ones obtained using dense position-driven control (A). The sensor that suffers more significantly from jerks when using dense position-driven scheme is S_2. The darker colour in the top of the columns indicates the increment of RMSE in the test stage. The test results are always slightly higher. The variance value obtained with the cross-validation method for the test results is also included. Figures 6(a) and 6(b) are related to the cases of a nonfixed and a fixed robot respectively.

a cross-validation method defining different datasets for the training and test stages. We used the data of seven experiments out of ten for training and three for the test stage. We repeated this approach three times shuffling the experiments of the datasets for training and test. The neural network is built using the MBC model toolbox of MATLAB [62]. We chose the following algorithms: **TrialWidths** [63] for the width selection and **StepItRols** for the Lambda and centres selection [64]. We measured the performance calculating the RMSE in degrees/s^2 in the acceleration sensory outputs (S_1 and S_2).

We addressed a comparative study evaluating how accurately the neural network can abstract the dynamic model using a dense position-driven and a sparse velocity-driven schemes. Plots in Figure 4 illustrate how the motor commands are highly related with specific sensor responses (mainly in sensor 1 along this piece of trajectory). Figures 4(a) and 4(c) illustrate a piece of movement. Figures 4(b), 4(d) and 4(e) show that sparse velocity-driven control leads to much more stable sensor responses.

3.1. Abstracting the Dynamic Model during Manipulation. The goal of the RBF network is to model the sensory outputs given the motor commands along well defined trajectories (manipulating several objects). The purpose is to study how accurate models can be acquired if the neural network is trained with individual objects or with groups of "sets of objects" (as a single class). When we select the target "models" to abstract in the training stage with the neural network, we can define several target cases: 5 classes for abstracting single object models (Figure 5(a)), 10 classes for abstracting models for pairs of objects (Figure 5(b)), 10 classes for abstracting models of sets of "three objects" (Figure 5(c)), 5 classes for models of sets of "four objects" (Figure 5(d)), and finally, a single global model for the set of five objects (Figure 5(e)). The neural network, trained with data of two objects, presents the minimum of RMSE using 25 neurons in the hidden layer. For three objects, we used 66–80 neurons depending on the specific case, and for four objects, 80 neurons. But even with these larger numbers of neurons, the network was able to accurately classify the objects in the case of sets of two of them. During manipulation, if the model prediction significantly deviates from the actual sensory outputs, the system can assume that the model being applied is incorrect and should proceed to "switch" to another model or to "learn modifications" on the current model. In this task, we evaluate the performance of the "abstraction process" calculating the RMSE between the model sensor response and the actual sensor response along the movement. Figure 6 compares the performance achieved between the two control schemes under study for the cases of a nonfixed (Figure 6(a)) and a fixed robot (Figure 6(b)). Results indicate that the velocity driven strategy reduces the error to very low values in both cases.

Results in Figure 5 are obtained using a sparse velocity-driven control scheme. In Figure 5(a), they show how the RBF achieves different performance when trying to abstract models of different objects. For objects 1 and 3, both sensors lead to similar RMSE values as their dynamic models are similar. Objects 2 and 4 lead to higher RMSE values since they have more inertia than others. As we can see (comparing the results of Figures 5(a) and 5(b), if the network has to learn the dynamic model of more objects in the same class (shared model learning) at the same time, the error rate increases slightly.

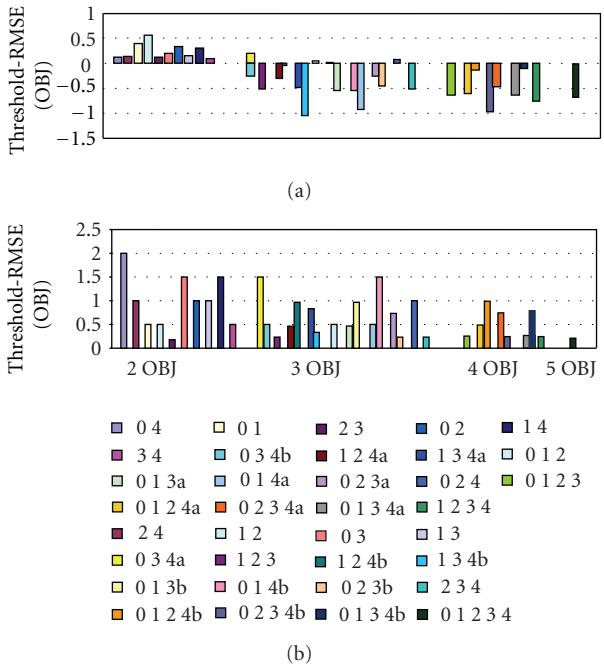


Figure 7: Bars represent the object discrimination power of the network (RBF) for all combinations of two, three, four, or five objects respectively (see paragraph 3 for object classification). If the RMSE (OBJ) value of the label output for a specific set of objects lies below the discrimination threshold, an accurate classification of objects can be achieved. Graphs show the result obtained with a non-fixed robot (Figure 7(a)) and with a fixed robot (Figure 7(b)), and each bar is associated to a combination of objects listed in the legend from left to right, respectively. A combination with two thresholds (i.e., 0 3 4) has two bars and two labels (0 3 4a and 0 3 4b) and bars are joined together in the graph.

3.2. Classifying Objects during Manipulation. In Figure 7, the discrimination power of the RBF based on the dynamic model itself is shown. In this case, we evaluate if the RBF is able to accurately distinguish among different objects assigning the correct labels (0, 1, 2, 3, and 4) by using only the inputs of the network (i.e., sensorimotor vectors). Evaluation of the classification performance is not straightforward and is based on the difference between the "discrimination threshold" (half of the distance between the closest target label values) and the RMSE value of the label output for a specific set of objects. We have chosen a single dimension distribution of the classes, therefore, distances between the different class labels are diverse. For instance, the "distance" between labels one and two is one, while the distance between labels one and four is three and the threshold is the half of that distance. For any set of two or five objects (i.e. A (0 and 1), B (1 and 3), and C (0, 1, 2, 3, and 4)) only one threshold value exists (Th(A) = 0.5, Th(B) = 1, and Th(C) = [0.5, 0.5, 0.5, 0.5]), while sets of three or four objects (i.e., D (1, 2, and 3), E (0, 2, and 3), and F (0, 1, 2, and 4), and G (1, 2, 3, and 4)) may have two threshold values (Th(D) = [0.5, 0.5], Th(E) = [1, 0.5], Th(F) = [0.5, 0.5, 1], and Th(G) = [0.5, 0.5, 0.5]). We see that the RBF is able to correctly classify the objects when using only two of them (values above zero)

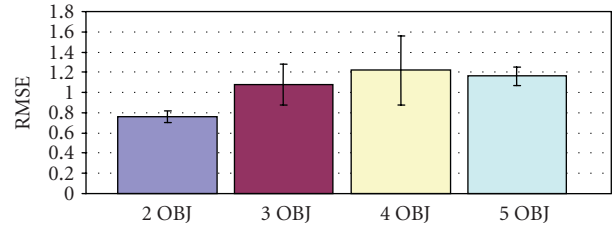


Figure 8: Mean of RMSE for objects (OBJ) using the sparse velocity-driven control scheme.

but the system becomes unreliable (negative values) when using three objects or more in the case of a standing robot (Figure 7(a)), while the RBF always correctly classifies the objects when the robot is fixed to the ground (Figure 7(b)).

3.3. Accurate Manipulation Strategy: Classification with Custom Control. We have abstracted dynamic models from objects and can use them for accurate control (Figure 9).

(a) Direct Control. We control different objects directly. During manipulation, the sensor responses are compared with a "shared model" to evaluate the performance. We measure the RMSE comparing the global model output (model of the sensor outputs of two different manipulated objects) and the actual sensor outputs.

(b) Classification with Control. Since we are able to accurately distinguish between two objects (Figure 7(a)), we select the corresponding individual model from the RBF network (see Figure 3(a)) to obtain the acceleration values of the object being manipulated instead of using a shared model with the data of the two objects. In fact, the individual models of the five objects were previously learnt. The RMSE is always lower when single object models are used (Figure 9) (after an object classification stage). Direct control using a global (shared) model achieves a lower performance because the applied model does not take into account the singularities of each of the individual models.

The *classification with control scheme* is feasible when dealing with objects easy to be discriminated and the actual trajectory of the movement follows more reliably the single object model than a "shared model." In Figure 9, the RMSE is plotted when *comparing sensor model response* and the *actual sensor responses*. Figure 8 shows the accuracy of the classification task when training the network with different numbers of objects (classification errors along the trajectory). Nevertheless, we have shown (Figure 7) that with the objects used in our study, the network can only reliably classify the object (along the whole trajectory) when we focus on distinguishing between two objects.

4. Conclusions

The first conclusion of the presented work is that the applied control scheme is critical to facilitate reliable dynamic model abstraction. In fact, the dynamic models or computed

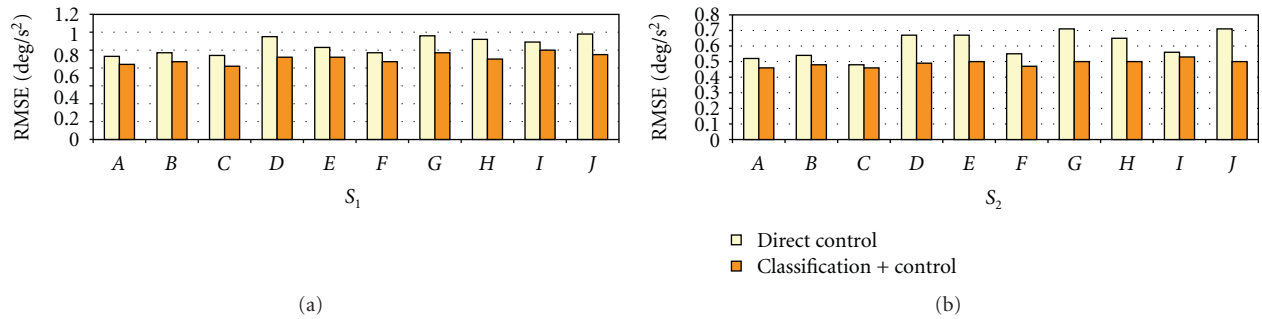


Figure 9: Comparison between the individual and the shared models related to predicted sensory values $S1$ and $S2$. The columns of a lighter colour indicate RMSE for the direct control while the columns of a darker colour represent RMSE for the controller of a single object once it is accurately classified. Cases (objects): A (0 and 1), B (0 and 2), C (0 and 3), D (0 and 4), E (1 and 2), F (1 and 3), G (1 and 4), H (2 and 3), I (2 and 4), and J (3 and 4).

torques are used for high performance and compliant robot control in terms of precision and energy efficiency [3, 65].

We have compared a dense position-driven versus a sparse velocity-driven control strategy. The second scheme leads to smoother movements when manipulating different objects. This produces more stable sensor responses which allow to model the dynamics of the movement. As model abstraction engine, we have used a well-known RBF network (widely used for function approximation tasks) and we compared its performance for abstracting dynamics models of the different objects using a sparse velocity-driven scheme (Figure 5(a)).

In a second part of the work, we have evaluated if the object classification is feasible. More concretely, we have checked out if we can perform it using sensorimotor complexes (motor commands, velocities, and local accelerations) obtained during object manipulation (exploration task) as inputs. Figure 7(a) shows that only the discrimination between any pair of two objects was possible. Robots with low gains will produce different actual movements when manipulating different objects, because these objects affect significantly their dynamic model. If the robot is controlled with high gains, the differences in the dynamic model of the robot are compensated by the high control gains. If the robot is tightly fixed but is controlled with low gains the actual movement (sensorimotor representation of an object manipulation) is different for each object (Figure 7(b)). If the robot is slightly fixed, the control commands produce vibrations that perturb the dynamic model and, therefore, the different sensorimotor representations are highly affected by these noisy artefacts (motor command driven vibrations) (see Figure 7(a)).

Finally, by assuming that we are able to classify objects during manipulation and to switch to the best dynamic model, we have evaluated the accuracy when comparing the movement's actual dynamics (actual sensor responses) with the abstracted ones. Furthermore, we have evaluated the gain in performance if we compare it with a specific dynamic model instead of a "global" dynamics model. We have evaluated the impact of this two-step control strategy obtaining an improvement of 30% (Figure 9) in predicting the sensor outputs along the trajectory. This abstracted model can be used in predicted control strategies or for stabilisation purposes.

Summarising, our results prove that the presented robot and artificial neural network can abstract dynamic models of objects within the stream sensorimotor primitives during manipulation tasks. One of the most important results of the work shown in Figures 6(a) (nonfixed robot) and 6(b) (fixed robot) indicates that robot platforms can highly benefit from nonrigid sparse velocity-driven control scheme. The high error obtained with the dense position-driven control scheme (Figure 6(a)) is caused by vibrations as the slight fixation of the robot to the ground is only supported by its own weight. Furthermore, when we try to abstract a movement model, the performance achieved by the neural network (RBF) also represents a measurement of the stability of the movements (the repeatability of such trajectories when applying different control schemes).

Acknowledgments

This work has been supported in part by the EU Grant SENSOPAC (FP6-IST-028056) and by the Spanish National Grants DPI-2004-07032 and TEC2010-15396.

References

[1] H. Hoffmann, S. Schaal, and S. Vijayakumar, "Local dimensionality reduction for non-parametric regression," *Neural Processing Letters*, vol. 29, no. 2, pp. 109–131, 2009.

[2] D. Nguyen-Tuong and J. Peters, "Learning robot dynamics for computed torque control using local Gaussian processes regression," in *Proceedings of the ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems (LAB-RS '08)*, pp. 59–64, Edinburgh, UK, August 2008.

[3] G. Petkos, M. Toussaint, and S. Vijayakumar, "Learning multiple models of nonlinear dynamics for control under varying contexts," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN '06)*, pp. 898–907, Athens, Greece, 2006.

[4] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[5] S. Vijayakumar, A. D'souza, T. Shibata, J. Conradt, and S. Schaal, "Statistical learning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 55–69, 2002.

[6] R. Miall, "Motor control, biological and theoretical," in *Handbook of Brain Theory and Neural Network*, M. Arbib, Ed., pp. 686–689, Bradford Books/MIT Press, Cambridge, Mass, USA, 2nd edition, 2003.

[7] S. Schaal and N. Schweighofer, "Computational motor control in humans and robots," *Current Opinion in Neurobiology*, vol. 15, no. 6, pp. 675–682, 2005.

[8] Sensopac, Eu project(SENSOrimotor structuring of Perception and Action for emergent Cognition), 2010, http://www.sensopac.org/.

[9] C. Q. Huang, S. J. Shi, X. G. Wang, and W. K. Chung, "Parallel force/position controllers for robot manipulators with uncertain kinematics," *International Journal of Robotics and Automation*, vol. 20, no. 3, pp. 158–167, 2005.

[10] J. Roy and L. L. Whitcomb, "Adaptive force control of position/velocity controlled robots: theory and experiment," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 2, pp. 121–137, 2002.

[11] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[12] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7-8, pp. 1317–1329, 1998.

[13] D. H. Shin and A. Ollero, "Mobile robot path planning for fine-grained and smooth path specifications," *Journal of Robotic Systems*, vol. 12, no. 7, pp. 491–503, 1995.

[14] M. H. Raibert and J. J. Craig, "Hybrid position/force control of manipulators," *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 103, no. 2, pp. 126–133, 1981.

[15] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, John Wiley & Sons, New York, NY, USA, 2006.

[16] W. Yu and M. A. Moreno-Armendariz, "Robust visual servoing of robot manipulators with neuro compensation," *Journal of the Franklin Institute*, vol. 342, no. 7, pp. 824–838, 2005.

[17] W. Yu and X. Li, "PD control of robot with velocity estimation and uncertainties compensation," *International Journal of Robotics and Automation*, vol. 21, no. 1, pp. 1–9, 2006.

[18] J. J. de Rubio and L. A. Soriano, "An asymptotic stable proportional derivative control with sliding mode gravity compensation and with a high gain observer for robotic arms," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 10, pp. 4513–4525, 2010.

[19] J. de Jesus Rubio, C. Torres, and C. Aguilar, "Optimal control based in a mathematical model applied to robotic arms," *International Journal of Innovative Computing, Information and Control*, vol. 7, no. 8, pp. 5045–5062, 2011.

[20] M. Haruno, D. M. Wolpert, and M. Kawato, "MOSAIC model for sensorimotor learning and control," *Neural Computation*, vol. 13, no. 10, pp. 2201–2220, 2001.

[21] B. Daya, "Multilayer perceptrons model for the stability of a bipedal robot," *Neural Processing Letters*, vol. 9, no. 3, pp. 221–227, 1999.

[22] S. Tolu, E. Ros, and R. Agís, "Bio-inspired control model for object manipulation by humanoid robots," in *Proceedings of the Proceedings of the 9th international work conference on Artificial neural networks (IWANN '07)*, pp. 822–829, San Sebastián, Spain, June 2007.

[23] R. R. Carrillo, E. Ros, C. Boucheny, and O. J. M. D. Coenen, "A real-time spiking cerebellum model for learning robot control," *BioSystems*, vol. 94, no. 1-2, pp. 18–27, 2008.

[24] N. R. Luque, J. A. Garrido, R. R. Carrillo, O. J.-M. D. Coenen, and E. Ros, "Cerebellar input configuration toward object model abstraction in manipulation tasks," *IEEE Transactions on Neural Networks*, vol. 22, no. 8, pp. 1321–1328, 2011.

[25] N. R. Luque, J. A. Garrido, R. R. Carrillo, O. J.-M. D. Coenen, and E. Ros, "Cerebellarlike corrective model inference engine for manipulation tasks," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 5, pp. 1299–1312, 2011.

[26] N. R. Luque, J. A. Garrido, R. R. Carrillo, S. Tolu, and E. Ros, "Adaptive cerebellar spiking model embedded in the control loop: context switching and robustness against noise," *International Journal of Neural Systems*, vol. 21, no. 5, pp. 385–401, 2011.

[27] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, Massachusetts Institute of Technology, Cambridge, Mass, USA, 1995.

[28] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems—a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[29] J. Constantin, C. Nasr, and D. Hamad, "Control of a robot manipulator and pendubot system using artificial neural networks," *Robotica*, vol. 23, no. 6, pp. 781–784, 2005.

[30] T. D'Silva and R. Miikkulainen, "Learning dynamic obstacle avoidance for a robot arm using neuroevolution," *Neural Processing Letters*, vol. 30, no. 1, pp. 59–69, 2009.

[31] C. C. Lee, P. C. Chung, J. R. Tsai, and C. I. Chang, "Robust radial basis function neural networks," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 29, no. 6, pp. 674–685, 1999.

[32] C. Chez, *In Principles of Neural Science*, Prentice Hall, London, UK, 3rd edition, 1991.

[33] M. Ito, "Mechanisms of motor learning in the cerebellum," *Brain Research*, vol. 886, no. 1-2, pp. 237–245, 2000.

[34] C. Assad, S. Trujillo, S. Dastoor, and L. Xu, "Cerebellar dynamic state estimation for a biomorphic robot arm," in *Proceedings of the International Conference on Systems, Man and Cybernetics*, pp. 877–882, Waikoloa, Hawaii, USA, October 2005.

[35] M. Ito, "Control of mental activities by internal models in the cerebellum," *Nature Reviews Neuroscience*, vol. 9, no. 4, pp. 304–313, 2008.

[36] P. Van Der Smagt, "Cerebellar control of robot arms," *Connection Science*, vol. 10, no. 3-4, pp. 301–320, 1998.

[37] R. C. Miall, J. G. Keating, M. Malkmus, and W. T. Thach, "Simple spike activity predicts occurrence of complex spikes in cerebellar Purkinje cells," *Nature neuroscience*, vol. 1, no. 1, pp. 13–15, 1998.

[38] R. Apps and M. Garwicz, "Anatomical and physiological foundations of cerebellar information processing," *Nature Reviews Neuroscience*, vol. 6, no. 4, pp. 297–311, 2005.

[39] Ikarus, "Acceleration sensors," 2011, http://www.ikarus-modellbau.de.

[40] V. J. Lumelsky, M. S. Shur, and S. Wagner, "Sensitive skin," *IEEE Sensors Journal*, vol. 1, no. 1, pp. 41–51, 2001.

[41] C. J. Khoh and K. K. Tan, "Adaptive robust control for servo manipulators," *Neural Computing and Applications*, vol. 12, no. 3-4, pp. 178–184, 2003.

[42] M. Zhihong, X. H. Yu, and H. R. Wu, "An RBF neural network-based adaptive control for SISO linearisable nonlinear systems," *Neural Computing and Applications*, vol. 7, no. 1, pp. 71–77, 1998.

[43] Celoxica Inc, 2011, http://www.celoxica.com/.

[44] Hitec Robotics Inc, 2011, http://www.robonova.de/store/home.php.

[45] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 101–119, 1986.

[46] D. Kubus, T. Kröger, and F. M. Wahl, "On-line rigid object recognition and pose estimation based on inertial parameters," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 1402–1408, San Diego, Calif, USA, November 2007.

[47] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, Pearson/Prentice Hall, Upper Saddle River, NJ, USA, 3rd edition, 2005.

[48] M. Krabbes, C. Doschner et al., "Modelling of robot dynamics based on a multidimensional rbf-like neural network," in *Proceedings of the International Conference on Information Intelligence and Systems (ICIIS'99)*, pp. 180–187, Bethesda, Md, USA, October/November 1999.

[49] R. J. Shilling, *Fundamentals of Robotics: Analysis and Control*, Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[50] P. Angelov, "Fuzzily connected multimodel systems evolving autonomously from data streams," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 41, no. 4, pp. 898–910, 2011.

[51] P. Angelov and D. Filev, "Simpl_eTS: a simplified method for learning evolving Takagi-Sugeno fuzzy models," in *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1068–1073, Reno, Nev, USA, May 2005.

[52] G. P. Liu, V. Kadirkamanathan, and S. A. Billings, "Variable neural networks for adaptive control of nonlinear systems," *IEEE Transactions on Systems, Man and Cybernetics C*, vol. 29, no. 1, pp. 34–43, 1999.

[53] H. J. Rong, N. Sundararajan, G. B. Huang, and P. Saratchandran, "Sequential Adaptive Fuzzy Inference System (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[54] J. de Jesús Rubio, D. M. Vázquez, and J. Pacheco, "Backpropagation to train an evolving radial basis function neural network," *Evolving Systems*, vol. 1, no. 3, pp. 173–180, 2010.

[55] J. J. Rubio, F. Ortiz-Rodriguez, C. Mariaca-Gaspar, and J. Tovar, "A method for online pattern recognition of abnormal-eye movements," *Neural Computing & Applications*. In press.

[56] D. K. Wedding and A. Eltimsahy, "Flexible link control using multiple forward paths, multiple RBF neural networks in a direct control application," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 2619–2624, Nashville, Tenn, USA, October 2000.

[57] Z. D. Wang, E. Nakano, and T. Takahashi, "Solving function distribution and behavior design problem for cooperative object handling by multiple mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 33, no. 5, pp. 537–549, 2003.

[58] J. Flanagan, K. Merritt, and R. Johansson, "Predictive mechanisms and object representations used in object manipulation," in *Sensorimotor Control of Grasping: Physiology and Pathophysiology*, pp. 161–177, Cambridge University Press, Cambridge, UK, 2009.

[59] C. C. Pagano, J. M. Kinsella-Shaw, P. E. Cassidy, and M. T. Turvey, "Role of the inertia tensor in haptically perceiving where an object is grasped," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 20, no. 2, pp. 276–285, 1994.

[60] J. S. Bay, "Fully autonomous active sensor-based exploration concept for shape-sensing robots," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 4, pp. 850–860, 1991.

[61] G. Petkos and S. Vijayakumar, "Load estimation and control using learned dynamics models," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '07)*, pp. 1527–1532, San Diego, Calif, USA, November 2007.

[62] Mathworks Inc, 2011, http://www.mathworks.com/products/matlab.

[63] M. Orr, J. Hallam, K. Takezawa et al., "Combining regression trees and radial basis function networks," *International Journal of Neural Systems*, vol. 10, no. 6, pp. 453–465, 2000.

[64] S. Chen, E. S. Chng, and K. Alkadhimi, "Regularized orthogonal least squares algorithm for constructing radial basis function networks," *International Journal of Control*, vol. 64, no. 5, pp. 829–837, 1996.

[65] D. Nguyeii-Tuoiig, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *Proceedings of the American Control Conference (ACC '08)*, pp. 212–217, Seattle, Wash, USA, June 2008.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 545731, 9 pages
doi:10.1155/2012/545731

Research Article

An Output-Recurrent-Neural-Network-Based Iterative Learning Control for Unknown Nonlinear Dynamic Plants

Ying-Chung Wang and Chiang-Ju Chien

Department of Electronic Engineering, Huafan University, Shihding, New Taipei City 223, Taiwan

Correspondence should be addressed to Chiang-Ju Chien, cjc@cc.hfu.edu.tw

Received 31 July 2011; Revised 9 November 2011; Accepted 1 December 2011

Academic Editor: Isaac Chairez

Copyright © 2012 Y.-C. Wang and C.-J. Chien. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We present a design method for iterative learning control system by using an output recurrent neural network (ORNN). Two ORNNs are employed to design the learning control structure. The first ORNN, which is called the output recurrent neural controller (ORNC), is used as an iterative learning controller to achieve the learning control objective. To guarantee the convergence of learning error, some information of plant sensitivity is required to design a suitable adaptive law for the ORNC. Hence, a second ORNN, which is called the output recurrent neural identifier (ORNI), is used as an identifier to provide the required information. All the weights of ORNC and ORNI will be tuned during the control iteration and identification process, respectively, in order to achieve a desired learning performance. The adaptive laws for the weights of ORNC and ORNI and the analysis of learning performances are determined via a Lyapunov like analysis. It is shown that the identification error will asymptotically converge to zero and repetitive output tracking error will asymptotically converge to zero except the initial resetting error.

1. Introduction

Iterative learning control (ILC) system has become one of the most effective control strategies in dealing with repeated tracking control of nonlinear plants. The ILC system improves the control performance by a self-tuning process in the traditional PID-type ILC algorithms for linear plants or affine nonlinear plants with nonlinearities satisfying global Lipschitz continuous condition [1–3]. Recently, the ILC strategies combined with other control methodologies such as observer-based iterative learning control [4], adaptive iterative learning control [5], robust iterative learning control [6], or adaptive robust iterative learning control [7], have been widely studied in order to extend the applications to more general class of nonlinear systems. However, more and more restrictions are required in theory to develop these learning controllers. Among these ILC algorithms, PID-type ILC algorithms are still attractive to engineers since they are simple and effective for real implementations and industry applications. A main problem of the PID-type ILC algorithms is that a sufficient condition required to guarantee learning stability and convergence will depend on plant's input/output coupling function (matrix). In general, it is hard to design the learning gain if the nonlinear dynamic plant is highly nonlinear and unknown. In order to get the input/output coupling function (matrix), the ILC using a neural or fuzzy system to solve the learning gain implementation problem can be found in [8, 9]. A neural network or a fuzzy system was used to approximate the inverse of plant's input/output coupling function (matrix). The inverse function (matrix) is claimed to be an optimal choice of the learning gain from a convergent condition point of view. As the nonlinear system is assumed to be unknown, some offline adaptive mechanisms are applied to update the network parameters in order to approximate the ideal optimal learning gain.

Actually, for control of unknown nonlinear systems, neural-network-based controller has become an important strategy in the past two decades. Multilayer neural networks, recurrent neural networks and dynamic neural network [10–16] were used for the design of adaptive controllers. On the other hand, fuzzy logic system, fuzzy neural network,

recurrent fuzzy neural networks and dynamic fuzzy neural network were also a popular tool for the design of adaptive controllers [17–22]. These concepts have also been applied to the design of adaptive iterative learning control of nonlinear plants [23–25]. However, few ILC works were developed for general unknown nonlinear plants, especially nonaffine nonlinear plants. As the authors can understand, a real-time recurrent network (RTRN) was developed in [26] for real-time learning control of general unknown nonlinear plants. But unfortunately, their learning algorithm depends on the generalized inverse of weight matrix in the RTRN. If the generalized inverse of weight matrix does not exist, the learning control scheme is not implementable.

In this paper, we consider the design of an iterative learning controller for a class of unknown nonlinear dynamic plants. Motivated by our previous work in [27], an improved version of an identifier-based iterative learning controller is proposed by using an output recurrent neural network (ORNN). Two ORNNs are used to design an ORNN-based iterative learning control system. The proposed ORNN-based ILC system includes an ORNN controller (ORNC) and an ORNN identifier (ORNI). The ORNC is used as an iterative learning controller to achieve the repetitive tracking control objective. The weights of ORNC are tuned via adaptive laws determined by a Lyapunov-like analysis. In order to realize the adaptive laws and guarantee the convergence of learning error, some information of the unknown plant sensitivity is required for the design of adaptive laws. Hence, the ORNI is then applied as an identifier to provide the required information from plant sensitivity. In a similar way, the weights of ORNI are tuned via some adaptive laws determined by a Lyapunov-like analysis. Both of the proposed ORNC and ORNI update their network weights along the control iteration and identification process, respectively. This ORNN-based ILC system can be used to execute a repetitive control task of a general nonlinear plant. It is shown that the identification error will asymptotically converge to zero and repetitive output tracking error will asymptotically converge to zero except the initial resetting error.

This paper is organized as follows. The structure of ORNN is introduced in Section 2. In Section 3, we present the design of ORNC and ORNI for the ORNN-based ILC system. The adaptation laws are derived and the learning performance is guaranteed based on a Lyapunov-like analysis. To illustrate the effectiveness of the proposed ILC system, a numerical example is used in Section 4 for computer simulation. Finally a conclusion is made in Section 5.

In the subsequent discussions, the following notations will be used in all the sections.

(i) $|z|$ denotes the absolute value of a function z.

(ii) $\|v\| = \sqrt{(v^\top v)}$ denotes the usual Euclidean norm of a vector $v = [v_1, \ldots, v_n]^\top \in \mathbb{R}^n$.

(iii) $\|A\| = \max_{1\le i\le n}\{\sum_{j=1}^{m} |a_{ij}|\}$ denotes the norm of a matrix $A = \{a_{ij}\} \in \mathbb{R}^{n\times m}$.

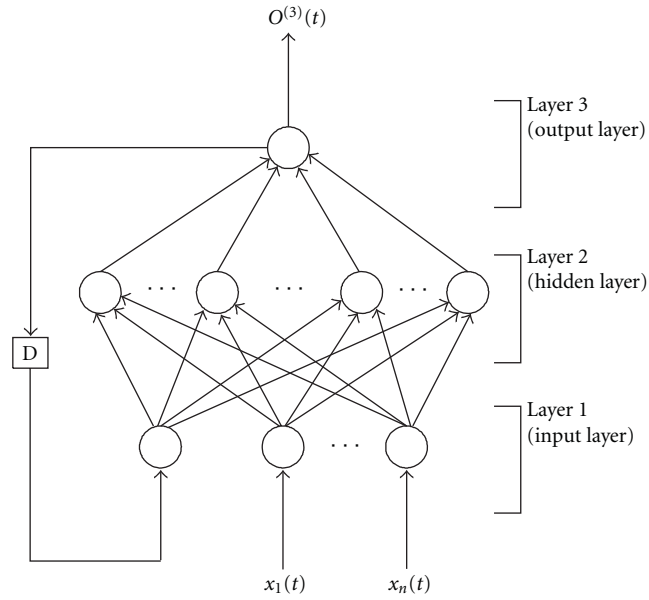


Figure 1: Structure of the ORNN.

2. The Output Recurrent Neural Network

In this paper, two ORNNs are used to design an iterative learning control system. The structure of the ORNN is shown in Figure 1, which comprises an input layer, a hidden layer, and an output layer.

(i) Layer 1 (Input Layer): Each node in this layer represents an input variable, which only transmits input value to the next layer directly. For the ith input node, $i = 1, \ldots, n+1$,

$$\begin{aligned} \text{net}_i^{(1)} &= \begin{cases} x_i, & i = 1, \ldots, n \\ \mathbf{D}\left[O^{(3)}\right], & i = n+1 \end{cases} \\ O_i^{(1)} &= f_i^{(1)}\left(\text{net}_i^{(1)}\right) = \text{net}_i^{(1)}, \end{aligned} \tag{1}$$

where x_i, $i = 1, \ldots, n$ represents the ith external input signal to the ith node of layer 1, and $\mathbf{D}[O^{(3)}]$ denotes the delay of ORNN output $O^{(3)}$ whcih can be further defined as $x_{n+1} = \mathbf{D}[O^{(3)}]$.

(ii) Layer 2 (Hidden Layer): Each node in this layer performs an activation function whose inputs come from input layer. For the ℓth hidden node, a sigmoid function is adopted here such that the ℓth node, $\ell = 1, \ldots, M$ will be represented as

$$\begin{aligned} \text{net}_\ell^{(2)} &= \sum_{i=1}^{n+1} V_{i\ell} x_i^{(2)}, \\ O_\ell^{(2)} &= f_\ell^{(2)}\left(\text{net}_\ell^{(2)}\right) = \frac{1}{1+\exp\left(-\text{net}_\ell^{(2)}\right)}, \end{aligned} \tag{2}$$

where $x_i^{(2)} = O_i^{(1)}$, $V_{i\ell}$ is the connective weight between the input layer and the hidden layer, M is the number of neuron in the hidden layer.

(iii) Layer 3 (Output Layer): Each node in this layer represents an output node, which computes the overall output as the summation of all input signals from hidden layer. For the output node,

$$\begin{aligned} \text{net}^{(3)} &= \sum_{\ell=1}^{M} w_\ell \cdot x_\ell^{(3)}, \\ O^{(3)} &= f^{(3)}\left(\text{net}^{(3)}\right) = \text{net}^{(3)}, \end{aligned} \tag{3}$$

where $x_\ell^{(3)} = O_\ell^{(2)}$ and w_ℓ is the connective weight between the hidden layer and the output layer.

Let n denotes the dimension of input vector $X = [x_1, \ldots, x_n]^\top \in \mathbb{R}^{n\times 1}$ of nonlinear function $f(X)$ and M denotes the number of neurons in the hidden layer, the ORNN which performs as an approximator of the nonlinear function $f(X)$ is now described in a matrix form as follows:

$$O^{(3)}\left(\mathbf{D}\left[O^{(3)}\right], X, W, V\right) = W^\top O^{(2)}(V^\top X_a), \tag{4}$$

where $W \in \mathbb{R}^{M\times 1}$ and $V \in \mathbb{R}^{(n+1)\times M}$ are output-hidden wight matrix and hidden-input weight matrix, respectively, $X \in \mathbb{R}^{n\times 1}$ is the external input vector, $X_a \equiv [X^\top, \mathbf{D}[O^{(3)}]]^\top \in \mathbb{R}^{(n+1)\times 1}$ is the augmented neural input vector, and $\mathbf{D}[O^{(3)}]$ denotes the delay of ORNN output $O^{(3)}$. The activation function vector is defined as $O^{(2)}(V^\top X_a) \equiv [O_1^{(2)}(V_1^\top X_a), \ldots, O_M^{(2)}(V_M^\top X_a)]^\top \in \mathbb{R}^{M\times 1}$ where $V = [V_1, V_2, \ldots, V_M]$ with $V_\ell \in \mathbb{R}^{(n+1)\times 1}$ being the ℓth column vector, and $O_\ell^{(2)}(V_\ell^\top X_a) \equiv 1/(1+\exp(-V_\ell^\top X_a)) \in \mathbb{R}$, $\ell = 1, \ldots, M$ is a sigmoid function.

3. Design of Output-Recurrent-Neural- Network-Based Iterative Learning Control System

In this paper, we consider an unknown nonlinear dynamic plant which can perform a given task repeatedly over a finite time sequence $t = \{0, \ldots, N\}$ as follows:

$$y^j(t+1) = f\left(y^j(t), \ldots, y^j(t-n+1), u^j(t)\right), \tag{5}$$

where $j \in \mathcal{Z}_+$ denotes the index of control iteration number and $t = \{0, \ldots, N\}$ denotes the time index. The signals $y^j(t)$ and $u^j(t) \in \mathbb{R}$ are the system output and input, respectively. $f : \mathbb{R}^{n+1} \to \mathbb{R}$ is the unknown continuous function, n represents the respective output delay order. Given a specified desired trajectory $y_d(t)$, $t \in \{0, \ldots, N\}$, the control objective is to design an output-recurrent-neural-network-based iterative learning control system such that when control iteration number j is large enough, $|y_d(t) - y^j(t)|$ will converge to some small positive error tolerance bounds for all $t \in \{0, \ldots, N\}$ even if there exists an initial resetting error. Here the initial resetting error means that $y_d(0) \neq y^j(0)$ for all $j \geq 1$. To achieve the control objective, an iterative learning control system based on ORNN design is proposed in Figure 2. In this figure, D denotes the delay in time domain and M denotes the memory in control iteration domain.

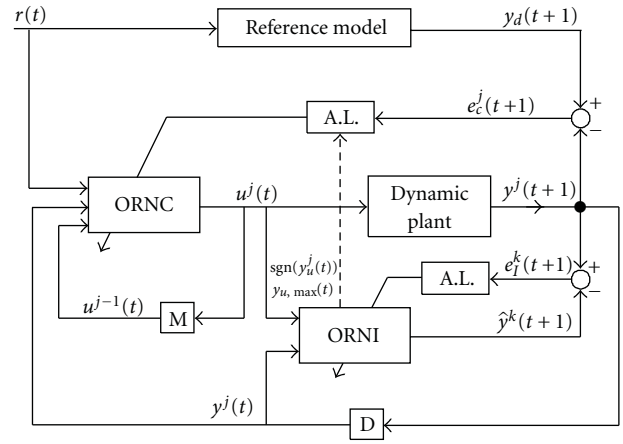


Figure 2: Block diagram of the ORNN-based ILC system.

Before we state the design steps of the proposed control structure, some assumptions on the unknown nonlinear system and desired trajectories are given as follows.

(A1) The nonlinear dynamic plant is a relaxed system whose input $u^j(t)$ and output $y^j(t)$ are related by $y^j(t) = 0$ for all $t \in \{-\infty, \ldots, -1\}$.

(A2) There exists a bounded unknown upper bounding function $y_{u,\max}(t) = \max_{\forall j}|y_u^j(t)|$ such that $0 < |y_u^j(t)| \leq y_{u,\max}(t)$, where the factor $y_u^j(t) = \partial y^j(t+1)/\partial u^j(t)$ represents the sensitivity of the plant with respect to its input.

(A3) The reference model is designed to generate the bounded desired trajectory $y_d(t+1) = f_d(y_d(t), \ldots, y_d(t-n+1), r(t))$, which is based on a specified bounded reference input $r(t)$ with $f_d : \mathbb{R}^{n+1} \to \mathbb{R}$ being a continuous function.

The design of the ORNN-based iterative learning control system is divided into two parts.

3.1. Part 1: Design of ORNC and Corresponding Adaptive Laws. Based on the assumptions on the nonlinear plant (5), we define a tracking error $e_c^j(t)$ at jth control iteration as follows:

$$e_c^j(t) = y_d(t) - y^j(t). \tag{6}$$

It is noted that there exist bounded constants ε_c^j, $j \in Z_+$ such that the initial value of $e_c^j(t)$ will satisfy

$$\left|e_c^j(0)\right| = \varepsilon_c^j. \tag{7}$$

The difference of $e_c^j(t)$ between two successive iterations can be computed as [28]

$$\begin{aligned}\Delta e_c^j(t+1) &= e_c^{j+1}(t+1) - e_c^j(t+1)\\ &= -\left(y^{j+1}(t+1) - y^j(t+1)\right)\\ &\approx -\frac{\partial y^j(t+1)}{\partial u^j(t)}\left(u^{j+1}(t) - u^j(t)\right)\\ &\equiv -y_u^j(t)\left(u^{j+1}(t) - u^j(t)\right).\end{aligned} \tag{8}$$

The ORNN is used to design an ORNC in order to achieve the iterative learning control objective. Let n_c be the dimension of the external input vector $X_c^j(t) = [r(t), y^j(t), u^{j-1}(t),]^\top \in \mathbb{R}^{n_c\times 1}$ and M_c denote the number of neurons in hidden layer of the ORNC. The ORNC which performs as an iterative learning controller is described in a matrix form as follows:

$$\begin{aligned}u^j(t) &= O_c^{(3)}\left(\mathbf{D}\left[O_c^{(3)^j}(t)\right], X_c^j(t), W_c^j(t), V_c^j(t)\right)\\ &= W_c^j(t)^\top O_c^{(2)}\left(V_c^j(t)^\top X_{ca}^j(t)\right),\end{aligned} \tag{9}$$

where $W_c^j(t) \in \mathbb{R}^{M_c\times 1}$ and $V_c^j(t) \in \mathbb{R}^{(n_c+1)\times M_c}$ are output-hidden wight matrix and hidden-input weight matrix to be tuned via some suitable adaptive laws, respectively, and $X_{ca}^j(t) \equiv [X_c^j(t)^\top, \mathbf{D}[O_c^{(3)j}(t)]]^\top \in \mathbb{R}^{(n_c+1)\times 1}$ is the augmented neural input vector. For the sake of convenience, we define $O_c^{(2)}(V_c^j(t)^\top X_{ca}^j(t)) \equiv O_c^{(2)^j}(t)$. Now substituting (9) into (8), we will have

$$\Delta e_c^j(t+1) = -y_u^j(t)\left(W_c^{j+1}(t)^\top O_c^{(2)^{j+1}}(t) - W_c^j(t)^\top O_c^{(2)^j}(t)\right). \tag{10}$$

For simplicity, we define $\Delta X_{ca}^j(t) = X_{ca}^{j+1}(t) - X_{ca}^j(t)$, $\Delta W_c^j(t) = W_c^{j+1}(t) - W_c^j(t)$, $\Delta V_c^j(t) = V_c^{j+1}(t) - V_c^j(t)$. After adding and subtracting $W_c^j(t)^\top O_c^{(2)^{j+1}}(t)$ to (10), we can find that

$$\begin{aligned}\Delta e_c^j(t+1) = &-y_u^j(t)\Delta W_c^j(t)^\top O_c^{(2)^{j+1}}(t)\\ &- y_u^j(t) W_c^j(t)^\top\left(O_c^{(2)^{j+1}}(t) - O_c^{(2)^j}(t)\right).\end{aligned} \tag{11}$$

Investigating the second term in the right hand side of (11) by using the mean-value theorem, we have

$$\begin{aligned}&O_c^{(2)^{j+1}}(t) - O_c^{(2)^j}(t)\\ &\quad= O_c^{(2)'^j}(t)\left(V_c^{j+1}(t)^\top X_{ca}^{j+1}(t) - V_c^j(t)^\top X_{ca}^j(t)\right)\\ &\quad= O_c^{(2)'^j}(t)\left(V_c^{j+1}(t)^\top X_{ca}^{j+1}(t) - V_c^j(t)^\top X_{ca}^{j+1}(t)\right.\\ &\qquad\left.+V_c^j(t)^\top X_{ca}^{j+1}(t) - V_c^j(t)^\top X_{ca}^j(t)\right)\\ &\quad= O_c^{(2)'^j}(t)\left(\Delta V_c^j(t)^\top X_{ca}^{j+1}(t) + V_c^j(t)^\top \Delta X_{ca}^j(t)\right),\end{aligned} \tag{12}$$

where $O_c^{(2)'^j}(t) = \mathrm{diag}[O_{c,1}^{(2)'^j}(t), \ldots, O_{c,M_c}^{(2)'^j}(t)] \in \mathbb{R}^{M_c\times M_c}$ with $O_{c,l}^{(2)'^j}(t) \equiv dO_{c,\ell}^{(2)^j}(Z_{c,\ell}(t))/dZ_{c,\ell}(t)|_{Z_{c,\ell}(t)}$, $Z_{c,\ell}(t)$ has a value between $V_{c,\ell}^{j+1}(t)^\top X_{ca}^{j+1}(t)$ and $V_{c,\ell}^j(t)^\top X_{ca}^j(t)$, $\ell = 1, \ldots, M_c$.

Now if we substitute (12) into (11), we will have

$$\begin{aligned}\Delta e_c^j(t+1) = &-y_u^j(t)\Delta W_c^j(t)^\top O_c^{(2)^{j+1}}(t)\\ &- y_u^j(t) W_c^j(t)^\top O_c^{(2)'^j}(t)\\ &\times\left(\Delta V_c^j(t)^\top X_{ca}^{j+1}(t) + V_c^j(t)^\top \Delta X_{ca}^j(t)\right).\end{aligned} \tag{13}$$

The adaptation algorithms for weights $W_c^{j+1}(t)$ and $V_c^{j+1}(t)$ of ORNC at (next) $j+1$th control iteration to guarantee the error convergence are given as follows:

$$W_c^{j+1}(t) = W_c^j(t) + \frac{\mathrm{sgn}\left(y_u^j(t)\right) e_c^j(t+1) O_c^{(2)^{j+1}}(t)}{y_{u,\max}(t) M_c}, \tag{14}$$

$$V_c^{j+1}(t) = V_c^j(t) - \frac{X_{ca}^{j+1}(t)\Delta X_{ca}^j(t)^\top V_c^j(t)}{\left\|X_{ca}^{j+1}(t)\right\|^2}, \tag{15}$$

where $y_{u,\max}(t)$ is defined in assumption (A2). If we substitute adaptation laws (14) and (15) into (13), we can find that

$$\begin{aligned}e_c^{j+1}(t+1) = &\, e_c^j(t+1) - e_c^j(t+1)\\ &\times \frac{\left|y_u^j(t)\right| O_c^{(2)^{j+1}}(t)^\top O_c^{(2)^{j+1}}(t)}{y_{u,\max}(t) M_c}.\end{aligned} \tag{16}$$

Theorem 1. *Consider the nonlinear plant* (5) *which satisfies assumptions (A1)–(A3). The proposed ORNC* (9) *and adaptation laws* (14) *and* (15) *will ensure the asymptotic convergence of tracking error as control iteration approaches infinity.*

Proof. Let us choose a discrete-type Lyapunov function as

$$E_c^j(t+1) = \frac{1}{2}\left(e_c^j(t+1)\right)^2, \tag{17}$$

then the change of Lyapunov function is

$$\begin{aligned}\Delta E_c^j(t+1) &= E_c^{j+1}(t+1) - E_c^j(t+1)\\ &= \frac{1}{2}\left[\left(e_c^{j+1}(t+1)\right)^2 - \left(e_c^j(t+1)\right)^2\right].\end{aligned} \tag{18}$$

Taking norms on (16), it yields

$$\begin{aligned}\left|e_c^{j+1}(t+1)\right| &= \left|e_c^j(t+1)\left[1 - \frac{\left|y_u^j(t)\right|\left\|O_c^{(2)^{j+1}}(t)\right\|^2}{y_{u,\max}(t) M_c}\right]\right|\\ &< \left|e_c^j(t+1)\right|\end{aligned} \tag{19}$$

for iteration $j \geq 1$. This further implies that $E_c^j(t+1) > 0$, $\Delta E_c^j(t+1) < 0$, for all $t \in \{0, \ldots, N\}$ for $j \geq 1$. Using Lyapunov stability of $E_c^j(t+1) > 0$, $\Delta E_c^j(t+1) < 0$ and (7), the tracking error $e_c^j(t)$ will satisfy

$$\lim_{j\to\infty}\left|e_c^j(t)\right| = \begin{cases}\varepsilon_c^\infty, & t = 0\\ 0, & t \neq 0.\end{cases} \tag{20}$$

This proves Theorem 1. □

Remark 2. If the plant sensitivity $y_u^j(t)$ is completely known so that $\operatorname{sgn}(y_u^j(t))$ and $y_{u,\max}(t)$ are available, then the control objective can be achieved by using the adaptation algorithms (14) and (15). However, the plant sensitivity $y_u^j(t)$ is in general unknown or only partially known. In part 2, we will design an ORNN-based identifier (ORNI) to estimate the unknown plant sensitivity $y_u^j(t)$ and then provide the sign function and upper bounding function of $y_u^j(t)$ for adaptation algorithms of ORNC.

3.2. Part 2: Design of ORNI and Corresponding Adaptive Laws. After each control iteration, the ORNI subsequently begins to perform identification process. The trained ORNI will then provide the approximated plant sensitivity to the ORNC to start the next control iteration. We would like to emphasize that the ORNI only identifies the nonlinear plant after each control iteration. This concept is quite different from traditional control tasks [29] and very important to the proposed ORNN-based ILC structure.

The structure of ORNN is further applied to design an ORNI to identify the nonlinear plant after the jth control iteration. The identification process is stated as follows. After each trial of controlling the nonlinear system, we collect the input output data $u^j(t)$ and $y^j(t)$, $t = 0, 1, \ldots, N+1$ as the training data for the identifier. When discussing the identification, we omit the control iteration index j and introduce a new identification iteration index $k \in Z_+$ to represent the number of identification process. That is, the notation for the training data $u^j(t)$, $y^j(t)$ and the ORNI output $\hat{y}^{j,k}(t)$ are simplified as $u(t)$, $y(t)$, and $\hat{y}^k(t)$, respectively. For the ORNI, let n_I be the dimension of external input vector $X_I(t) = [u(t), y(t)]^\top \in \mathbb{R}^{n_I \times 1}$ and M_I denote the number of neurons in hidden layer of the ORNI. The ORNI which performs as an iterative learning identifier for nonlinear plant (5) is now described in a matrix form as follows:

$$\begin{aligned}\hat{y}^k(t+1) &= O_I^{(3)}\left(\mathbf{D}\left[O_I^{(3)^k}(t)\right], X_{Ia}^k(t), W_I^k(t), V_I^k(t)\right)\\ &= W_I^k(t)^\top O_I^{(2)}\left(V_I^k(t)^\top X_{Ia}^k(t)\right),\end{aligned} \tag{21}$$

where $W_I^k(t) \in \mathbb{R}^{M_I \times 1}$ and $V_I^k(t) \in \mathbb{R}^{(n_I+1)\times M_I}$ are output-hidden wight matrix and hidden-input weight matrix to be tuned via some suitable adaptive laws, respectively, and $X_{Ia}^k(t) \equiv [X_I(t)^\top, \mathbf{D}[O_I^{(3)k}(t)]]^\top \in \mathbb{R}^{(n_I+1)\times 1}$ is the augmented neural input vector. For the sake of convenience, we define $O_I^{(2)}(V_I^k(t)^\top X_{Ia}^k(t)) \equiv O_I^{(2)^k}(t)$.

Based on the assumptions on the nonlinear plant (5), we define an identification error $e_I^k(t)$ at kth identification process as follows:

$$e_I^k(t) = y(t) - \hat{y}^k(t). \tag{22}$$

The difference of $e_I^k(t)$ between two successive identification process can be computed as

$$\begin{aligned}\Delta e_I^k(t+1) &= e_I^{k+1}(t+1) - e_I^k(t+1)\\ &= -\left(\hat{y}^{k+1}(t+1) - \hat{y}^k(t+1)\right).\end{aligned} \tag{23}$$

Now substituting (21) into (23), we will have

$$\Delta e_I^k(t+1) = -\left(W_I^{k+1}(t)^\top O_I^{(2)^{k+1}}(t) - W_I^k(t)^\top O_I^{(2)^k}(t)\right). \tag{24}$$

For simplicity, we define $\Delta X_{Ia}^k(t) = X_{Ia}^{k+1}(t) - X_{Ia}^k(t)$, $\Delta W_I^k(t) = W_I^{k+1}(t) - W_I^k(t)$, $\Delta V_I^k(t) = V_I^{k+1}(t) - V_I^k(t)$. After adding and subtracting $W_I^k(t)^\top O_I^{(2)^{k+1}}(t)$ to (24), we can find

$$\begin{aligned}\Delta e_I^k(t+1) = &-\Delta W_I^k(t)^\top O_I^{(2)^{k+1}}(t)\\ &- W_I^k(t)^\top\left(O_I^{(2)^{k+1}}(t) - O_I^{(2)^k}(t)\right).\end{aligned} \tag{25}$$

Investigating the second term in the right hand side of (25) by using the mean-value theorem, we can derive

$$\begin{aligned}&O_I^{(2)^{k+1}}(t) - O_I^{(2)^k}(t)\\ &\quad= O_I^{(2)'^k}(t)\left(V_I^{k+1}(t)^\top X_{Ia}^{k+1}(t) - V_I^k(t)^\top X_{Ia}^k(t)\right)\\ &\quad= O_I^{(2)'^k}(t)\left(V_I^{k+1}(t)^\top X_{Ia}^{k+1}(t) - V_I^k(t)^\top X_{Ia}^{k+1}(t)\right.\\ &\qquad\left.+ V_I^k(t)^\top X_{Ia}^{k+1}(t) - V_I^k(t)^\top X_{Ia}^k(t)\right)\\ &\quad= O_I^{(2)'^k}(t)\left(\Delta V_I^k(t)^\top X_{Ia}^{k+1}(t) + V_I^k(t)^\top \Delta X_{Ia}^k(t)\right),\end{aligned} \tag{26}$$

where $O_I^{(2)'^k}(t) = \operatorname{diag}[O_{I,1}^{(2)'^k}(t), \ldots, O_{I,M_I}^{(3)'^k}(t)] \in \mathbb{R}^{M_I \times M_I}$ with $O_{I,\ell}^{(2)'^k}(t) \equiv dO_{I,\ell}^{(2)^k}(Z_{I,\ell}(t))/dZ_{I,\ell}(t)|_{Z_{I,\ell}(t)}$, $Z_{I,\ell}(t)$ has a value between $V_{I,\ell}^{k+1}(t)^\top X_{Ia}^{k+1}(t)$ and $V_{I,\ell}^j(t)^\top X_{Ia}^k(t)$, $\ell = 1, \ldots, M_I$.

Now if we substitute (26) into (25), we will have

$$\begin{aligned}\Delta e_I^k(t+1) = &-\Delta W_I^k(t)^\top O_I^{(2)^{k+1}}(t)\\ &- W_I^k(t)^\top O_I^{(2)'^k}(t)\\ &\times\left(\Delta V_I^k(t)^\top X_{Ia}^{k+1}(t) + V_I^k(t)^\top \Delta X_{Ia}^k(t)\right).\end{aligned} \tag{27}$$

The adaptation algorithms for weights $W_I^{k+1}(t)$ and $V_I^{k+1}(t)$ of ORNI at (next) $k+1$th identification process are given as follows:

$$\begin{aligned}W_I^{k+1}(t) &= W_I^k(t) + \frac{e_I^k(t+1)O_I^{(2)^{k+1}}(t)}{M_I}\\ V_I^{k+1}(t) &= V_I^k(t) - \frac{X_{Ia}^{k+1}(t)\Delta X_{Ia}^k(t)^\top V_I^k(t)}{\left\|X_{Ia}^{k+1}(t)\right\|^2}.\end{aligned} \tag{28}$$

If we substitute adaptation laws (28) into (27), we have

$$e_I^{k+1}(t+1) = e_I^k(t+1) - e_I^k(t+1)\frac{O_I^{(2)^{k+1}}(t)^\top O_I^{(2)^{k+1}}(t)}{M_I}. \tag{29}$$

Theorem 3. *Consider the nonlinear dynamic plant* (5) *which satisfies assumptions (A1)–(A3). The proposed ORNI* (21) *and adaptation laws* (28) *will ensure that the asymptotic convergence of identification error is guaranteed as the numbers of identification approach infinity.*

Proof. Let us choose a discrete-type Lyapunov function as

$$E_I^k(t+1) = \frac{1}{2}\left(e_I^k(t+1)\right)^2, \quad \forall t \in \{0,\ldots,N\}, \tag{30}$$

then we can derive the change of Lyapunov function as

$$\begin{aligned}\Delta E_I^k(t+1) &= E_I^{k+1}(t+1) - E_I^k(t+1)\\ &= \frac{1}{2}\left[\left(e_I^{k+1}(t+1)\right)^2 - \left(e_I^k(t+1)\right)^2\right].\end{aligned} \tag{31}$$

Taking norms on (29), we have

$$\begin{aligned}\left|e_I^{k+1}(t+1)\right| &= \left|e_I^k(t+1)\right|\left|1 - \frac{\left\|O_I^{(2)^{k+1}}(t)\right\|^2}{M_I}\right|\\ &< \left|e_I^k(t+1)\right|\end{aligned} \tag{32}$$

for iteration $k \geq 1$. This implies that $E_I^k(t+1) > 0$, $\Delta E_I^k(t+1) < 0$, for all $t \in \{0,\ldots,N\}$ for $k \geq 1$, and hence the identification error $e_I^k(t)$ will satisfy $\lim_{k\to\infty}|e_I^k(t)| = 0$, for all $t \in \{0,1,\ldots,N\}$. This proves Theorem 3. □

Remark 4. The ORNN is a promising tool for identification because it can approximate any "well-behaved" nonlinear function to any desired accuracy. This good function approximation is applied to estimate the unknown plant sensitivity in this paper. The plant sensitivity $y_u^j(t)$ in (8) can be approximated as follows:

$$y_u^j(t) \equiv \frac{\partial y^j(t+1)}{\partial u^j(t)} \approx \frac{\partial \hat{y}^j(t+1)}{\partial u^j(t)}. \tag{33}$$

Note that the index k in the identifier output $\hat{y}^k(t)$ is removed once the identification process stops. Applying the chain rule to (21), it yields

$$\begin{aligned}\frac{\partial \hat{y}^j(t+1)}{\partial u^j(t)} &= \frac{\partial O_I^{(3)^j}(t)}{\partial u^j(t)} = \sum_{\ell=1}^{M_I}\frac{\partial \hat{y}^j(t+1)}{\partial O_{I,\ell}^{(2)^j}(t)}\frac{\partial O_{I,\ell}^{(2)^j}(t)}{\partial u^j(t)}\\ &= \sum_{\ell=1}^{M_I} w_{I,\ell}^j(t)\frac{\partial O_{I,\ell}^{(2)^j}(t)}{\partial u^j(t)}.\end{aligned} \tag{34}$$

Also from (21), we have

$$\frac{\partial O_{I,\ell}^{(2)^j}(t)}{\partial u^j(t)} = f_{I,\ell}^{(2)'}\left(\mathrm{net}_{I,\ell}^{(2)^j}(t)\right)\frac{\partial \mathrm{net}_{I,\ell}^{(2)^j}(t)}{\partial u^j(t)}. \tag{35}$$

Since the inputs to ORNI are $u^j(t)$, $y^j(t)$ and $\mathbf{D}[O_I^{(3)^j}(t)]$, we further have

$$\begin{aligned}\mathrm{net}_{I,\ell}^{(2)^j}(t) &= V_{I,1\ell}^j(t)u^j(t) + V_{I,2\ell}^j(t)y^j(t)\\ &\quad + V_{I,3\ell}^j(t)\mathbf{D}\left[O_I^{(3)^j}(t)\right].\end{aligned} \tag{36}$$

Thus,

$$\frac{\partial \mathrm{net}_{I,\ell}^{(2)^j}(t)}{\partial u^j(t)} = V_{I,1\ell}^j(t). \tag{37}$$

From (34), (35) and (37), we obtain

$$\hat{y}_u^j(t) = \frac{\partial \hat{y}^j(t+1)}{\partial u^j(t)} = \sum_{\ell=1}^{M_I} w_{I,\ell}^j(t) f_{I,\ell}^{(2)'}\left(\mathrm{net}_{I,\ell}^{(2)^j}(t)\right)V_{I,1\ell}^j(t), \tag{38}$$

where $0 < f_{I,\ell}^{(2)'}(\mathrm{net}_{I,\ell}^{(2)^j}(t)) < 0.5$. If we define $\|w_I^j(t)\| \equiv \max_\ell|w_{I,\ell}^j(t)|$, and $\|V_{I,1}^j(t)\| \equiv \max_\ell|V_{I,1\ell}^j(t)|$, then

$$\begin{aligned}\left|\hat{y}_u^j(t)\right| &\leq M_I\left\|w_I^j(t)\right\|\left\|f_{I,\ell}^{(2)'}\left(\mathrm{net}_{I,\ell}^{(2)^j}(t)\right)V_{I,1}^j(t)\right\|\\ &\leq \frac{M_I}{2}\left\|w_I^j(t)\right\|\left\|V_{I,1}^j(t)\right\| \equiv \hat{y}_{u,\max}^j(t).\end{aligned} \tag{39}$$

The sign function and upper bounding function of plant sensitivity after finishing the identification process at jth control iteration can be obtained as follows:

$$\begin{aligned}\mathrm{sgn}\left(y_u^j(t)\right) &= \mathrm{sgn}\left(\hat{y}_u^j(t)\right)\\ y_{u,\max}(t) &= \max\left\{\hat{y}_{u,\max}^j(t), \hat{y}_{u,\max}^{j-1}(t)\right\}.\end{aligned} \tag{40}$$

It is noted that we do not need the exact plant sensitivity $y_u^j(t)$ for the design of adaptive law (14). Even though there may exist certain approximation error between $y_u^j(t)$ and $\hat{y}_u^j(t)$, we can still guarantee the convergence of learning error since only a upper bounding function is required. Also note that the value of $\mathrm{sgn}(y_u^j(t))$ (+1 or −1) can be easily determined from the identification result.

4. Simulation Example

In this section, we use the proposed ORNN-based ILC to iteratively control an unknown non-BIBO nonlinear dynamic plant [26, 29]. The difference equation of the nonlinear dynamic plant is given as

$$\begin{aligned}y^j(t+1) &= 0.2\left(y^j(t)\right)^2 + 0.2y^j(t-1)\\ &\quad + 0.4\sin\left(0.5\left(y^j(t-1)+y^j(t)\right)\right)\\ &\quad \times\cos\left(0.5\left(y^j(t-1)+y^j(t)\right)\right)\\ &\quad + 1.2u^j(t),\end{aligned} \tag{41}$$

where $y^j(t)$ is the system output, $u^j(t)$ is the control input. The reference model is chosen as

$$y_d(t+1) = 0.6y_d(t) + r(t), \qquad y_d(0) = 0, \tag{42}$$

where $r(t) = \sin(2\pi t/25) + \sin(2\pi t/10)$ is a bounded reference input. The control objective is to force $y^j(t)$ to track the desired trajectory $y_d(t)$ as close as possible over a finite time interval $t \in \{1,\ldots,200\}$ except the initial point. The network weight adaptation for the ORNI and ORNC is designed according to (14), (15), and (28), respectively. In the ORNC, we set $W_c^j(t) \in \mathbb{R}^{2\times 1}$ and $V_c^j(t) \in \mathbb{R}^{4\times 2}$, that is, only two hidden nodes in layer 2 are used to construct the ORNC. In a similar way, we let $W_I^k(t) \in \mathbb{R}^{2\times 1}$

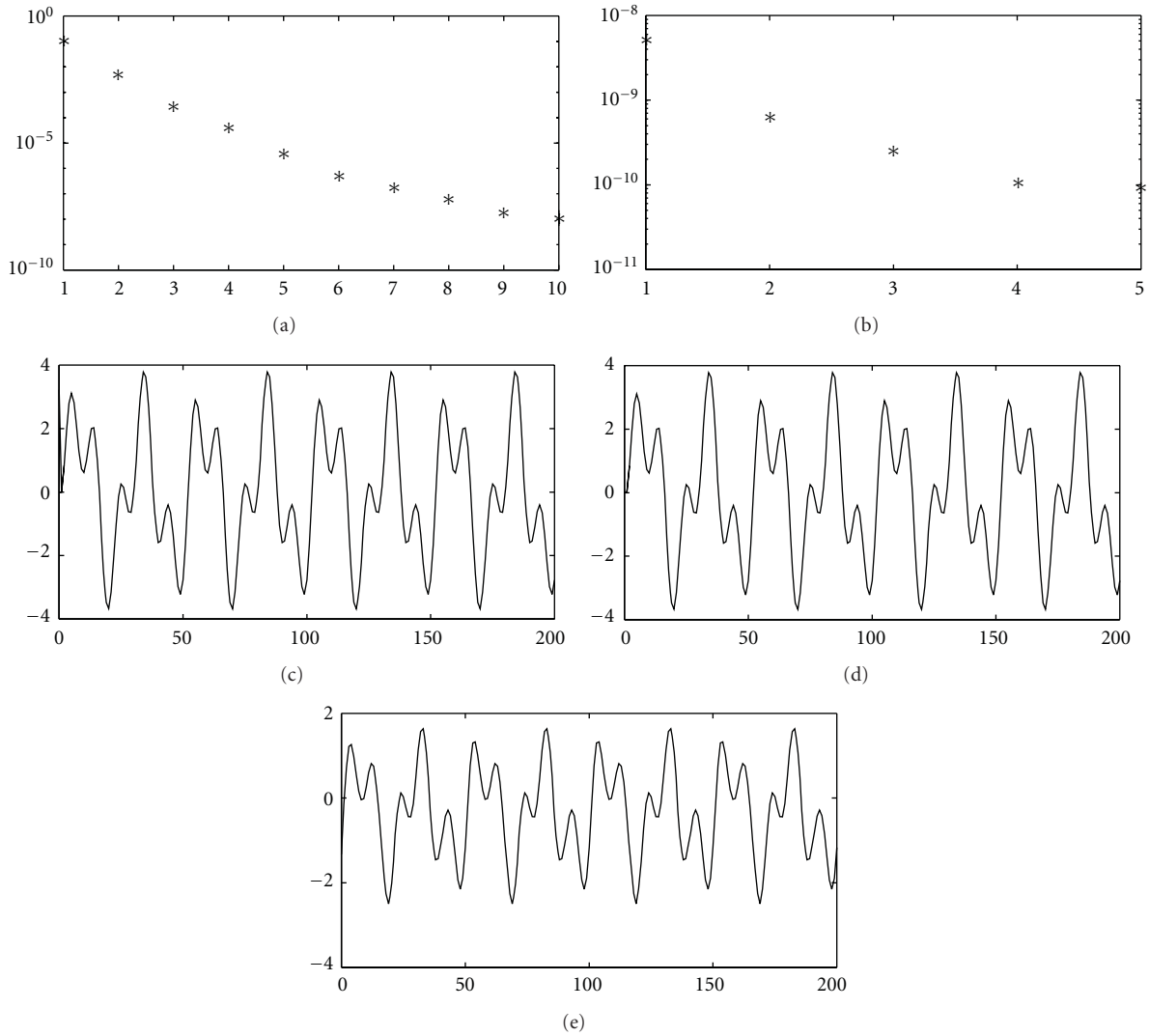


Figure 3: (a) $\max_{t\in\{1,\ldots,200\}}|e_c^j(t)|$ versus control iteration j. (b) $\max_{t\in\{0,\ldots,200\}}|e_I^{10,k}(t)|$ versus identification process k at the 10th control iteration. (c) $y^{10}(t)$ (dotted line) and $y_d(t)$ (solid line) versus time t at the 10th control iteration. (d) $\hat{y}^{10}(t)$ (dotted line) and $y^{10}(t)$ (solid line) versus time t at the 10th control iteration. (e) $u^{10}(t)$ at the 10th control iteration versus time t.

and $V_I^k(t) \in \mathbb{R}^{3\times 2}$, that is, only two hidden nodes in layer 2 are used to set up the ORNI. For simplicity, all the initial conditions of ORNC parameters are set to be 0 at the first control iteration. In addition, the initial ORNI parameters are set to be 0 at the first identification process which begins after the first control iteration. We assume that the plant initial condition satisfies $y^j(0) = 2 + \text{randn}$ where randn is a generator of random number with normal distribution, mean = 0 and variance = 1. To study the effects of learning performances, we first show the maximum value of tracking error $|e_c^j(t)|$, $t \in \{1,\ldots,200\}$ with respect to control iteration j in Figure 3(a). It is noted that $|e_c^j(0)|$ is omitted in calculating the maximum value of tracking error since it is not controllable. The identification error at 10th control iteration $|e_I^{10,k}(t)|$ with respect to identification process k is shown in Figure 3(b). According to the simulation results, it is clear that the asymptotic convergence proved in Theorems 1 and 3 is achieved. Since a reasonable tracking performance is almost observed at the 10th control iteration, the trajectories between the desired output $y_d(t)$ and plant output $y^{10}(t)$ at the 10th control iteration are shown to demonstrate the control performance in Figure 3(c). Figure 3(d) shows the comparison between the identification result of $\hat{y}^{10}(t)$ and the plant output $y^{10}(t)$. The nice identification result enables the ORNI to provide the required information for the design of ORNC. Finally, the bounded control input $u^{10}(t)$ is plotted in Figure 3(e).

5. Conclusion

For controlling a repeatable nonaffine nonlinear dynamic plant, we propose an output-recurrent-neural-network-based iterative learning control system in this paper. The control structure consists of an ORNC used as an iterative learning controller and an ORNI used as an identifier.

The ORNC is the main controller utilized to achieve the repetitive control task. The ORNI is an auxiliary component utilized to provide some useful information from plant sensitivity for the design of ORNC's adaptive laws. All the network weights of ORNC and ORNI will be tuned during the control iteration and identification process so that no prior plant knowledge is required. The adaptive laws for the weights of ORNC and ORNI and the analysis of learning performances are determined via a Lyapunov-like analysis. We show that if the ORNI can provide the knowledge of plant sensitivity for ORNC, then output tracking error will asymptotically converge to zero except an initial resetting error. We also show that the objective of identification can be achieved by the ORNI if the number of identifications is large enough.

Acknowledgment

This work is supported by National Science Council, R.O.C., under Grant NSC99-2221-E-211-011-MY2.

References

[1] K. L. Moore and J. X. Xu, "Special issue on iterative learning control," *International Journal of Control*, vol. 73, no. 10, pp. 819–823, 2000.

[2] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning," *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[3] H. S. Ahn, Y. Q. Chen, and K. L. Moore, "Iterative learning control: brief survey and categorization," *IEEE Transactions on Systems, Man and Cybernetics. Part C*, vol. 37, no. 6, pp. 1099–1121, 2007.

[4] J. X. Xu and J. Xu, "Observer based learning control for a class of nonlinear systems with time-varying parametric uncertainties," *IEEE Transactions on Automatic Control*, vol. 49, no. 2, pp. 275–281, 2004.

[5] I. Rotariu, M. Steinbuch, and R. Ellenbroek, "Adaptive iterative learning control for high precision motion systems," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 5, pp. 1075–1082, 2008.

[6] A. Tayebi, S. Abdul, M. B. Zaremba, and Y. Ye, "Robust iterative learning control design: application to a robot manipulator," *IEEE/ASME Transactions on Mechatronics*, vol. 13, no. 5, pp. 608–613, 2008.

[7] J. X. Xu and B. Viswanathan, "Adaptive robust iterative learning control with dead zone scheme," *Automatica*, vol. 36, no. 1, pp. 91–99, 2000.

[8] J.Y. Choi and H. J. Park, "Neural-based iterative learning control for unknown systems," in *Proceedings of the 2nd Asian Control Conference*, pp. II243–II246, Seoul, Korea, 1997.

[9] C. J. Chien, "A sampled-data iterative learning control using fuzzy network design," *International Journal of Control*, vol. 73, no. 10, pp. 902–913, 2000.

[10] J. H. Park, S. H. Huh, S. H. Kim, S. J. Seo, and G. T. Park, "Direct adaptive controller for nonaffine nonlinear systems using self-structuring neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 414–422, 2005.

[11] J. S. Wang and Y. P. Chen, "A fully automated recurrent neural network for unknown dynamic system identification and control," *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 6, pp. 1363–1372, 2006.

[12] C. F. Hsu, C. M. Lin, and T. T. Lee, "Wavelet adaptive backstepping control for a class of nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1–9, 2006.

[13] C. M. Lin, L. Y. Chen, and C. H. Chen, "RCMAC hybrid control for MIMO uncertain nonlinear systems using sliding-mode technology," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 708–720, 2007.

[14] Z. G. Hou, M. M. Gupta, P. N. Nikiforuk, M. Tan, and L. Cheng, "A recurrent neural network for hierarchical control of interconnected dynamic systems," *IEEE Transactions on Neural Networks*, vol. 18, no. 2, pp. 466–481, 2007.

[15] Z. Liu, R. E. Torres, N. Patel, and Q. Wang, "Further development of input-to-state stabilizing control for dynamic neural network systems," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 38, no. 6, pp. 1425–1433, 2008.

[16] Z. Liu, S. C. Shih, and Q. Wang, "Global robust stabilizing control for a dynamic neural network system," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 39, no. 2, pp. 426–436, 2009.

[17] Y. G. Leu, W. Y. Wang, and T. T. Lee, "Observer-based direct adaptive fuzzy-neural control for nonaffine nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 853–861, 2005.

[18] S. Labiod and T. M. Guerra, "Adaptive fuzzy control of a class of SISO nonaffine nonlinear systems," *Fuzzy Sets and Systems*, vol. 158, no. 10, pp. 1126–1137, 2007.

[19] B. Chen, X. Liu, and S. Tong, "Adaptive fuzzy output tracking control of MIMO nonlinear uncertain systems," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 2, pp. 287–300, 2007.

[20] Y. J. Liu and W. Wang, "Adaptive fuzzy control for a class of uncertain nonaffne nonlinear system," *Information Sciences*, vol. 177, no. 18, pp. 3901–3917, 2007.

[21] R. J. Wai and C. M. Liu, "Design of dynamic petri recurrent fuzzy neural network and its application to path-tracking control of nonholonomic mobile robot," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 7, pp. 2667–2683, 2009.

[22] C. H. Lee, Y. C. Lee, and F. Y. Chang, "A dynamic fuzzy neural system design via hybridization of EM and PSO algorithms," *IAENG International Journal of Computer Science*, vol. 37, no. 3, 2010.

[23] W. G. Seo, B. H. Park, and J. S. Lee, "Adaptive fuzzy learning control for a class of nonlinear dynamic systems," *International Journal of Intelligent Systems*, vol. 15, no. 12, pp. 1157–1175, 2000.

[24] C. J. Chien and L. C. Fu, "Iterative learning control of nonlinear systems using neural network design," *Asian Journal of Control*, vol. 4, no. 1, pp. 21–29, 2002.

[25] C. J. Chien, "A combined adaptive law for fuzzy iterative learning control of nonlinear systems with varying control tasks," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 1, pp. 40–51, 2008.

[26] T. W.S. Chow and Y. Fang, "A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 1, pp. 151–161, 1998.

[27] Y. C. Wang, C. J. Chien, and D. T. Lee, "An output recurrent fuzzy neural network based iterative learning control for nonlinear systems," in *Proceedings of the IEEE International*

Conference on Fuzzy Systems, pp. 1563–1569, Hong Kong, 2008.

[28] Y. C. Wang and C. C. Teng, "Output recurrent fuzzy neural networks based model reference control for unknown nonlinear systems," *International Journal of Fuzzy Systems*, vol. 6, no. 1, pp. 28–37, 2004.

[29] C. C. Ku and K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 144–156, 1995.

Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2012, Article ID 618403, 10 pages
doi:10.1155/2012/618403

Research Article

3D Nonparametric Neural Identification

Rita Q. Fuentes,[1] Isaac Chairez,[2] Alexander Poznyak,[1] and Tatyana Poznyak[3]

[1] *Automatic Control Department, CINVESTAV-IPN, 07360 México, DF, Mexico*
[2] *Bioprocess Department, UPIBI-IPN, 07360 México, DF, Mexico*
[3] *SEPI, ESIQIE-IPN, 07738 México, DF, Mexico*

Correspondence should be addressed to Rita Q. Fuentes, rita.q.fuentes.a@gmail.com

Received 23 August 2011; Accepted 2 October 2011

Academic Editor: Haibo He

Copyright © 2012 Rita Q. Fuentes et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents the state identification study of 3D partial differential equations (PDEs) using the differential neural networks (DNNs) approximation. There are so many physical situations in applied mathematics and engineering that can be described by PDEs; these models possess the disadvantage of having many sources of uncertainties around their mathematical representation. Moreover, to find the exact solutions of those uncertain PDEs is not a trivial task especially if the PDE is described in two or more dimensions. Given the continuous nature and the temporal evolution of these systems, differential neural networks are an attractive option as nonparametric identifiers capable of estimating a 3D distributed model. The adaptive laws for weights ensure the "practical stability" of the DNN trajectories to the parabolic three-dimensional (3D) PDE states. To verify the qualitative behavior of the suggested methodology, here a nonparametric modeling problem for a distributed parameter plant is analyzed.

1. Introduction

1.1. 3D Partial Differential Equations. Partial differential equations (PDEs) are of vast importance in applied mathematics, physics, and engineering since so many real physical situations can be modelled by them. The dynamic description of natural phenomenons are usually described by a set of differential equations using mathematical modeling rules [1]. Almost every system described in PDE has already appeared in the one- and two-dimensional situations. Appending a third dimension ascends the dimensional ladder to its ultimate rung, in physical space at least. For instance, linear second-order 3D *partial differential equations* appear in many problems modeling equilibrium configurations of solid bodies, the three-dimensional wave equation governing vibrations of solids, liquids, gases, and electromagnetic waves, and the three-dimensional heat equation modeling basic spatial diffusion processes. These equations define a state representing rectangular coordinates on $\mathbb{R}^3$. There are some basic underlying solution techniques to solve 3D PDEs: separation of variables and Green's functions or fundamental solutions. Unfortunately, the most powerful of the planar tools, conformal mapping, does not carry over to higher dimensions. In this way, many numerical techniques solving such PDE, for example, the finite difference method (FDM) and the finite element method (FEM), have been developed (see [2, 3]). The principal disadvantage of these methods is that they require the complete mathematical knowledge of the system to define a *mesh* (domain discretization), where the functions are approximated locally. The construction of a mesh in two or more dimensions is a nontrivial task. Usually, in practice, only low-order approximations are employed resulting in a continuous approximation of the function across the mesh but not in its partial derivatives. The approximation discontinuities of the derivative can adversely affect the stability of the solution. However, all those methods are well defined if the PDE structure is perfectly known. Actually, the most of suitable numerical solutions could be achieved only if the PDE is linear. Nevertheless, there are not so many methods to solve or approximate the PDE solution when its structure (even in a linear case) is uncertain. This paper suggests a different numerical solution for uncertain systems (given by a 3D PDE) based on the *Neural Network* approach [4].

1.2. Application of Neural Networks to Model PDEs. Recent results show that neural networks techniques seem to be very effective to identify a wide class or systems when we have no complete model information, or even when the plant is considered as a gray box. It is well known that radial basis function neural networks (RBFNNs) and MultiLayer Perceptrons (MLPs) are considered as a powerful tool to approximate nonlinear uncertain functions (see [5]): any continuous function defined on a compact set can be approximated to arbitrary accuracy by such class of neural networks [6]. Since the solutions of interesting PDEs are uniformly continuous and the viable sets that arise in common problems are often compact, neural networks seem like ideal candidates for approximating viability problems (see [7, 8]). Neural networks may provide exact approximation for PDE solutions; however, numerical constraints avoid this possible exactness because it is almost impossible to simulate NN structures with infinite number of nodes (see [7, 9, 10]). The *Differential Neural Network* (DNN) approach avoids many problems related to global extremum search converting the learning process to an adequate feedback design (see [11, 12]). Lyapunov's stability theory has been used within the neural networks framework (see [4, 11, 13]). The contributions given in this paper regard the development of a nonparametric identifier for 3D uncertain systems described by partial differential equations. The method produces an artificial mathematical model in three dimensions that is able to describe the PDEs dynamics. The required numerical algorithm to solve the non-parametric identifier was also developed.

2. 3D Finite Differences Approximation

The problem requires the proposal of a non-parametric identifier based on DNN in three dimensions. The problem here may be treated within the PDEs framework. Therefore, this section introduces the DNN approximation characteristics to reconstruct the trajectories profiles for a family of 3D PDEs.

Consider the set of uncertain second-order PDEs

$$u_t = f\left(u, u_x, u_{xx}, u_y, u_{yy}, u_z, u_{zz}, u_{xy}, u_{yx}, u_{xz}, u_{yz}\right) + \xi, \quad (1)$$

here $u_h = u_h(x, y, z, t)$, where h represents t, x, y, z, xx, yy, zz, xy, xz, yx, yz and $u = u(x, y, z, t)$ has n components ($u(x, y, z, t) \in \Re^n$) defined in a domain given by $[x, y, z) \in [0, 1] \times [0, 1] \times [0, 1]$, $t \geq 0$, $\xi = \xi(x, y, z, t) \in \Re^n$ is a noise in the state dynamics. This PDE has a set of initial and boundary conditions given by

$$\begin{aligned} u_x(0, y, z, t) &= 0 \in \Re^n, \qquad u(x, 0, z, t) = u_0 \in \Re^n, \\ u(x, y, 0, t) &= u_0 \in \Re^n, \qquad u(x, y, z, 0) = c \in \Re^n. \end{aligned} \quad (2)$$

In (1), $u_t(x, y, z, t)$ stands for $\partial u(x, y, z, t)/\partial t$.

System (1) armed with boundary and initial conditions (2) is driven in a Hilbert space H equipped with an inner product $(\cdot, \cdot)$. Let us consider a vector function $g(t) \in H$ to be a piecewise continuous in t. By $L_\infty(a, b; H)$ we denote the set of H-valued functions g such that $(g(\cdot), u)$ is Lebesgue measurable for all $u \in H$ and $ess \max_{t\in[a,b]} \|g(\gamma, t)\| < \infty$, $\gamma \in \Re^n$. Suppose that the nonlinear function $g(\gamma, t)$ satisfies the *Lipschitz condition* $\|g(\gamma, t) - g(\eta, t)\| \leq L\|\gamma - \eta\|$,

$$\forall \gamma, \eta \in B_r(\gamma_0) := \{\gamma \in \Re^n \mid ||\gamma - \gamma_0|| \leq r, \ \forall t \in [t_0, t_1]\}, \quad (3)$$

where L is positive constant and $\|g\|^2 = (g, g)$ is used just to ensure that there exists some $\delta > 0$ such that the state equation $\dot{\gamma} = g(\gamma, t)$ with $\gamma(t_0) = \gamma_0$ has a unique solution over $[t_0, t_0 + \delta]$ (see [14]). The norm defined above stands for the Sobolev space defined in [15] as follows.

Definition 1. Let Ω be an open set in $\Re^n$, and let $\nu \in C^m(\Omega)$. Define the norm of $\nu(\gamma)$ as

$$\|\nu\|_{m,p} := \sum_{0\leq|\alpha|\leq m} \left(\int_\Omega |D^\alpha \nu(\gamma)|^p d\gamma\right)^{1/p} \quad (4)$$

($1 \leq p < \infty$, $D^\alpha \nu(\gamma) := (\partial^\alpha/\partial\gamma^\alpha)\nu(\gamma)$). This is the Sobolev norm in which the integration is performed in the Lebesgue sense. The completion of the space of function $\nu(\gamma) \in C^m(\Omega)$: $\|\nu\|_{m,p} < \infty$ with respect to $\|\cdot\|_{m,p}$ is the Sobolev space $H^{m,p}(\Omega)$. For $p = 2$, the Sobolev space is a Hilbert space (see [14, 15]).

Below we will use the norm (4) for the functions $u(\cdot, t)$ for each fixed t.

2.1. Numerical Approximation for Uncertain Functions. Now consider a function $h_0(\cdot)$ in $H^{m,2}(\Omega)$. By [16], $h_0(\cdot)$ can be rewritten as

$$\begin{aligned} h_0(\gamma, \theta^*) &= \sum_i \sum_j \sum_k \theta^*_{ijk} \Psi_{ijk}(\gamma), \\ \theta_{ijk} &= \int_{-\infty}^{+\infty} h_0(\gamma)\Psi_{ijk}(\gamma) d\gamma, \ \forall i, j, k \in \mathbb{Z}, \end{aligned} \quad (5)$$

where $\{\Psi_{ijk}(\gamma)\}$ are functions constituting a basis in $H^{m,2}(\Omega)$. Last expression is referred to as a vector function series expansion of $h_0(\gamma, \theta^*)$. Based on this series expansion, an NN takes the following mathematical structure:

$$h_0(\gamma, \theta) := \sum_{i=L_1}^{L_2} \sum_{j=M_1}^{M_2} \sum_{k=N_1}^{N_2} \theta_{ijk} \Psi_{ijk}(\gamma) = \Theta^T W(\gamma) \quad (6)$$

that can be used to approximate a nonlinear function $h_0(\gamma, \theta^*) \in S$ with an adequate selection of integers L_1, L_2, M_1, M_2, N_1, $N_2 \in \mathbb{Z}^+$, where

$$\begin{aligned} \Theta &= [\theta_{L_1M_1N_1}, \ldots, \theta_{L_1M_1N_2}, \ldots \theta_{L_2M_1N_1}, \ldots \theta_{L_2M_2N_2}]^\intercal, \\ W(\gamma) &= [\Psi_{L_1M_1N_1}, \ldots, \Psi_{L_1M_1N_2}, \ldots \Psi_{L_2M_1N_1}, \ldots \Psi_{L_2M_2N_2}]^\intercal. \end{aligned} \quad (7)$$

Following the Stone-Weierstrass Theorem [17], if

$$\epsilon(L_1, L_2, M_1, M_2, N_1, N_2) = h_0(\gamma, \theta^*) - h_0(\gamma, \theta) \quad (8)$$

is the NN approximation error, then for any arbitrary positive constant ε there are some constants $L_1, L_2, M_1, M_2, N_1, N_2 \in \mathbb{Z}$ such that for all $x \in X \subset \Re$

$$\|\epsilon(L_1, L_2, M_1, M_2, N_1, N_2)\|_2 \leq \varepsilon. \quad (9)$$

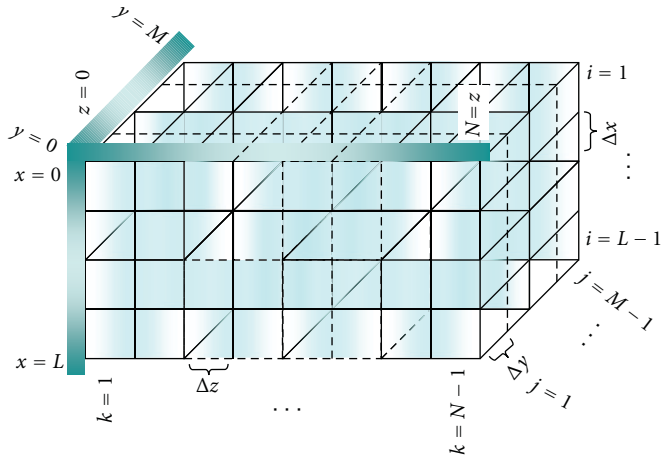


Figure 1: Constructed grid for 3 dimensions.

The main idea behind the application of DNN [11] to approximate the 3D PDEs solution is to use a class of finite-difference methods but for uncertain nonlinear functions. So, it is necessary to construct an interior set (commonly called *grid* or *mesh*) that divides the subdomain $x \in [0,1]$ in L equidistant sections, $y \in [0,1]$ in M, and $z \in [0,1]$ in N equidistant sections, each one of them (Figure 1) defined as (x^i, y^j, z^k) in such a way that $x^0 = y^0 = z^0 = 0$ and $x^L = y^M = z^N = 1$.

Using the mesh description, one can use the next definitions:

$$\begin{aligned} u^{i,j,k}(t) &:= u\left(x^i, y^j, z^k, t\right), \\ u_t^{i,j,k}(t) &:= \left.\frac{\partial u(x,y,z,t)}{\partial t}\right|_{x=x^i, y=y^j, z=z^k}, \\ u_x^{i,j,k}(t) &:= \left. u_x(x,y,z,t)\right|_{x=x^i, y=y^j, z=z^k}, \\ u_{xx}^{i,j,k}(t) &:= \left. u_{xx}(x,y,z,t)\right|_{x=x^i, y=y^j, z=z^k}. \end{aligned} \tag{10}$$

Analogously, we may consider the other cases (u_{xx}, u_y, u_{yy}, u_z, u_{zz}, u_{xy}, u_{xz}, u_{yz}). Using the mesh description and applying the finite-difference representation, one gets

$$\begin{aligned} u_x^{i,j,k}(t) &\simeq \frac{u^{i,j,k}(t) - u^{i-1,j,k}(t)}{\Delta x}, \\ u_{xx}^{i,j,k}(t) &\simeq \frac{u_x^{i,j,k}(t) - u_x^{i-1,j,k}(t)}{\Delta^2 x}, \end{aligned} \tag{11}$$

and it follows for all cases such that the $(\Delta x, \Delta y, \Delta z)$-*approximation* of the nonlinear PDE (1) can be represented as

$$\begin{aligned} \dot{u}^{i,j,k}(t) &= u_t^{i,j,k}(t) \\ &\simeq \Phi\Big(u^{i,j,k}, u^{i-1,j,k}, u^{i-2,j,k}, u^{i,j-1,k}, \\ &\qquad u^{i,j-2,k}, u^{i,j,k-1}, u^{i,j,k-2}, u^{i-1,j-1,k}, \\ &\qquad u^{i-1,j,k-1}, u^{i,j-1,k-1}, u^{i-1,j-1,k-1}\Big) \\ &\qquad (i = 0,\ldots,L;\ j = 0,\ldots,M,\ k = 0,\ldots,N). \end{aligned} \tag{12}$$

2.2. 3D Approximation for Uncertain PDE. By simple adding and subtracting the corresponding terms, one can describe (1) as

$$\begin{aligned} u_t = {} & Au + \mathring{V}_1\sigma u + \mathring{V}_2\varphi^1 u_x + \mathring{V}_3\gamma^1 u_{xx}\mathring{V}_4\varphi^2 u_y + \mathring{V}_5\gamma^2 u_{yy} \\ & + \mathring{V}_6\varphi^3 u_z + \mathring{V}_7\gamma^3 u_{zz} + \mathring{V}_8\psi^1 u_{xy} \\ & + \mathring{V}_9\psi^2 u_{yz}\mathring{V}_{10}\psi^3 u_{xz} + \mathring{V}_{11}\sigma^2 u_{xyz} + \tilde{f}, \end{aligned} \tag{13}$$

where $u_t = u_t(x,y,z,t)$, $u = u(x,y,z,t)$, $\sigma = \sigma(x,y,z)$, $u_x = u_x(x,y,z,t)$, $u_{xx} = u_{xx}(x,y,z,t)$, the same for σ^2, φ^i, γ^i, ψ^i, u_y, u_{yy}, u_z, u_{zz}, u_{xy}, u_{yz}, u_{xz}, and u_{xyz} $(i = \overline{1,3})$, $A \in \Re^{n\times n}$, $\mathring{V}_1 \in \Re^{n\times s_1}$, $\mathring{V}_2 \in \Re^{n\times s_2}$, $\mathring{V}_3 \in \Re^{n\times s_3}$, $\tilde{f} = \tilde{f}(x,y,z,t)$, $\mathring{V}_4 \in \Re^{n\times s_4}$, $\mathring{V}_5 \in \Re^{n\times s_5}$, $\mathring{V}_6 \in \Re^{n\times s_6}$, $\mathring{V}_7 \in \Re^{n\times s_7}$, $\mathring{V}_8 \in \Re^{n\times s_8}$, $\mathring{V}_9 \in \Re^{n\times s_9}$, $\mathring{V}_{10} \in \Re^{n\times s_{10}}$.

Here $\tilde{f}(x,t) \in \Re^n$ represents the *modelling error* term, A and $\mathring{V}_k$ $(k = \overline{1,6})$ any constant matrices and the set of sigmoidal functions have the corresponding size $(\sigma(x,y,z) \in \Re^{s_1}$, $\varphi^1(x,y,z) \in \Re^{s_2}$, $\gamma^1(x,y,z) \in \Re^{s_3}$, $\varphi^2(x,y,z) \in \Re^{s_4}$, $\gamma^2(x,y,z) \in \Re^{s_5}$, $\psi^1(x,y,z) \in \Re^{s_6}$, $\varphi^3(x,y,z) \in \Re^{s_7}$, $\gamma^3(x,y,z) \in \Re^{s_8}$, $\psi^2(x,y,z) \in \Re^{s_9}$, $\psi^3(x,y,z) \in \Re^{s_{10}}$, and $\sigma^2(x,y,z) \in \Re^{s_{11}})$ and are known as *the neural network set of activation functions*. These functions obey the following sector conditions:

$$\begin{aligned} &\left\|\sigma^1(x,y,z) - \sigma^1(x',y',z')\right\|^2 \\ &\quad \le L_{\sigma^1}\left(\|x-x'\|^2 + \|y-y'\|^2 + \|z-z'\|^2\right), \\ &\left\|\sigma^2(x,y,z) - \sigma^2(x',y',z')\right\|^2 \\ &\quad \le L_{\sigma^2}\left(\|x-x'\|^2 + \|y-y'\|^2 + \|z-z'\|^2\right), \\ &\left\|\phi^s(x,y,z) - \phi^s(x',y',z')\right\|^2 \\ &\quad \le L_{\phi^s}\left(\|x-x'\|^2 + \|y-y'\|^2 + \|z-z'\|^2\right) \end{aligned} \tag{14}$$

which are bounded in x, y, and z, that is,

$$\|\sigma(\cdot)\|^2 \leq \sigma^{(l-1)+}, \quad \left\|\varphi^l(\cdot)\right\|^2 \leq \varphi^{l+},$$
$$\left\|\gamma^l(\cdot)\right\|^2 \leq \gamma^{l+}, \quad \left\|\psi^l(\cdot)\right\| \leq \psi^{l+}, \quad l = 1, 3. \tag{15}$$

Following the methodology of DNN [11] and applying the same representation to (12), we get for each $i \in (1,\ldots,L)$, $j \in (1,\ldots,M)$, $k \in (1,\ldots,N)$ the following robust adaptive non-parametric identifier:

$$u_t^{i,j,k}(t) = A^{i,j,k}u^{i,j,k}(t) + \sum_{p=1}^{11} \overset{\circ}{W}{}_p^{i,j,k}\phi^s U(t) + \widetilde{f}^{i,j,k}(t). \tag{16}$$

In the sum we have that $s = \overline{1,3}$, ϕ represents functions σ, φ, γ, ψ, and U can be taken as the corresponding $u^{i,j,k}$, $u^{i-1,j,k}$, $u^{i-2,j,k}$, $u^{i,j-1,k}$, $u^{i,j-2,k}$, $u^{i,j,k-1}$, $u^{i,j,k-2}$, $u^{i-1,j-1,k}$, $u^{i,j-1,k-1}$, $u^{i-1,j,k-1}$, $u^{i-1,j-1,k-1}$. In this equation the term $\widetilde{f}^{i,j,k}(t)$, which is usually recognized as the modeling error, satisfies the following identify, and here, it has been omitted the dependence to x^i, y^j, z^k of each sigmoidal function:

$$\begin{aligned}
&\widetilde{f}^{i,j,k}(t) \\
&:= \Phi\Big(u^{i,j,k}, u^{i-1,j,k}, u^{i-2,j,k}, u^{i,j-1,k}, u^{i,j-2,k}, u^{i,j,k-1}, u^{i,j,k-2}, \\
&\qquad u^{i-1,j-1,k}, u^{i-1,j,k-1}, u^{i,j-1,k-1}, u^{i-1,j-1,k-1}\Big) \\
&\quad - A^{i,j,k}u^{i,j,k}(t) - \sum_{p=1}^{11} \overset{\circ}{W}{}_p^{i,j,k}\phi^s U(t),
\end{aligned} \tag{17}$$

where $\overset{\circ}{W}{}_p^{i} \in \Re^{n\times s_p}$, $p = \overline{1,11}$, ϕ^s, ϕ represents functions σ, φ, γ, ψ and $s = \overline{1,3}$, $U(t)$ represents the corresponding ($u^{i,j,k}$, $u^{i-1,j,k}$, $u^{i-2,j,k}$, $u^{i,j-1,k}$, $u^{i,j-2,k}$, $u^{i,j,k-1}$, $u^{i,j,k-2}$, $u^{i-1,j-1,k}$, $u^{i-1,j,k-1}$, $u^{i,j-1,k-1}$, $u^{i-1,j-1,k-1}$).

We will assume that the modeling error terms satisfy the following.

Assumption 2. The modeling error is absolutely bounded in Ω:

$$\left\|\widetilde{f}^{i,j,k}\right\|^2 \leq f_1^{i,j,k}. \tag{18}$$

Assumption 3. The error modeling gradient

$$\nabla_s \widetilde{f}(x,y,z,t)\Big|_{s=s_i} := \nabla_s \widetilde{f}^{i,j,k} \tag{19}$$

is bounded as $\|\nabla_s \widetilde{f}^{i,j,k}\|^2 \leq f_r^{i,j,k}$, where $s = x, y, z$ and $f_r^{i,j,k}$ $(r = \overline{1,3})$ are constants.

3. DNN Identification for Distributed Parameters Systems

3.1. DNN Identifier Structure. Based on the DNN methodology [11], consider the DNN identifier

$$\frac{d}{dt}\widehat{u}^{i,j,k} = A^{i,j,k}\widehat{u}^{i,j,k} + \sum_{p=1}^{11} W_p^{i,j,k}(t)\phi^s\widehat{U} \tag{20}$$

for all $i = 0,\ldots,L$; $\widehat{u}_{-1}(t) = \widehat{u}_{-2}(t) = 0$, where ϕ represents activation functions σ, φ, γ, and ψ, $s = \overline{1,3}$, $\widehat{U}$ is each one of the states $\widehat{u}^{i,j,k}$, $\widehat{u}^{i-1,j,k}$, $\widehat{u}^{i-2,j,k}$, $\widehat{u}^{i,j-1,k}$, $\widehat{u}^{i,j-2,k}$, $\widehat{u}^{i,j,k-1}$, $\widehat{u}^{i,j,k-1}$, $\widehat{u}^{i,j,k-2}$, $\widehat{u}^{i-1,j-1,k}$, $\widehat{u}^{i-1,j,k-1}$, $\widehat{u}^{i-1,j-1,k-1}$, and $A^{i,j,k} \in \Re^{n\times n}$ is a constant matrix to be selected, $\widehat{u}^{i,j,k}(t)$ is the estimate of $u^{i,j,k}(t)$. Obviously that proposed methodology implies the designing of individual DNN identifier for each point x_i, y_j, z_k. The collection of such identifiers will constitute a DNN net containing $N \times M$ connected DNN identifiers working in parallel. Here $\sigma^1(x^i, y^j, z^k)$, $\varphi^1(x^i, y^j, z^k)$, $\varphi^2(x^i, y^j, z^k)$, $\varphi^3(x^i, y^j, z^k)$, $\gamma^1(x^i, y^j, z^k)$, $\gamma^2(x^i, y^j, z^k)$, $\gamma^3(x^i, y^j, z^k)$, $\psi^1(x^i, y^j, z^k)$, $\psi^2(x^i, y^j, z^k)$, $\psi^3(x^i, y^j, z^k)$, and $\sigma^2(x^i, y^j, z^k)$ are the NN activation vectors. This means that the applied DNN-approximation significantly simplifies the specification of $\sigma^1(\cdot,\cdot)$, $\varphi^1(\cdot,\cdot)$, $\varphi^2(\cdot,\cdot)$, $\varphi^3(\cdot,\cdot)$, $\gamma^1(\cdot,\cdot)$, $\gamma^2(\cdot,\cdot)$, $\gamma^3(\cdot,\cdot)$ and $\psi^1(\cdot,\cdot)$, $\psi^2(\cdot,\cdot)$, $\psi^3(\cdot,\cdot)$, $\sigma^2(\cdot,\cdot)$ which now are constant for any x^i, y^j, z^k fixed.

3.2. Learning Laws for Identifier's Weights. For each $i = 0, \ldots, L$, $j = 0,\ldots,M$, $k = 0,\ldots,N$, define the vector-functions defining the error between the trajectories produced by the model and the DNN-identifier as well as their derivatives with respect to x, y, and z for each i, j, k:

$$\begin{aligned}
\widetilde{u}^{i,j,k}(t) &:= \widehat{u}^{i,j,k}(t) - u^{i,j,k}(t), \\
\widetilde{u}_s^{i,j,k}(t) &:= \widehat{u}_s^{i,j,k}(t) - u_s^{i,j,k}(t), \\
s &= x, y, z.
\end{aligned} \tag{21}$$

Let $W_r^{i,j,k}(t) \in \Re^n$, $r = \overline{1,11}$ be time-variant matrices. These matrices satisfy the following nonlinear matrix differential equations:

$$\begin{aligned}
\dot{W}_r^{i,j,k}(t) := \frac{d}{dt}\overline{W}_r^{i,j,k}(t) &= -\alpha\widetilde{W}_r^{i,j,k}(t) \\
&\quad - K_r^{-1}P^{i,j,k}\widehat{u}^{i,j,k}\left(\widehat{U}^{(i,j,k),r}\right)^{\mathsf{T}}\left(\Omega^r(x^i,y^j,z^j)\right)^{\mathsf{T}} \\
&\quad - K_r^{-1}S_1^{i,j,k}\widetilde{u}_x^{i,j,k}\left(\widehat{U}^{(i,j,k),r}\right)^{\mathsf{T}}\left(\Omega_x^r(x^i,y^j,z^j)\right)^{\mathsf{T}} \\
&\quad - K_r^{-1}S_2^{i,j,k}\widetilde{u}_y^{i,j,k}\left(\widehat{U}^{(i,j,k),r}\right)^{\mathsf{T}}\left(\Omega_y^r(x^i,y^j,z^j)\right)^{\mathsf{T}} \\
&\quad - K_r^{-1}S_3^{i,j,k}\widetilde{u}_z^{i,j,k}\left(\widehat{U}^{(i,j,k),r}\right)^{\mathsf{T}}\left(\Omega_z^r(x^i,y^j,z^j)\right)^{\mathsf{T}},
\end{aligned} \tag{22}$$

where $\Omega^h(x^i, y^j, z^j) = S^l(x^i, y^j, z^j)$ $(h = \overline{1,10}, l = \overline{1,3})$ represents the corresponding sigmoidal functions $\sigma^l(x^i, y^j, z^j)$, $\varphi^l(x^i, y^j, z^j)$, $\gamma^l(x^i, y^j, z^j)$, and $\psi^l(x^i, y^j, z^j)$. Here

$$\begin{aligned}
\widehat{U}^{(i,j,k),1}(t) &= \widehat{u}^{i,j,k}(t), & \widetilde{U}^{(i,j,k),2}(t) &= \widehat{u}^{i-1,j,k}(t), \\
\widetilde{U}^{(i,j,k),3}(t) &= \widehat{u}^{i-2,j,k}(t), & \widetilde{U}^{(i,j,k),4}(t) &= \widehat{u}^{i,j-1,k}(t), \\
\widetilde{U}^{(i,j,k),5}(t) &= \widehat{u}^{i,j-2,k}(t), & \widetilde{U}^{(i,j,k),6}(t) &= \widehat{u}^{i-1,j-1,k}(t), \\
\widetilde{U}^{(i,j,k),7}(t) &= \widehat{u}^{i-1,j-1,k}(t), & \widetilde{U}^{(i,j,k),8}(t) &= \widehat{u}^{i-1,j-1,k}(t), \\
\widetilde{U}^{(i,j,k),9}(t) &= \widehat{u}^{i-1,j-1,k}(t), & \widetilde{U}^{(i,j,k),10}(t) &= \widehat{u}^{i-1,j-1,k}(t), \\
\widetilde{U}^{(i,j,k),11}(t) &= \widehat{u}^{i-1,j-1,k-1}(t)
\end{aligned} \tag{23}$$

with positive matrices $K_r (r = \overline{1,11})$ and $P^{i,j,k}$, $S_1^{i,j,k}$, and $S_2^{i,j,k}$ $(i = \overline{0,N}; j = \overline{0,M})$ which are positive definite solutions $(P^{i,j} > 0, S_1^{i,j} > 0, S_2^{i,j} > 0)$ and $S_3^{i,j,k}$ $(i = \overline{0,N}; j = \overline{0,M}, k = \overline{0,L})$ of the algebraic Riccati equations defined as follows:

$$\begin{aligned}
\mathrm{Ric}\left(P^{i,j,k}\right) &:= P^{i,j,k}A^{i,j,k} + \left[A^{i,j,k}\right]^{\mathsf{T}} P^{i,j,k} \\
&\quad + P^{i,j,k}R_P^{i,j,k}P^{i,j,k} + Q_P^{i,j,k} = 0, \\
\mathrm{Ric}\left(S_1^{i,j,k}\right) &:= S_1^{i,j,k}A^{i,j,k} + \left[A^{i,j}\right]^{\mathsf{T}} S_1^{i,j,k} \\
&\quad + S_1^{i,j,k}R_{S_1}^{i,j,k}S_1^{i,j,k} + Q_{S_1}^{i,j,k} = 0, \\
\mathrm{Ric}\left(S_2^{i,j,k}\right) &:= S_2^{i,j,k}A^{i,j,k} + \left[A^{i,j,k}\right]^{\mathsf{T}} S_2^{i,j,k} \\
&\quad + S_2^{i,j,k}R_{S_2}^{i,j,k}S_2^{i,j,k} + Q_{S_2}^{i,j,k} = 0, \\
\mathrm{Ric}\left(S_3^{i,j,k}\right) &:= S_3^{i,j,k}A^{i,j,k} + \left[A^{i,j,k}\right]^{\mathsf{T}} S_3^{i,j,k} \\
&\quad + S_3^{i,j,k}R_{S_3}^{i,j,k}S_3^{i,j,k} + Q_{S_3}^{i,j,k} = 0,
\end{aligned} \tag{24}$$

where each $R_B^{i,j,k}$ has the form

$$R_B^{i,j,k} := \sum_{r=1}^{11} \overset{\circ}{W}{}_r^{i,j,k} \Lambda_{r+b} \left(\overset{\circ}{W}{}_r^{i,j,k}\right)^{\mathsf{T}} + \Lambda_b, \tag{25}$$

where B can be P, S_1, S_2, and S_3 and $b = (7, 14, 21, 28)$.

Matrices $Q_B^{i,j,k}$ have the form

$$\begin{aligned}
Q_B^{i,j,k} &:= \left\|\Omega_m^1(x^i, y^j, z^k)\right\|^2_{\Lambda_1^{-1}} + \left\|\Omega_m^2(x^{i-1}, y^j, z^k)\right\|^2_{\Lambda_2^{-1}} \\
&\quad + \left\|\Omega_m^3(x^{i-2}, y^j, z^k)\right\|^2_{\Lambda_3^{-1}} + \left\|\Omega_m^4(x^i, y^{j-1}, z^k)\right\|^2_{\Lambda_4^{-1}} \\
&\quad + \left\|\Omega_m^5(x^i, y^{j-2}, z^k)\right\|^2_{\Lambda_5^{-1}} + \left\|\Omega_m^6(x^i, y^j, z^{k-1})\right\|^2_{\Lambda_6^{-1}} \\
&\quad + \left\|\Omega_m^7(x^i, y^j, z^{k-2})\right\|^2_{\Lambda_7^{-1}} + \left\|\Omega_m^8(x^{i-1}, y^{j-1}, z^k)\right\|^2_{\Lambda_8^{-1}} \\
&\quad + \left\|\Omega_m^9(x^i, y^{j-1}, z^{k-1})\right\|^2_{\Lambda_9^{-1}} + \left\|\Omega_m^{10}(x^{i-1}, y^j, z^{k-1})\right\|^2_{\Lambda_{10}^{-1}} \\
&\quad + \left\|\Omega_m^{11}(x^{i-1}, y^{j-1}, z^{k-1})\right\|^2_{\Lambda_{11}^{-1}} + Q_B^{i,j,k},
\end{aligned} \tag{26}$$

where B can be P, S_1, S_2, or S_3, representing the partial derivative; for S_1 it is with respect to x, for S_2 with respect to y, and for S_3 with respect to z, and Λ_l^{-1} $(l = \overline{1,46})$, where

$$\begin{aligned}
\Omega_x^r\left(x_i, y_j, z_j\right) &:= \frac{d}{dx}\Omega^r(x, y, z)\Big|_{x=x_i, y=y_j, z=z_k}, \\
\Omega_y^r\left(x_i, y_j, z_j\right) &:= \frac{d}{dy}\Omega^r(x, y, z)\Big|_{x=x_i, y=y_j, z=z_k}, \\
\Omega_z^r\left(x_i, y_j, z_j\right) &:= \frac{d}{dz}\Omega^r(x, y, z)\Big|_{x=x_i, y=y_j, z=z_k}.
\end{aligned} \tag{27}$$

Here, $\widetilde{W}_k^{i,j,k}(t) := W_k^{i,j,k}(t) - \overset{\circ}{W}{}_k^{i,j,k}$, $k = \overline{1,11}$.

The *special class* of Riccati equation

$$PA + A^T P + PRP + Q = 0 \tag{28}$$

has a unique positive solution P if and only if the four conditions given in [11] (page 65, chapter 2 *Nonlinear System Identification: Differential Learning*) are fulfilled (see [11]): (1) matrix A is stable, (2) pair $(A, R^{1/2})$ is controllable, (3) pair $(Q^{1/2}, A)$ is observable, and (4) matrices (A, Q, R) should be selected in such a way to satisfy the following inequality:

$$\frac{1}{4}(A^{\mathsf{T}}R^{-1} - R^{-1}A)R(A^{\mathsf{T}}R^{-1} - R^{-1}A)^{\mathsf{T}} + Q \le A^{\mathsf{T}}R^{-1}A \tag{29}$$

which restricts the largest eigenvalue of R guarantying the existence of a unique positive solution. The main result obtained in this part is in the practical stability framework.

4. Practical Stability and Stabilization

The following definition and proposition are needed for the main results of the paper. Consider the following ODE nonlinear system:

$$\dot{z}_t = g(z_t, \nu_t) + \varpi_t \tag{30}$$

with $z_t \in \Re^n$, and $\nu_t \in \Re^m$, and ϖ_t an external perturbation or uncertainty such that $\|\varpi_t\|^2 \le \bar{\varpi}^+$.

Definition 4 (Practical Stability). Assume that a time interval T and a fixed function $\nu_t^* \in \Re^m$ over T are given. Given $\varepsilon > 0$, the nonlinear system (30) is said to be ε-practically stable over T under the presence of ϖ_t if there exists a $\delta > 0$ (δ depends on ε and the interval T) such that $z_t \in B[0, \varepsilon]$, for all $0 \le t$, whenever $z_{t_0} \in B[0, \delta]$.

Similarly to the Lyapunov stability theory for nonlinear systems, it was applied the aforementioned direct method for the ε-practical stability of nonlinear systems using-practical Lyapunov-like functions under the presence of external perturbations and model uncertainties. Note that these functions have properties differing significantly from the usual Lyapunov functions in classic stability theory.

The subsequent proposition requires the following lemma.

Lemma 5. *Let a nonnegative function $V(t)$ satisfying the following differential inequality:*

$$\dot{V}(t) \le -\alpha V(t) + \beta, \tag{31}$$

where $\alpha > 0$ and $\beta \ge 0$. Then

$$\left[1 - \frac{\mu}{\sqrt{V(t)}}\right]_+ \longrightarrow 0, \tag{32}$$

with $\mu = \sqrt{\beta/\alpha}$ and the function $[\cdot]_+$ defined as

$$[z]_+ := \begin{cases} z & \text{if } z \ge 0, \\ 0 & \text{if } z < 0. \end{cases} \tag{33}$$

Proof. The proof of this lemma can be obtained directly by the application of the Gronwall-Bellman Lemma. □

Proposition 6. *Given a time interval T and a function $v(\cdot)$ over a continuously differentiable real-valued function $V(z,t)$ satisfying $V(0,t) = 0$, for all $t \epsilon T$, is said to be ε-practical Lyapunov-like function over T under v if there exists a constant $\alpha > 0$ such that*

$$\dot{V}(z,t) \leq -\alpha V(z,t) + H(\varpi^{+}), \tag{34}$$

with H a bounded nonnegative nonlinear function with upper bound H^{+}. Moreover, the trajectories of z_t belong to the zone $\varepsilon := H^{+}/\alpha$ when $t \to \infty$. In this proposition $\dot{V}(z_t,t)$ denotes the derivative of $V(z,t)$ along z_t, that is, $\dot{V}(z,t) = V_z(z,t) \cdot (g(z_t,v_t) + \varpi_t) + V_t(x,t)$.

Proof. The proof follows directly from Lemma 5. □

Definition 7. Given a time interval T and a function $v(\cdot)$ over T, nonlinear system (30) is ε-practically stable, T under v if there exists an ε-practical Lyapunov-like function $V(x,t)$ over T under v.

5. Identification Problem Formulation

The state identification problem for nonlinear systems (13) analyzed in this study, could be now stated as follows.

Problem. For the nonlinear system, given by the vector PDE (20), to study the quality of the DNN identifier supplied with the adjustment (learning) laws (22), estimate the upper bound of the identification error δ given by

$$\delta := \overline{\lim_{t \to \infty}} \sum_{i=0}^{L} \sum_{j=0}^{M} \sum_{k=0}^{N} \left\| \widehat{u}^{i,j,k}(t) - u^{i,j,k}(t) \right\|_{P^{i,j,k}}^{2} \tag{35}$$

(with $P^{i,j,k}$ from (24)) and, if it is possible, to reduce to its lowest possible value, selecting free parameters participating into the DNN identifier $(A, K_r, r = \overline{1,11})$.

This implicates that the reduction of the identification error δ means that the differential neural network has converged to the solution of the 3D PDE; this can be observed in the matching of the DNN to the PDE state.

6. Main Result

The main result of this paper is presented in the following theorem.

Theorem 8. *Consider the nonlinear model* (1) $(i = 0,\ldots,L;\ j = 0,\ldots,M,\ k = 0,\ldots,N)$, *given by the system of PDEs with uncertainties (perturbations) in the states, under the border conditions* (2). *Let us also suppose that the DNN-identifier is given by* (20) *which parameters are adjusted by the learning laws* (22) *with parameters* $\alpha_m^{i,j,k} (i = 0,\ldots,L; j = 0,\ldots,M; k = 0,\ldots,N)$. *If positive definite matrices* $Q_1^{i,j,k}$, $Q_2^{i,j,k}$, *and* $Q_3^{i,j,k}$ *provide the existence of positive solutions* $P^{i,j}$, $S_1^{i,j,k}$, $S_2^{i,j,k}$, *and*

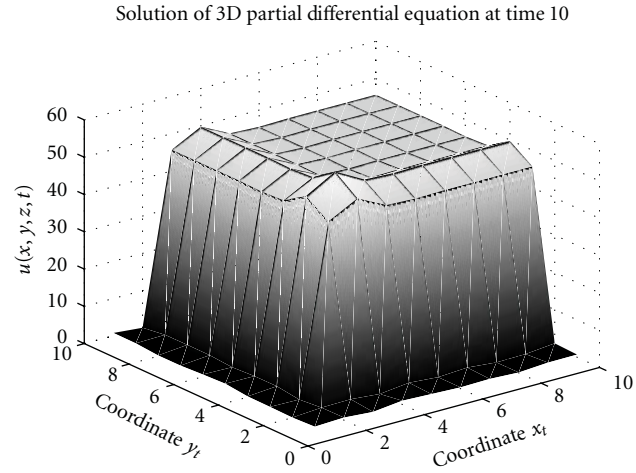


Figure 2: Numerical trajectory produced by the mathematical model described by 3D partial differential equation at time 10 s along the whole domain.

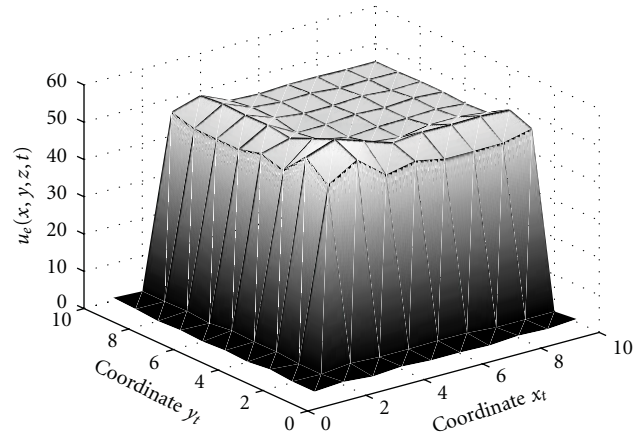


Figure 3: Numerical trajectory produced by the neural network described reproducing the model at time 10 s along the whole domain.

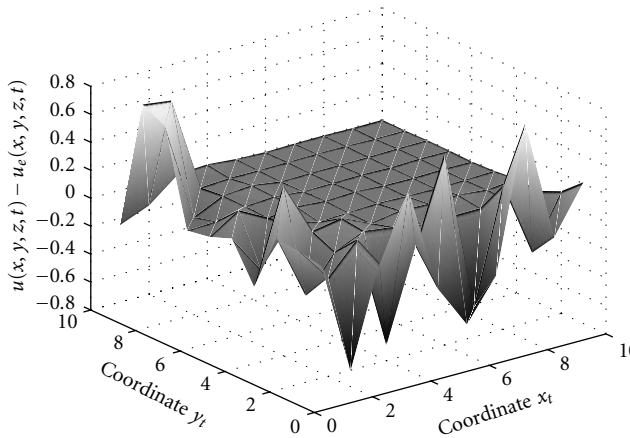


Figure 4: Error of the approximation. Difference between 3D PDE numerical solution and 3D DNN approximation at time 10 s along the whole domain.

$S_3^{i,j,k}(i = 0,\dots,L)$ *to the Riccati equations* (24)*, then for all* $i = 0,\dots,L$*;* $j = 0,\dots,M$*;* $k = 0,\dots,N$ *the following* ρ*-upper bound:*

$$\begin{aligned}
&\overline{\lim_{t\to\infty}} \sum_{i=0}^{L}\sum_{j=0}^{M} \\
&\quad\times \sum_{k=0}^{N}\Big[\Big|\Big|\widehat{u}^{i,j,k}(t) - u^{i,j,k}(t)\Big|\Big|^2_{P^{i,j,k}} + \Big|\Big|\widehat{u}_x^{i,j,k}(t) - u_x^{i,j,k}(t)\Big|\Big|_{S_1^{i,j,k}} \\
&\qquad + \Big|\Big|\widehat{u}_y^{i,j,k}(t) - u_y^{i,j,k}(t)\Big|\Big|_{S_2^{i,j,k}} + \Big|\Big|\widehat{u}_z^{i,j,k}(t) - u_z^{i,j,k}(t)\Big|\Big|_{S_3^{i,j,k}}\Big] \\
&\quad \le \mu
\end{aligned} \tag{36}$$

is ensured with $\mu_0 := \sqrt{LMN}\alpha^{-1/2}$ *and*

$$\begin{aligned}
\mu &:= \mu_0\sqrt{\max_{i,j,k}\Psi}, \\
\Psi &:= \varpi\sum_{s=1}^{3} f_s^{i,j,k} \\
&\ge \left(\widetilde{f}^{i,j,k}(t)\right)^T \Lambda_7^{-1}\widetilde{f}^{i,j,k}(t) + \left(\widetilde{f}_x^{i,j,k}(t)\right)^T \Lambda_{14}^{-1}\widetilde{f}_x^{i,j,k}(t) \\
&\quad + \left(\widetilde{f}_y^{i,j,k}(t)\right)^T \Lambda_{21}^{-1}\widetilde{f}_y^{i,j,k}(t) + \left(\widetilde{f}_z^{i,j,k}(t)\right)^T \Lambda_{28}^{-1}\widetilde{f}_z^{i,j,k}(t), \\
\varpi &:= \max\{\lambda_{\max}(\Lambda_7^{-1}), \lambda_{\max}(\Lambda_{14}^{-1}), \lambda_{\max}(\Lambda_{21}^{-1}), \lambda_{\max}(\Lambda_{28}^{-1})\}.
\end{aligned} \tag{37}$$

Moreover, the weights $W_{r,t} r = \overline{1,11}$ *remain bounded being proportional to* μ*, that is,* $\|W_{r,t}\| \le K_r\mu$*,* $r = \overline{1,11}$.

Proof. The proof is given in the appendix. □

7. Simulation Results

7.1. Numerical Example. Below, the numerical smulation shows the qualitative illustration for a benchmark system. Therefore, consider the following three-dimensional PDE described as follows:

$$\begin{aligned}
u_t(x,y,z,t) &= -c_1 u_{xx}(x,y,z,t) - c_2 u_{yy}(x,y,z,t) \\
&\quad - c_3 u_{zz}(x,y,z,t) + \xi(x,y,z,t),
\end{aligned} \tag{38}$$

where $c_1 = c_2 = c_3 = 0.15$. It is assumed that there is access to discrete measurements of the state $u(x,y,z,t)$ along the whole domain, which is feasible in practice. $\xi(x,y,z,t)$ is a noise in the state dynamics. This model will be used just to generate the data to test the 3D identifier based on DNN. Boundary conditions and initial conditions were selected as follows:

$$\begin{aligned}
u(0,0,0,t) &= \text{rand}(1), \qquad u_x(0,0,0,t) = 0, \\
u_y(0,0,0,t) &= 0, \qquad u_z(0,0,0,t) = 0.
\end{aligned} \tag{39}$$

The trajectories of the model can be seen in Figure 3 as well as the estimated state, produced by the DNN identifier. The efficiency of the identification process provided by the suggested DNN algorithm shown in Figure 4.

In Figures 5 and 6 there are shown the trajectories of the PDE and the DNN for the coordinate z and the index error of the PDE and the DNN at 10 seconds for coordinates x, z, respectively Figure 2.

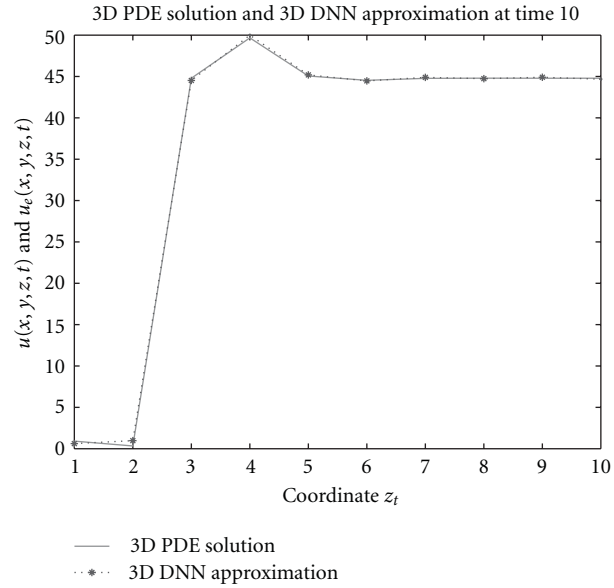


Figure 5: Comparison between 3D PDE trajectories and 3D DNN approximation for coordinate z at time 10 s.

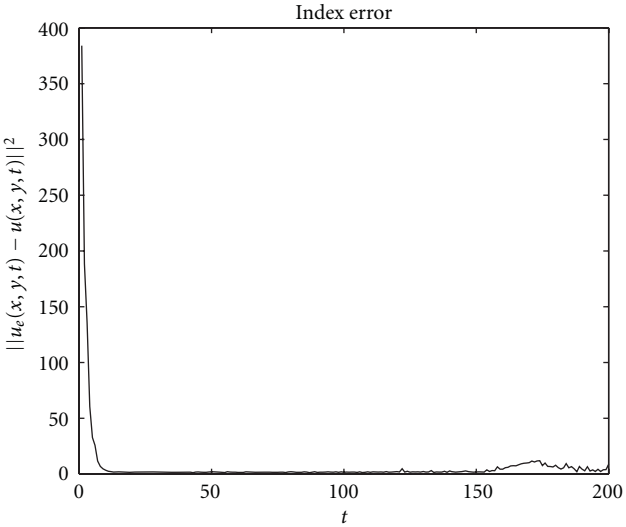


Figure 6: Index quadratic error for the whole time with coordinates x, y, and z fixed.

7.2. Tumour Growth Example. The mathematical model of the brain tumour growth is presented in this section based on the results of [18]. Here the diffusion coefficient is considered as a constant. Let consider the following three-dimensional

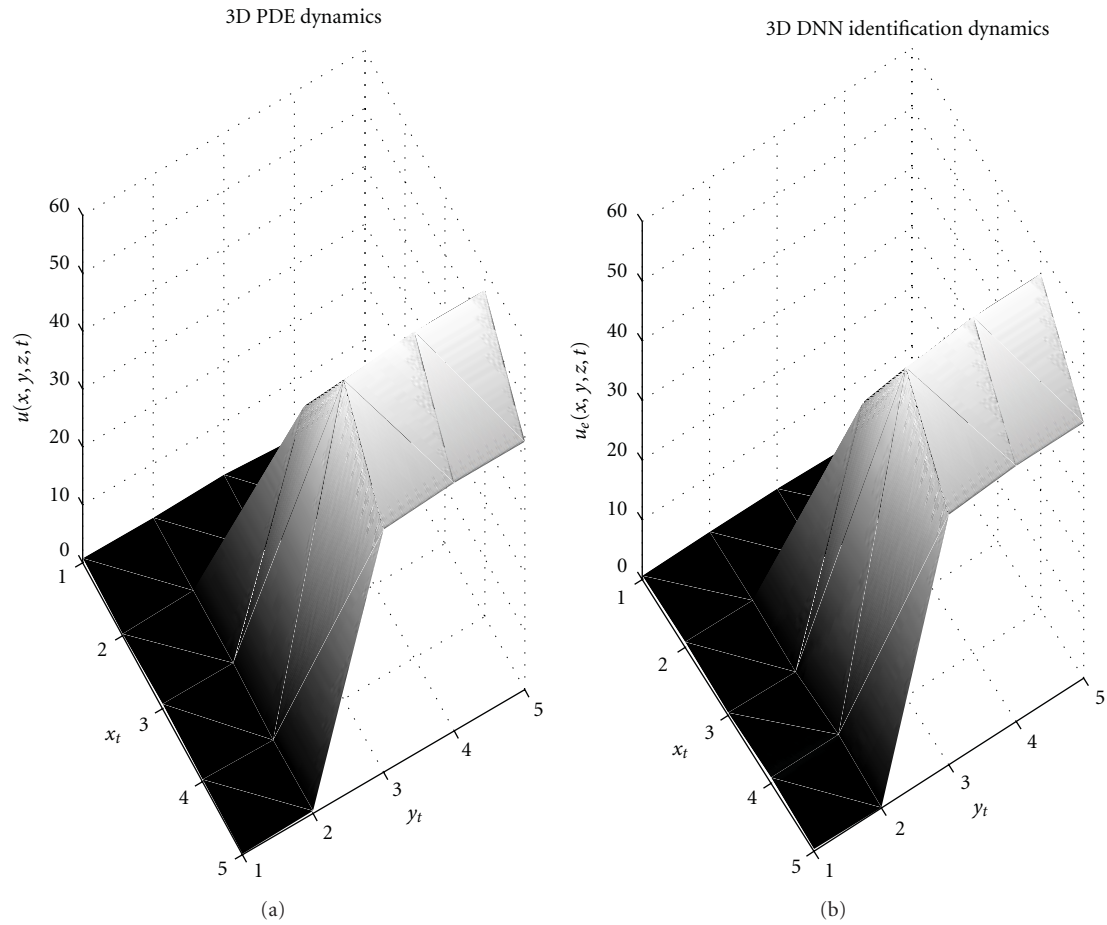


Figure 7: Numerical trajectory produced by the mathematical model described by 3D partial differential equation at time 46 days along the whole domain.

parabolic equation of the tumour growth described as follows:

$$\begin{aligned} u_t(x,y,z,t) = {} & -Pu_x(x,y,z,t) - Ru_y(x,y,z,t) \\ & - Su_z(x,y,z,t) + Qu_{xx}(x,y,z,t) \\ & + Qu_{yy}(x,y,z,t) + Qu_{zz}(x,y,z,t) \\ & + \Gamma - Lu(x,y,z,t), \end{aligned} \tag{40}$$

where $u(x, y, z, t)$ is the growth rate of a brain tumour, Q is the diffusion coefficient, $W = (P, R, S)$ is the drift velocity field, $\Gamma = \Gamma(u)$ is the proliferation coefficient, and $L = L(u)$ is the decay coefficient of cells. It is assumed that there is access to discrete measurements of the state $u(x, y, z, t)$ along the whole domain, which is feasible in practice by PET-CT (Positron emission tomography-computed tomography) technology. The domain $0 \le x \le 1$, $0 \le y \le 1$, and $0 \le z \le 1$ This model will be used just to generate the data to test the 3D identifier based on DNN. Boundary conditions and initial conditions were selected as follows:

$$\begin{aligned} u(0,0,0,t) &= 200 \pm 20\mu, \qquad u_x(0,0,0,t) = 0, \\ u_y(0,0,0,t) &= 0, \qquad u_z(0,0,0,t) = 0. \end{aligned} \tag{41}$$

The trajectories of the model and the estimate state produced by the DNN identifier can be seen in Figure 7. The dissimilarity between both trajectories depends on the *learning period* required for adjusting the DNN identifier. The error between trajectories produced by the model and the proposed identifier is close to zero almost for all x, y, z and all t that shows the efficiency of the identification process provided by the suggested DNN algorithm is shown in Figure 8.

In Figures 9 and 10 there are shown the trajectories of the PDE and the DNN for the coordinate z and the index error of the PDE and the DNN at 46 days for coordinates x, z, respectively.

8. Conclusions

The adaptive DNN method proposed here solves the problem of non-parametric identification of nonlinear systems (with uncertainties) given by 3D uncertain PDE. Practical stability for the identification process is demonstrated based on the Lyapunov-like analysis. The upper bound of the identification error is explicitly constructed. Numerical examples demonstrate the estimation efficiency of the suggested methodology.

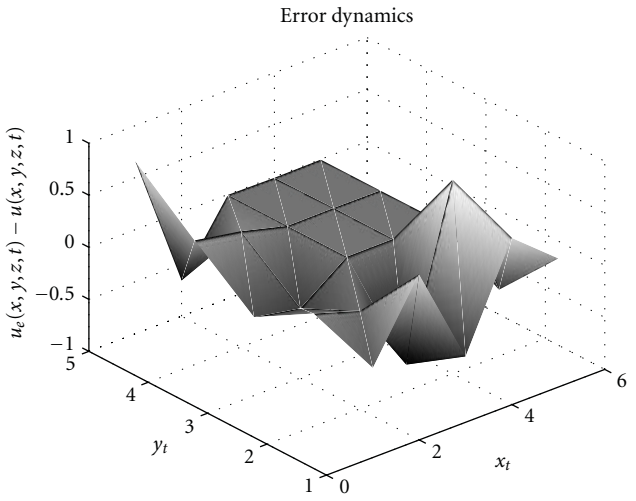


Figure 8: Error of the approximation. Difference between 3D PDE numerical solution and 3D DNN approximation at time 46 days along the whole domain.

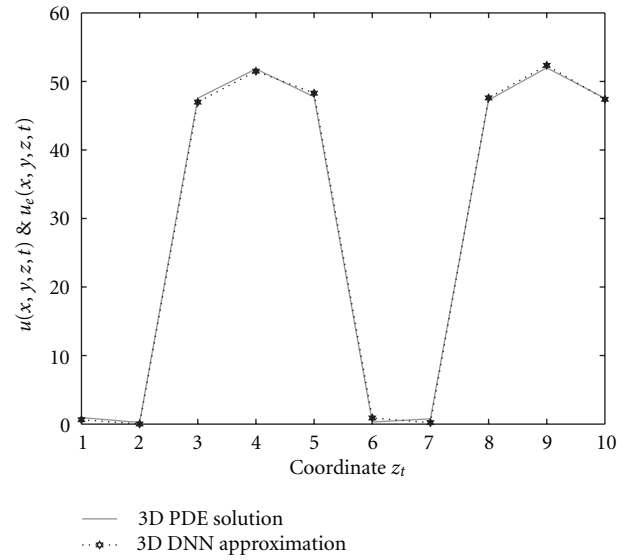


Figure 9: Comparison between 3D PDE trajectories and 3D DNN approximation for coordinate z at time 46 days.

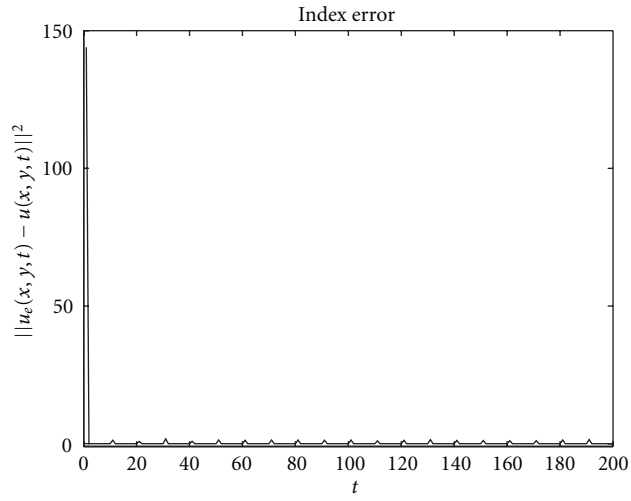


Figure 10: Index quadratic error for the whole time with coordinates x, y, and z fixed.

Appendix

Consider the Lyapunov-like function defined as the composition of NML individual Lyapunov functions $\overline{V}_{i,j,k}(t)$ along the whole space:

$$V(t) := \sum_{i=0}^{L}\sum_{j=0}^{M}\sum_{k=0}^{N}\left[\overline{V}^{i,j,k}(t) + \sum_{r=1}^{11}\mathrm{tr}\left\{\left[\widetilde{W}_r^i(t)\right]^T K_r \widetilde{W}_r^i(t)\right\}\right],$$

$$\overline{V}^{i,j,k}(t) := \left\|\widetilde{u}^{i,j,k}(t)\right\|^2_{P^{i,j,k}} + \left\|\widetilde{u}_x^{i,j,k}(t)\right\|^2_{S_1^{i,j,k}} + \left\|\widetilde{u}_y^{i,j,k}(t)\right\|^2_{S_2^{i,j,k}} + \left\|\widetilde{u}_z^{i,j,k}(t)\right\|^2_{S_3^{i,j,k}}. \tag{A.1}$$

The time derivative $\dot{V}(t)$ of $V(t)$ can be obtained so that

$$\begin{aligned}\dot{V}(t) = {} & 2\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left(\widetilde{u}^{i,j,k}(t)\right)^T(t)P^{i\cdot j,k}\frac{d}{dt}\widetilde{u}^{i,j,k}(t) \\ & + 2\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left[\widetilde{u}_x^{i,j,k}(t)\right]^T S_1^{i\cdot j,k}\frac{d}{dt}\widetilde{u}_x^{i.j,k}(t) \\ & + 2\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left[\widetilde{u}_y^{i,j,k}(t)\right]^T S_2^{i\cdot j,k}\frac{d}{dt}\widetilde{u}_y^{i.j,k}(t) \\ & + 2\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left[\widetilde{u}_z^{i,j,k}(t)\right]^T S_3^{i\cdot j,k}\frac{d}{dt}\widetilde{u}_z^{i.j,k}(t) \\ & + 2\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\sum_{r=1}^{3}\mathrm{tr}\left\{\left[\widetilde{W}_r^{i,j,k}(t)\right]^T K_r \dot{W}_r^{i,j,k}(t)\right\}.\end{aligned} \tag{A.2}$$

Applying the matrix inequality given in [19]

$$XY^T + YX^T \le X\Lambda X^T + Y\Lambda^{-1}Y^T \tag{A.3}$$

valid for any $X, Y \in R^{r\times s}$ and for any $0 < \Lambda = \Lambda^T \in R^{s\times s}$ to the terms containing $\widetilde{f}^i(t)$ and their derivatives. We obtain

$$\begin{aligned}\dot{V}(t) \le {} & I_1(t) + I_2(t) + I_3(t) + I_4(t) \\ & - 2\alpha\sum_{r=1}^{11}\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\mathrm{tr}\left\{\left(\widetilde{W}_r^{i,j}(t)\right)^T K_r \overline{W}_r^{i,j,k}(t)\right\} \\ & + 2\alpha\sum_{r=1}^{11}\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\mathrm{tr}\left\{\left(\widetilde{W}_r^{i,j}(t)\right)^T K_r \overline{W}_r^{i,j,k}(t)\right\} \\ & + 2\sum_{r=1}^{11}\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\mathrm{tr}\left\{\left(\widetilde{W}_r^{i,j}(t)\right)^T K_r^2 \dot{W}_1^{i,j,k}(t)\right\} \\ & - \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\alpha\overline{V}^{i,j,k}(t) + \varpi\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\sum_{s=1}^{3}f_s^{i,j,k},\end{aligned} \tag{A.4}$$

where

$$I_1(t) := \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left(\widetilde{u}^{i.j,k}(t)\right)^T \mathrm{Ric}\left(P^{i,j}\right)\widetilde{u}^{i,j,k}(t),$$

$$I_2(t) := \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left(\widetilde{u}_x^{i.j,k}(t)\right)^T \mathrm{Ric}\left(S_1^{i,j,k}\right)\widetilde{u}_x^{i,j,k}(t),$$

$$I_3(t) := \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left(\widetilde{u}_y^{i.j,k}(t)\right)^T \mathrm{Ric}\left(S_2^{i,j}\right)\widetilde{u}_y^{i,j,k}(t),$$

$$I_4(t) := \sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\left(\widetilde{u}_z^{i.j,k}(t)\right)^T \mathrm{Ric}\left(S_3^{i,j}\right)\widetilde{u}_z^{i,j,k}(t). \tag{A.5}$$

By the Riccati equations, defined in (24), $I_1(t) = I_2(t) = I_3(t) = I_4(t) = 0$, and in view of the adjust equations of the weights (22), the previous inequality is simplified to

$$\begin{aligned}\dot{V}(t) \le{}& -\alpha\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\overline{V}^{i,j,k}(t)\\ &+ 2\alpha\sum_{r=1}^{11}\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}\mathrm{tr}\left\{\left(\widetilde{W}_r^{i,j}(t)\right)^T K_r\overline{W}_r^{i,j,k}(t)\right\}\\ &+ \Psi\dot{V}(t) \le -\alpha\sum_{i=0}^{N}\sum_{j=0}^{M}\sum_{k=0}^{L}V^{i,j,k}(t) + \Psi.\end{aligned} \tag{A.6}$$

Applying Lemma 5, one has $[1 - (\mu/\sqrt{V(t)})]_+ \to 0$, which completes the proof.

References

[1] Y. Pinchover and J. Rubinstein, *An Introduction to Partial Differential Equations*, Cambridge University Press, 2008.

[2] G. D. Smith, *Numerical Solution of Partial Differential Equations: Infinite Difference Methods*, Clarendon Press, Oxford, UK, 1978.

[3] T. J. R. Hughes, *The Finite Element Method*, Prentice Hall, Upper Saddle River, NJ, USA, 1987.

[4] R. Fuentes, A. Poznyak, T. Poznyak, and I. Chairez, "Neural numerical modeling for uncertain distributed parameter system," in *Proceedings of the International Joint Conference in Neural Networks*, pp. 909–916, Atlanta, Ga, USA, June 2009.

[5] S. Haykin, *Neural Networks. A Comprehensive Foundation*, IEEE Press, Prentice Hall U.S., New York, NY, USA, 2nd edition, 1999.

[6] G. Cybenko, "Approximation by superposition of sigmoidal activation function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[7] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[8] M. W.M Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Communications in Numerical Methods in Engineering*, vol. 10, no. 3, pp. 195–201, 1994.

[9] N. Mai-Duy and T. Tran-Cong, "Numerical solution of differential equations using multiquadric radial basis function networks," *Neural Networks*, vol. 14, no. 2, pp. 185–199, 2001.

[10] S. He, K. Reif, and R. Unbehauen, "Multilayer neural networks for solving a class of partial differential equations," *Neural Networks*, vol. 13, no. 3, pp. 385–396, 2000.

[11] A. Poznyak, E. Sanchez, and W. Yu, *Differential Neural Networks for Robust Nonlinear Control (Identification, Estate Estimation an trajectory Tracking)*, World Scientific, 2001.

[12] F. L. Lewis, A. Yeşildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 1–11, 1996.

[13] G. A. Rovithakis and M. A. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 3, pp. 400–412, 1994.

[14] H. K. Khalil, *Nonlinear Systems*, Prentice-Hall, Upper Saddle River, NJ, USA, 2002.

[15] R. A. Adams and J. Fournier, *Sobolev Spaces*, Academic Press, New York, NY, USA, 2nd edition, 2003.

[16] B. Delyon, A. Juditsky, and A. Benveniste, "Accuracy analysis for wavelet approximations," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 332–348, 1995.

[17] N. E. Cotter, "The Stone-Weierstrass theorem and its application to neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 290–295, 1990.

[18] M. R. Islam and N. Alias, "A case study: 2D Vs 3D partial differential equation toward tumour cell visualisation on multicore parallel computing atmosphere," *International Journal for the Advancement of Science and Arts*, vol. 10, no. 1, pp. 25–35, 2010.

[19] A. Poznyak, *Advanced Mathetical Tools for Automatic Control Engineers, Deterministic Technique*, vol. 1, Elsevier, 2008.