Machine Learning, 23, 191–220 (1996)
© 1996 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Performance Improvement of Robot Continuous-Path Operation through Iterative Learning Using Neural Networks

PETER C.Y. CHEN[†], JAMES K. MILLS[†] AND KENNETH C. SMITH[†‡] chencyp@me.utoronto.ca
[†]*Department of Mechanical Engineering, University of Toronto, 5 King's College Road, Toronto, Ontario, Canada M5S 1A4*

[‡]*Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 1A4*

Editors: Judy A. Franklin, Tom M. Mitchell, and Sebastian Thrun

Abstract. In this article, an approach to improving the performance of robot continuous-path operation is proposed. This approach utilizes a multilayer feedforward neural network to compensate for model uncertainty associated with the robotic operation. Closed-loop stability and performance are analyzed. It is shown that the closed-loop system is stable in the sense that all signals are bounded; it is further proved that the performance of the closed-loop system is improved in the sense that certain error measure of the closed-loop system decreases as the network learning process is iterated. These analytical results are confirmed by computer simulation. The effectiveness of the proposed approach is demonstrated through a laboratory experiment.

Keywords: robot control, neural networks, uncertainty compensation, stability and performance

1. Introduction

In many industrial operations, a robot is required to follow a continuous path accurately. An example of this type of operation is arc welding, where the end-effector of the robot is required to follow a prescribed path with a prescribed velocity. Currently, industrial robots for continuous-path operation (also known as trajectory tracking) are "programmed" by the so-called "walk-through" method, where an operator physically guides the robot through a sequence of desired movements which is then stored in the controller's memory. Such a programming method is time consuming and uneconomical, because during the walk-through, the robot is out of production activity.

An alternative to the walk-through method is the so-called "off-line programming" method, where a high-level programming language is used to write a control program which specifies actions of each actuator that would produce the desired motion of the robot. Currently industrial robots are PID-controlled. (A PID controller is a controller with three terms in which the output of the controller is the sum of a proportional term, an integral term, and a differentiating term, with adjustable gain for each term (Dorf, 1992).) Off-line programming for continuous-path operation based on PID control may not produce the desired robot motion for the following reason. Since the robot dynamics is nonlinear while PID control is a linear-control method, applying PID control to the trajectory tracking problem would require a gain scheduling approach using local models; that is, the robot dynamics is linearized about some operating points so that the control gains can be selected to achieve certain performance specification. The trajectory, however, varies with time (as opposed to a set-point), therefore the gains so selected may not be appropriate due to the linearization. As a result, significant tracking error may occur.

A technique known as "computed torque" (Spong & Vidyasagar, 1989) has been shown to be an effective alternative to PID control under a condition called "exact cancellation of nonlinearity". In the computed torque method, control actions are generated based on a mathematical model of the robot. Exact cancellation of nonlinearity would result in a linear decoupled system, but it requires that the parameters in the dynamics model of the robot be known exactly. Failure to meet this condition leads to certain undesirable signals called "uncertainty", which must be compensated in order to improve the performance of the robot in continuous-path operation.

Investigations dealing with uncertainty reported in the robotics literature can in principle be classified under the following two approaches: robust control and adaptive control. The premise of robust control (Corless & Letimann, 1981) is that although the uncertainty is unknown, it is possible to estimate the "worst case" bounds on its effect on the tracking performance of the manipulator. The robust control law is designed with the objective to overcome the effect of the uncertainty (rather than to "cancel" the uncertainty so that a linear decoupled system is obtained). In the adaptive control approach (Ortega & Spong, 1989), the basic premise is that by changing the values of gains or other parameters in the control law according to some on-line algorithm, the controller can find a set of values for these gains or parameters so that the trajectory tracking error is reduced. Stability analysis of these approaches often makes use of the Second Method of Lyapunov, which guarantees that the tracking error will be reduced to zero or a small neighborhood of zero as time goes to infinity. Other than in the context of exponential stability, which is much more difficult to obtain, Lyapunov stability generally provides no clear insight about the transient performance of the manipulator (Spong & Vidyasagar, 1989).

A class of computational models known as neural networks has been applied to system control in general and to robot control in particular, e.g., (Miller, Sutton & Werbos, 1990). (The use of the word "neural" to describe such computational models stems solely from modern convention. Although their structure may have been derived from neuronal models of the central nervous system, the computational models discussed in this article are, at most, only mathematical abstractions of biological neuronal systems.) Justification for using neural networks for robot control is based on the following properties of neural networks: (i) The ability of the neural network to "learn" (through a repetitive training process) (McClelland & Rumelhart, 1986) enables a controller incorporated with a neural network to improve its performance. (ii) The ability of the neural network to "generalize" what it has learned (Denker, et al., 1987) enables the controller also to respond to unexpected situations. (iii) The structure of neural networks allows massive parallel processing, especially when the neural networks are implemented in hardware using VLSI technology (Gelsner & Pöchmüller, 1994); such inherent collective processing capability enables the neural network to respond quickly in generating timely control actions.

It is due to these properties that a neural-network-based approach to uncertainty compensation could be considered potentially advantageous over both robust control and adaptive control. The learning ability of neural networks is especially desirable in controlling robots that perform repetitive manufacturing tasks. One reason for using robots instead of human workers in manufacturing is that robots can perform repetitive tasks with better quality and consistency. Unavoidable in repetitive robotic operation in an industrial setting, however, is the sustained "wear-and-tear" (e.g., joint friction, wear of gears, etc.) of the robot. Such wear-and-tear inevitably effects the dynamic characteristics of the robot. In other words, the wear-and-tear introduces uncertainty into the robotic system, and consequently degrades its performance. A neural network that learns (iteratively) to compensate for the effect of such wear-and-tear would enable the manipulator to maintain satisfactory performance consistently throughout its expected lifetime.

In this article, we propose an approach to robot trajectory tracking using a multilayer feedforward neural network. (In the sequel, the term "neural network" or just "network" is sometimes used instead of "multilayer feedforward neural network" for convenience.) The neural network is used explicitly to compensate for the uncertainty in the manipulator. We address the dynamical behavior of the manipulator in a two-step analysis. First we show that the closed-loop system based on the proposed approach is stable in the sense that all signals in the system are bounded. We then show that the neural network improves the performance of the robot in the sense that certain error measure of the closed-loop system is reduced as the learning process of the neural network is iterated. We subsequently present simulation results that confirm the analytical conclusions, and experimental results that demonstrate the effectiveness of the proposed approach.

The contributions of this work to the application of neural networks to robot control are: (i) The insight obtained (through the analysis on the dynamics of the neural network) on the stability and performance of the closed-loop system with the neural network learning on-line is significant. The results of the analysis confirm that neural networks could be used as plausible tools for robot control within the context of uncertainty compensation. (ii) The experimental implementation of the proposed approach together with the positive experimental results reported in this article clearly demonstrate the effectiveness of the neural network as an uncertainty compensator for practical robotic tasks. Many studies on the application of neural networks to system control in general and to robot control in particular rely on numerical simulation to verify the conclusions therein; very few proposed schemes have been physically implemented to verify their effectiveness. Extensive and conclusive experimental results are needed in order to affirm neural networks as viable tools for robot control. The experiment reported in this article represents an incremental step in gathering such results.

This article is organized as follows. Section 2 reviews the literature on the application of neural networks to robot trajectory tracking. Section 3 formulates the dynamics of the robotic system and presents the control law that incorporates a compensating signal from the neural network. Section 4 shows that a compensator exists in the form of a multilayer feedforward neural network, and describes the algorithm used for neural network learning. Section 5 presents analysis of stability and performance. Section 6 describes the computer simulation conducted to verify the analytical conclusions, and presents the results. Section 7 describes the experimental implementation of the proposed approach, and presents the experimental results. Section 8 discusses the implications of the results and suggests directions for future research.

2. Literature Review

Various robot control techniques using neural networks reported in the literature can in principle be classified into two general schemes according to the role of the neural network. Figure 1 illustrates the first scheme.

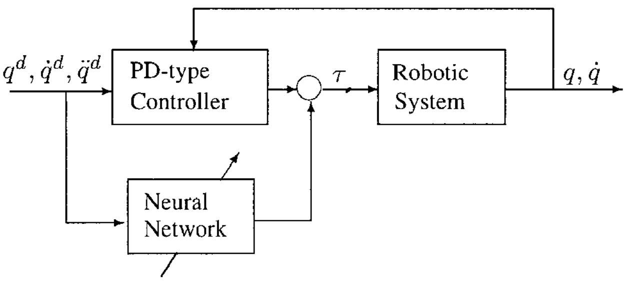

Figure 1. Neural Network as Inverse Dynamics Model.

In this first scheme, the neural network is to repetitively learn to represent the inverse dynamics of the manipulator, so that by feeding the desired trajectory into the neural network, the desired torque signal is produced as the output of the network. The PD controller is used mainly to stabilize the closed-loop system and to guide the repetitive learning process of the neural network. (A PD controller is a controller with two terms in which the output of the controller is the sum of a proportional term and a differentiating term, with adjustable gain for each term (Dorf, 1992).) Gomi and Kawato (1990) use this scheme for robot trajectory tracking, and present computer simulation results involving a two-link manipulator. Cılız and Işık (1990) utilize this scheme to control a manipulator under payload variation. Yabuta and Yamada (1991) apply this scheme (without the PD controller) to manipulator control, and demonstrate its effectiveness using an one degree-of-freedom servomechanism. Arai, Rong and Fukuda (1993) study the possibility of increasing the speed of the learning process.

In the second scheme, illustrated in Figure 2, the neural network is used to deal with uncertainty in the model parameters. The inverse dynamics control law is used to generate an approximate torque signal. This torque signal is then augmented by a compensating signal generated by a neural network. The neural network is to learn to generate the proper compensating signal by adjusting its weights so as to maximize some performance measure, such as reduction of the tracking error. Okuma and Ishiguro (1990) apply this scheme to the control of a manipulator with consideration of joint friction, and present results of computer simulation involving a two-link manipulator. Kuan and Bavarian (1992) also study, again through computer simulation, the problem of dealing with joint friction using this scheme. Zomaya and Nabhan (1993) apply this scheme, without the PD controller, to control a manipulator, and conduct computer simulation using the dynamics model of a *PUMA* 560 robot.

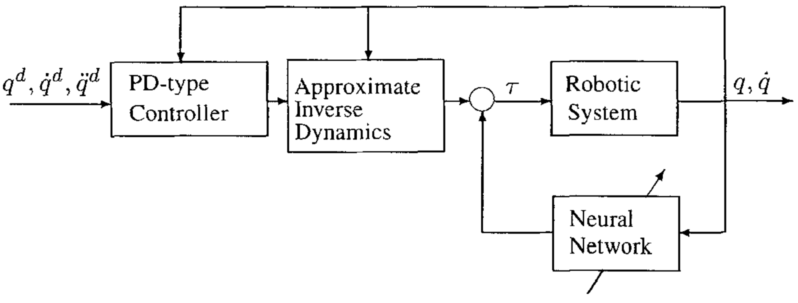

Figure 2. Neural Network as Compensator.

Although attempts have been made, among the studies cited above, to analyze the stability and performance of these schemes, the results reported remain inconclusive, mainly because they are obtained based on very restrictive assumptions. It appears that in the current literature on the application of neural networks to robot control, the issues of closed-loop stability and performance have not been resolved. The importance of resolving these issues is underscored by the fact that, as part of the control system, the dynamics of the neural network is coupled with that of the manipulator; consequently, the dynamical behavior of the neural network, namely the process of weight adjustment, will inevitably affect the stability as well as the performance of the closed-loop system.

It also appears that many studies on the application of neural networks to robot control rely on numerical simulation to verify the conclusions therein; very few proposed schemes have been physically implemented to verify their effectiveness.

It is in the context of these observations that the work described in this article complements other studies reported in the literature. In this article, we show that the closed-loop system based on the proposed approach is stable and that the neural network improves the performance of the closed-loop system through iterative learning. Our analytical results confirm that neural networks can be used as effective tools to improve the performance of a manipulator in continuous-path operation. We subsequently present the results of the experimental implementation of our proposed approach involving a laboratory manipulator. The experimental results clearly demonstrate the effectiveness of the neural network in improving the performance of the robotic system. In short, the work presented in this article complements other published studies both in analytical aspect and in experimental aspect.

3. Dynamics and Control

In general, the dynamics of a manipulator with n joints can be described by a set of nonlinear differential equations, compactly expressed in the form (Spong & Vidyasagar, 1989)

$$M(q)\ddot{q} + h(q, \dot{q}) = \tau, \tag{1}$$

where $q \in \mathcal{R}^n, \dot{q} \in \mathcal{R}^n$, and $\ddot{q} \in \mathcal{R}^n$ are respectively the joint position, joint velocity, and joint acceleration vectors, $M(\cdot) \in \mathcal{R}^{n \times n}$ is the inertia matrix, $h(\cdot) \in \mathcal{R}^n$ is a vector containing the Coriolis, gravitational, centrifugal and frictional terms, and $\tau \in \mathcal{R}^n$ is the input torque vector. With regard to an industrial robot, the following characteristics usually apply: (i) the manipulator is composed of serial links; (ii) the manipulator is not redundant; and (iii) the links and joints of the manipulator are rigid.

Supposing that the terms $M(\cdot)$ and $h(\cdot)$ are known precisely, the inverse dynamics control law can be written as

$$\tau = M(q)u + h(q, \dot{q}), \tag{2}$$

where u is a control input to be specified. Substituting (2) into (1) yields: $\ddot{q} = u$. Let u be a PD-type control of the form

$$u = \ddot{q}^d + K_v(\dot{q}^d - \dot{q}) + K_p(q^d - q), \tag{3}$$

where $q^d \in \mathcal{R}^n$, $\dot{q}^d \in \mathcal{R}^n$, and $\ddot{q}^d \in \mathcal{R}^n$ are respectively the desired joint position, joint velocity, and joint acceleration vectors, and $K_v \in \mathcal{R}^{n \times n}$ and $K_p \in \mathcal{R}^{n \times n}$ are diagonal gain matrices. Let $e = \begin{bmatrix} q^d - q \\ \dot{q}^d - \dot{q} \end{bmatrix}$ and $A = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}$. Then the error dynamics of the closed-loop system can be expressed as

$$\dot{e} = Ae, \tag{4}$$

which represents a linear decoupled system whose dynamical behavior can be completely specified by selecting appropriate values for the gains K_v and K_p (Dorf, 1992).

The inverse dynamics control law (2) is idealized because it results in a linear decoupled system only if the the values of the parameters in $M(\cdot)$ and $h(\cdot)$ are known precisely. In practice, such precise knowledge about a physical system is not available. Thus the realistic inverse dynamics control law takes the form

$$\tau = \hat{M}(q)u + \hat{h}(q, \dot{q}), \tag{5}$$

where $\hat{M}(\cdot)$ and $\hat{h}(\cdot)$ are respectively the estimates of $M(\cdot)$ and $h(\cdot)$. Now with (3) and (5), the resulting closed-loop error dynamics becomes

$$\dot{e} = Ae + B\eta(q, \dot{q}, \ddot{q}), \tag{6}$$

where $B = [0, I]^T$, and $\eta(q, \dot{q}, \ddot{q}) = (\hat{M}^{-1}M - I)\ddot{q} + \hat{M}^{-1}(h - \hat{h})$. Note that for brevity, the arguments of M, h, $\hat{M}$, and $\hat{h}$ have been omitted.

The system (6) is exactly the same as the linear decoupled system (4) if the term $\eta(\cdot)$ is identically zero. This term exists whenever $\hat{M}(\cdot) \neq M$ and/or $\hat{h}(\cdot) \neq h(\cdot)$, and is referred to as the *uncertainty*. Due to the presence of η, the dynamical behavior of the system (6) can no longer be "shaped" as desired by simply selecting appropriate values for the gains K_v and K_p.

To reduce the effect of the uncertainty, a compensating signal v is introduced as part of the control signal u as follows

$$u = \ddot{q}^d + K_v(\dot{q}^d - \dot{q}) + K_p(q^d - q) + v. \tag{7}$$

Substituting (5) and (7) into (1) yields the error dynamics of the closed-loop system

$$\dot{e} = Ae + B\Delta v, \tag{8}$$

where $\Delta v = \eta(q, \dot{q}, \ddot{q}) - v$ is referred to as the *control error*. If v is generated such that $\Delta v \rightarrow 0$, then (8) becomes: $\dot{e} = Ae$, which is again the linear decoupled system (4). The benefit gained through uncertainty compensation (i.e., $\Delta v \rightarrow 0$) is that, because the robotic system is rendered linear and decoupled even in the presence of the uncertainty η, the dynamics of the robotic system can now be controlled to meet specific performance requirements by selecting appropriate values for the gains. Figure 3 schematically depicts the closed-loop system.

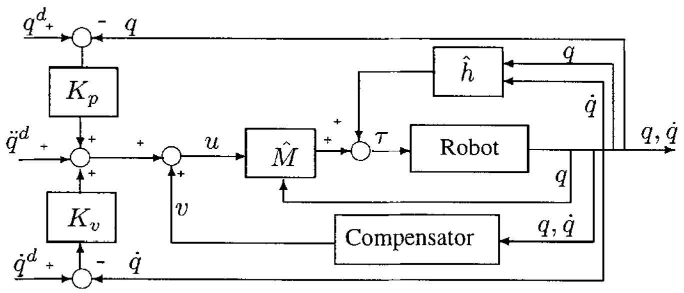

Figure 3. Uncertainty Compensation in Robot Control.

4. Uncertainty Compensation

The objective of uncertainty compensation is to generate the appropriate compensating signal v such that the control error Δv vanishes, i.e., $\Delta v \rightarrow 0$. An ideal compensator is a function whose output v exactly equals that of the function $\eta(\cdot)$. Based on such a premise, the problem of uncertainty compensation can then be considered as a function approximation problem. A multilayer feedforward neural network represents an attractive mechanism for dealing with such a problem, because it is capable of approximating any continuous function, and more importantly, it can *learn* to approximate any given continuous function (McClelland & Rumelhart, 1986).

A multilayer feedforward neural network consists of a collection of processing elements (or units) arranged in a layered structure (McClelland & Rumelhart, 1986) as shown in Figure 4.

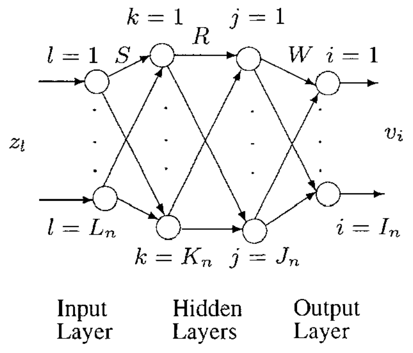

Figure 4. Neural Network Structure.

For the neural network with two hidden layers, as depicted in Figure 4, the mapping realized by the network can be described as follows. Let the number of units in the input layer, the first hidden layer, the second hidden layer, and the output layer be L_n, K_n, J_n, and I_n respectively. Let $\bar{\bar{r}}_k, \bar{r}_j$, and r_i represent the input to the first hidden layer, the second hidden layer, and the output layer respectively, and let $\bar{\bar{v}}_k, \bar{v}_j$ and v_i represent the corresponding output of these layers, then $\bar{\bar{r}}_k = \sum_{l=1}^{L_n} S_{kl} z_l$, $\bar{\bar{v}}_k = g(\bar{\bar{r}}_k)$, $\bar{r}_j = \sum_{k=1}^{K_n} R_{jk} \bar{\bar{v}}_k$, $\bar{v}_j = g(\bar{r}_j)$, $r_i = \sum_{j=1}^{J_n} W_{ij} \bar{v}_j$, and $v_i = g(r_i)$, where $g(x) = c \tanh(x)$, and c is a scaling factor. For convenience a generalized weight vector Θ is defined as: $\Theta = [W_1, .., W_i, .., W_{I_n}, R_1, .., R_j, .., R_{J_n}, S_1, .., S_k, .., S_{K_n}]$, where $(\cdot)_a$ represents the a^{th} row of the matrix $(\cdot)$. The mapping realized by the network can then be compactly expressed as: $v = N(Z, \Theta)$, where Z is the input vector, i.e. $Z = (z_1, z_2, ...z_l, ...z_{L_n})$, and N is used as a convenient notation to represent the mapping achieved by the network.

It has been proven that a multilayer feedforward neural network with one hidden layer (containing a sufficient number of units) is capable of approximating any continuous function to any degree of accuracy (Cybenko, 1989). It is in this sense that multilayer feedforward neural networks have been established as a class of "universal approximators". Thus, for the uncertainty function $\eta(Z)$, where $Z = (q, \dot{q}, \ddot{q})$, there exists a set of weights Θ^* for a neural network (with a sufficient number of hidden units) with the output $v^d = N(Z, \Theta^*)$ such that, for some $\epsilon \geq 0$, $\|\eta - v^d\| \leq \epsilon$, where $\|(\cdot)\|$ denotes the supremum of $(\cdot)$.

Note that the above statement only assures that the weights Θ^* exist, it does not indicate what their values are, or how to find them. To determine these weights is the objective of neural network learning. Let the cost function to be minimized be: $J_{\Delta v} = \frac{1}{2} \Delta v^T \Delta v$.

Applying the error-backpropagation algorithm (McClelland & Rumelhart, 1986) yields: $\dot{\Theta} = -\lambda_n \frac{\partial J_{\Delta v}}{\partial \Theta} = -\lambda_n \Delta v^T \frac{\partial \Delta v}{\partial \Theta}$, where λ_n is the learning rate. Since $\Delta v = v^d - v$, where v^d is the unknown desired output of the network, and $\frac{\partial v^d}{\partial \Theta} = 0$, the weight update rule becomes: $\dot{\Theta} = \lambda_n \Delta v^T \frac{\partial v}{\partial \Theta}$. Specifically, the dynamics of the weights W_{ij}, R_{jk}, and S_{kl} can be expressed (Muller & Reinhardt, 1990) as: $\dot{W}_{ij} = \lambda_n \Gamma_i \bar{v}_j$, $\dot{R}_{jk} = \lambda_n \bar{\Gamma}_j \bar{\bar{v}}_k$, and $\dot{S}_{kl} = \lambda_n \bar{\bar{\Gamma}}_k z_l$, where $\Gamma_i = \Delta v_i g'(v_i)$, $\bar{\Gamma}_j = g'(\bar{v}_j) \sum_{i=1}^{I_n} \Gamma_i W_{ij}$, $\bar{\bar{\Gamma}}_k = g'(\bar{\bar{v}}_k) \sum_{j=1}^{J_n} \bar{\Gamma}_j R_{jk}$, and $g'(\cdot) = \frac{\partial g(\cdot)}{\partial (\cdot)}$. The learning process is illustrated in Figure 5.

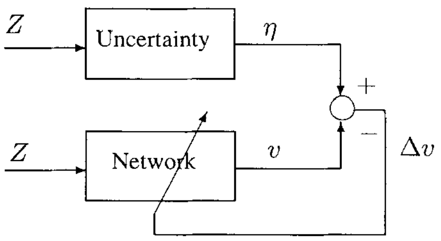

Figure 5. Neural Network for Uncertainty Compensation.

The control error Δv can be determined in real-time, according to (8), as follows

$$\Delta v = \ddot{q}^d - \ddot{q} + K_v(\dot{q}^d - \dot{q}) + K_p(q^d - q). \tag{9}$$

The closed-loop dynamics of the robotic system with the neural network learning on-line is described by

$$\begin{cases} \dot{e} = Ae + B\Delta v(q, \dot{q}, \ddot{q}, \Theta) \\ \dot{\Theta} = -\lambda_n \Delta v^T(q, \dot{q}, \ddot{q}, \Theta) \frac{\partial \Delta v(q, \dot{q}, \ddot{q}, \Theta)}{\partial \Theta}. \end{cases} \tag{10}$$

Stability and performance of the system (10) are examined next.

5. Analysis

We first prove that the closed-loop system with the neural network learning on-line is stable in the sense that all signals in the system are bounded. We then show that the performance of the system is improved in the sense that certain measure of the control error Δv decreases as the learning process is iterated (i.e., as the number of learning trials increases). The conjecture is that reduction in the control error eventually leads to reduction in the trajectory tracking error. This conjecture is verified by the results of the computer simulation presented in Section 6.

5.1. Stability

THEOREM 1 *Given a continuous and twice-differentiable reference position trajectory $q^d(t)$, the system (10) is bounded-input bounded-output (BIBO) stable for sufficiently large gains K_v and K_p.*

Note that a system is said to be BIBO stable if for a bounded input, the output of the system is also bounded. This term is defined rigorously in Appendix A. Proof of the above theorem is presented in Appendix B. It suffices to state here that this theorem asserts that for sufficiently large gains, the control input u and the trajectory tracking error e in (10) are bounded.

COROLLARY 1 *The acceleration signal $\ddot{q}(t)$ is bounded.*

COROLLARY 2 *The control error Δv is bounded.*

COROLLARY 3 *The weights of the neural network remain bounded during a given trial.*

Proofs for Corollaries 1, 2, and 3 can be found in Appendix C, D, and E respectively. We next examine how the network learning process affects the performance of the manipulator.

5.2. Performance Improvement

Preliminaries

Let t represent the continuous time variable, i.e., $0 \leq t < \infty$. Let learning start at time $t = 0$, and let each trial last T seconds. Then the p^{th} trial spans the time period from $t = (p-1)T$ to $t = pT$. Note that p is thus implicitly defined as a positive integer.

Let ξ be the time variable associated with one trial, i.e., $0 \leq \xi \leq T$. (The notation $\dot{x}$ from here on means either $\frac{dx}{dt}$ or $\frac{dx}{d\xi}$ as it should be clear from the context.) Let $x(p,\xi)$ denote the value of the variable x at the ξ^{th} second of the p^{th} trial. Then $x(p,0)$ represents the value of the variable x at the beginning of the p^{th} trial, and $x(p,T)$ represents the value of the variable x at the end of the p^{th} trial. Note that $x(p,0) = x(p-1,T)$. Let Θ_m be an element of the weight vector Θ. Let $\Delta\theta_m(p,\xi)$ denote the change of Θ_m during the first ξ seconds of the p^{th} trial, i.e., $\Delta\theta_m(p,\xi) = \int_0^\xi \dot{\Theta}_m(p,\sigma)d\sigma$.

The L_∞^n-norm of a Lebesgue integrable function $f(t) : R_+ \rightarrow R^n$, denoted by $\|f\|_\infty$, is defined as: $\|f\|_\infty = ess\,sup_{t\in[0,\infty)}\|f(t)\| < \infty$. The *extended* L_∞^n-space (with the truncated L_∞^n-norm $\|(\cdot)_T\|_\infty$), denoted by $L_{\infty e}^n$, is defined as: $L_{\infty e}^n = \{f : R_+ \rightarrow R^n \,|\, f_T \in L_\infty^n, \forall T < \infty\}$, where $f_T(t) = \begin{cases} f(t) & \text{for } t \in [0,T] \\ 0 & \text{otherwise} \end{cases}$. For convenience, the notation $\|f\|_{T\infty}$ is used to denote $\|f_T\|_\infty$.

Let $C[0,T]$ denote the family of Lebesgue integrable function $f_i(\xi)$ for all $\xi \in [0,T]$. The L_2-norm of a vector function $f(\xi) = (f_1, f_2, ..., f_i, ..., f_n), f_i \in C[0,T]$, over the time interval $[0,T]$ is defined (Vidyasagar, 1992) as: $\|f\|_{2T} = \left(\int_0^T f^T(\xi)f(\xi)d\xi\right)^{\frac{1}{2}}$. For convenience, the notation $\|\cdot\|$ instead of $\|\cdot\|_{2T}$ is used to denote this norm in the subsequent analysis.

PROPOSITION 1 *For the closed-loop system (10), there exists some small $\lambda_n > 0$ such that $\|\Delta\theta_m(p+1,\xi) - \Delta\theta_m(p,\xi)\|_{T\infty} << 1$, for all $\xi \in [0,T]$.*

The proof of this proposition is presented in Appendix F.

Remark: Since $\Delta\theta_m(p+1,\xi)$ represents the amount of weight change during the first ξ seconds of the $(p+1)^{th}$ trial, and $\Delta\theta_m(p,\xi)$ represents the amount of weight change during the first ξ seconds of the p^{th} trial, this proposition (illustrated in Figure 6) means that, for $\lambda_n << 1$, the difference between the change in Θ_m of any two successive trials can be considered negligible, i.e., $\Delta\theta_m(p+1,\xi) \approx \Delta\theta_m(p,\xi), \forall\xi \in [0,T]$.

A qualitative interpretation for this proposition can be constructed based on observation on the time-scale difference between the dynamics of the manipulator and the dynamics of the on-line learning process of the neural network. With a small learning rate λ_n, the overall robotic system can be considered as a two-time-scale system with the manipulator exhibiting a "fast" dynamics while the network exhibiting a "slow" dynamics. As the learning rate λ_n approaches zero, the change in the weights *per trial* can be expected to be infinitesimally small. Such small change in the weights will not have significant effect on the state of the robot. Therefore, between any two *successive* trials, the change in the state of the robot, and consequently the difference between the two amounts of weight change, can be considered negligible. This proposition is verified by computer simulation presented in Section 6.

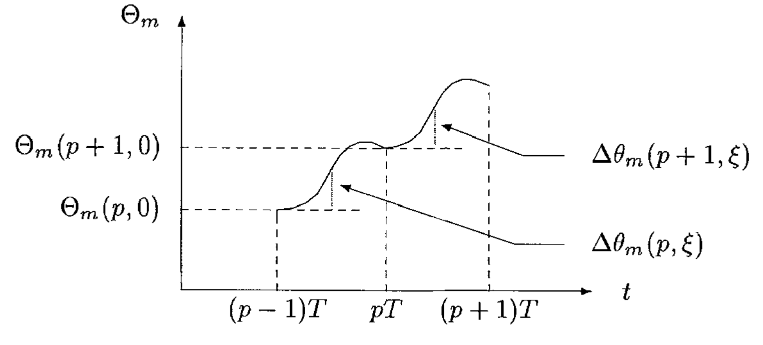

Figure 6. Weight and Weight Change.

Main Results

Recall from Section 4 that the objective of neural network learning is to reduce the control error Δv. The following theorem establishes rigorously that such an objective can be achieved.

THEOREM 2 *The L_2-norm of the control error Δv decreases as the number of trials p increases, i.e., $\|\Delta v(p+1)\| < \|\Delta v(p)\|$.*

Proof: Since (from Section 4) $J_{\Delta v}(p,\xi) = \frac{1}{2}\Delta v^T(p,\xi)\Delta v(p,\xi)$, from the definition of the L_2-norm, we have $\|\Delta v(p)\|^2 = 2\int_0^T J_{\Delta v}(p,\xi)d\xi$. To prove that $\|\Delta v(p+1)\| < \|\Delta v(p)\|$, it suffices to show that $\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 < 0$. Now $\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 = 2\int_0^T (J_{\Delta v}(p+1,\xi) - J_{\Delta v}(p,\xi))\,d\xi$. Based on the fact that the change in the control error Δv between any two successive trials p and $(p+1)$ is a direct consequence of the change in the network weights, we expand $J_{\Delta v}(p+1,\xi)$ about $J_{\Delta v}(p,\xi)$, while ignoring the higher order terms, to obtain

$$J_{\Delta v}(p+1,\xi) - J_{\Delta v}(p,\xi) \approx \sum_{m=1}^{c_\theta}\left[\frac{\partial J_{\Delta v}}{\partial \Theta_m}(\Theta_m(p+1,\xi) - \Theta_m(p,\xi))\right], \qquad (11)$$

where c_θ is the total number of weights, i.e., $c_\theta = I_n \times J_n + J_n \times K_n + K_n \times L_n$. Because $\frac{\partial J_{\Delta v}(p,\xi)}{\partial \Theta_m(p,\xi)} = \frac{\partial J_{\Delta v}(p,\xi)}{\partial \Delta v(p,\xi)}\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)} = \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}$, so (11) becomes

$$\begin{aligned} J_{\Delta v}(p+1,\xi) &- J_{\Delta v}(p,\xi) \\ &= \sum_{m=1}^{c_\theta}\left[\Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}(\Theta_m(p+1,\xi) - \Theta_m(p,\xi))\right]. \qquad (12)\end{aligned}$$

Note that $\Theta_m(p+1,\xi)$ and $\Theta_m(p,\xi)$ can be expressed as $\Theta_m(p+1,\xi) = \Theta_m(p+1,0) + \Delta\theta_m(p+1,\xi)$, and $\Theta_m(p,\xi) = \Theta_m(p,0) + \Delta\theta_m(p,\xi)$, where $\Theta_m(p+1,0) = \Theta_m(p,T)$ represents the weight value at the end of the p^{th} trial, while $\Theta_m(p,0)$ represents the weight value at the beginning of the p^{th} trial. From Proposition 1, we have $\|\Delta\theta_m(p+1,\xi) - \Delta\theta_m(p,\xi)\|_{T\infty} \ll 1$, hence $\Theta_m(p+1,\xi) - \Theta_m(p,\xi) \approx \Theta_m(p+1,0) - \Theta_m(p,0)$, and consequently

$$\begin{aligned} &\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 \\ &= 2\int_0^T (J_{\Delta v}(p+1,\xi) - J_{\Delta v}(p,\xi))\,d\xi \\ &= 2\int_0^T \sum_{m=1}^{c_\theta}\left[\Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}(\Theta_m(p+1,0) - \Theta_m(p,0))\right]d\xi \\ &= 2\sum_{m=1}^{c_\theta}\left[\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}(\Theta_m(p+1,0) - \Theta_m(p,0))d\xi\right]. \qquad (13)\end{aligned}$$

Since $\Theta_m(p+1,0)$ and $\Theta_m(p,0)$ are no longer functions of ξ, so (13) becomes

$$\begin{aligned} &\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 \\ &= 2\sum_{m=1}^{c_\theta}\left[\left(\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi\right)(\Theta_m(p+1,0) - \Theta_m(p,0))\right].\end{aligned}$$

Note that $\Theta_m(p+1,0) - \Theta_m(p,0) = \int_0^T \dot{\Theta}_m(p,\xi)d\xi$. But from the learning rule presented in Section 4, we have $\dot{\Theta}_m(p,\xi) = -\lambda_n\Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}$. Thus $\Theta_m(p+1,0) -$

$\Theta_m(p,0) = \int_0^T \dot{\Theta}_m(p,\xi)d\xi = -\lambda_n \int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi$. Consequently,

$$\begin{aligned}
&\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 \\
&= 2\sum_{m=1}^{c_\theta}\left[\left(\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi\right)\left(-\lambda_n\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi\right)\right] \\
&= -2\lambda_n\sum_{m=1}^{c_\theta}\left[\left(\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi\right)^2\right] \\
&\leq 0.
\end{aligned}$$

Now $\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 = 0$ if $\sum_{m=1}^{c_\theta}\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi = 0$. But this implies that $\int_0^T \dot{\Theta}_m d\xi = 0$, which states that the total change in the weight Θ_m for the trial p is zero. This means that the gradient search conducted by the error-backpropagation algorithm has reached either a global minimum or a local minimum. If this is not the case, then we have $\|\Delta v(p+1)\|^2 - \|\Delta v(p)\|^2 < 0$, that is, $\|\Delta v(p+1)\| < \|\Delta v(p)\|$, which implies that the controller error Δv decreases as the number of trials p increases.

■

Remark: An important practical issue concerning neural network learning is whether the weights will remain bounded. This issue can be resolved (in the context of the above analysis) by observing the facts that (i) the weights are bounded during a given trial (i.e., Corollary 3), and (ii) from an implementation standpoint, the learning process can be terminated once the L_2-norm of the control error no longer decreases from trial to trial. Thus, if we start the learning process with a set of finite weights, and if the weight dynamics condition (i.e., Proposition 1) is satisfied, then it is assured that the weight values are finite at the point when the learning process is terminated.

6. Simulation

The purpose of conducting computer simulation is to verify the analytical results presented in Section 5. Specifically, we conducted the simulation to confirm, through a numerical example, that (i) for a small learning rate λ_n, the proposition that the weight change between two successive trials can be considered negligible is valid, (ii) upon confirmation of (i), the L_2-norm of the control error Δv decreases as the number of learning trial p increases, and (iii) reduction in the control error Δv results in reduction in trajectory tracking error e.

The manipulator considered in the simulation, as shown in Figure 7, consists of two links with point masses; that is the mass of each link is assumed to be concentrated at the tip of each link.

The dynamics of this manipulator can be formulated (Craig, 1986) as

$$\tau_1 = m_2 l_2^2(\ddot{q}_1 + \ddot{q}_2) + m_2 l_1 l_2 (2\ddot{q}_1 + \ddot{q}_2)\cos q_2 + (m_1 + m_2) l_1^2 \ddot{q}_1$$

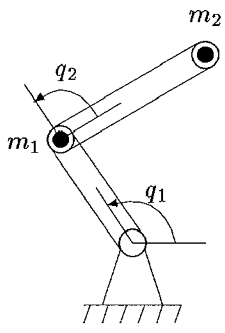

Figure 7. A Two-Link Manipulator.

$$
\begin{aligned}
& -m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 - 2 m_2 l_1 l_2 \dot{q}_1 \dot{q}_2 \sin q_2 + m_2 l_2 g \cos(q_1 + q_2) \\
& +(m_1 + m_2) l_1 g \cos q_1, \\
\tau_2 \; = \; & m_2 l_1 l_2 \ddot{q}_1 \cos q_2 + m_2 l_1 l_2 \dot{q}_1^2 \sin q_2 + m_2 l_2 g \cos(q_1 + q_2) + m_2 l_2^2 (\ddot{q}_1 + \ddot{q}_2),
\end{aligned}
$$

where q_1 and q_2 are the angles of the two joints (as indicated in Figure 7) with the corresponding torques τ_1 and τ_2, m_1 and m_2 are the mass of the links, l_1 and l_2 are the length of the links, and $g = 9.8 \;\; m/s^2$. To simulate the dynamics of the manipulator, the following parameter values were used: $m_1 = 5.0 \; kg$, $m_2 = 4.0 \; kg$, $l_1 = 0.7 \; m$, and $l_2 = 0.5 \; m$. In implementing the inverse dynamics control, the values for the masses were altered to introduce uncertainty into the system by setting: $\hat{m}_1 = 4.5 \; kg$ and $\hat{m}_2 = 5.0 \; kg$. The desired position trajectory for both joints, generated by a 5^{th}-order polynomial, is as shown in Figure 8.

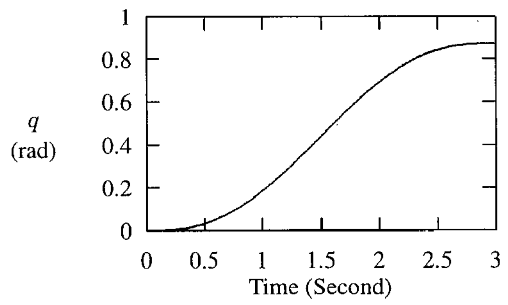

Figure 8. Desired Position Trajectory.

A network consisting of six input units, two hidden layers with twenty units in each layer, and two output units was used in the simulation. The learning λ_n was set at a very small value of 5×10^{-5}, so as to be consistent with the argument for Proposition 1 discussed in Section 5. The initial weights of the neural network for the first trial were set to arbitrary values in the order of 10^{-4}. The control gains were set to be: K_p = diag[30, 30] and K_v = diag[10, 10]. A series of simulation runs (i.e., trials) were carried out. The control error Δv was calculated according to (9), which requires the acceleration signal $\ddot{q}$. In the simulation, $\ddot{q}$ was estimated on-line based on $\dot{q}$.

To examine the dynamical behavior of the weights between two successive trials, Figures 9 and 10 show respectively the dynamics and the difference between the change of a connection weight $R_{(10,10)}$ (the weight between the 10th unit of the first hidden layer and the 10th unit of the second hidden layer) during the 10th trial and the 11th trial.

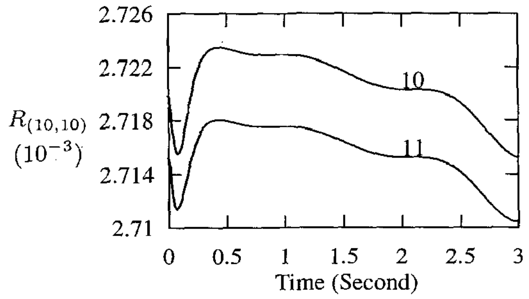

Figure 9. Dynamics of Weight $R_{(10,10)}$: 10th and 11th Trial.

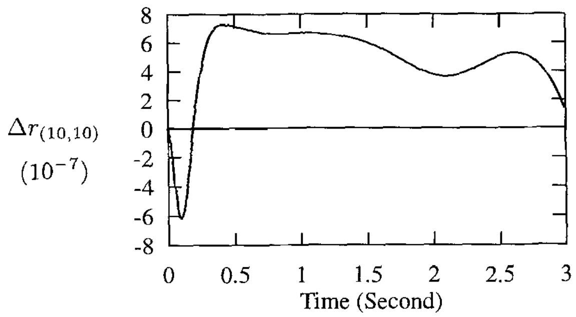

Figure 10. Difference between Change of Weight $R_{(10,10)}$: 10th and 11th Trial.

It is shown here, as an example, how the difference between the change in $R_{(10,10)}$ of the two trials, 10th and 11th, is calculated. Using the notations defined in Section 5, the difference of weight change between trial 10 and trial 11 can be written as: $\Delta r_{(10,10)}(\xi) = \Delta R_{(10,10)}(11,\xi) - \Delta R_{(10,10)}(10,\xi)$, where $\Delta R_{(10,10)}(11,\xi) = R_{(10,10)}(11,\xi) - R_{(10,10)}(11,0)$, $\Delta R_{(10,10)}(10,\xi) = R_{(10,10)}(10,\xi) - R_{(10,10)}(10,0)$, and ξ is the time variable, i.e., $\xi \in [0,3]$.

Figures 11 and 12 show respectively the dynamics and the difference between the change of the same weight during the 20th trial and the 21st trial. From Figures 9-12, it can be seen that the difference of the weight change between two successive trials (i.e., 10th and 11th, 20th and 21st) is indeed negligible.

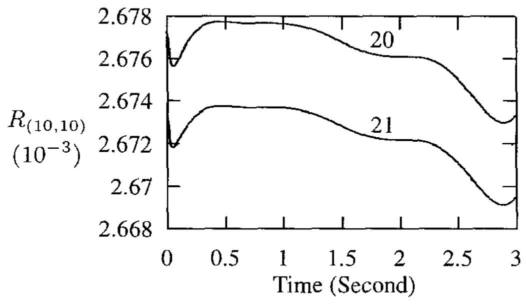

Figure 11. Dynamics of Weight $R_{(10,10)}$: 20th and 21st Trial.

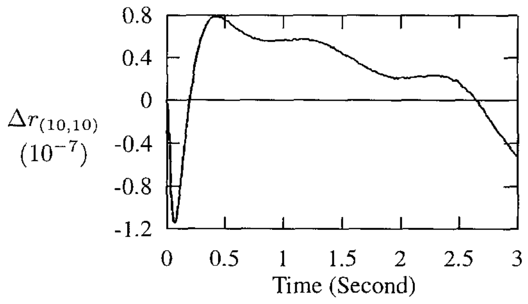

Figure 12. Difference between Change of Weight $R_{(10,10)}$: 20th and 21st Trial.

Figures 13-16 show the dynamics and the difference between the change of another weight $W_{(1,10)}$ (the weight between the 10th unit of the second hidden layer and the 1st unit of the output layer) during the same pair of trials (i.e., 10th and 11th, 20th and 21st).

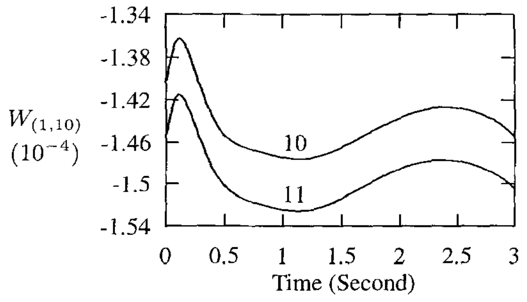

Figure 13. Dynamics of Weight $W_{(1,10)}$: 10th and 11th Trial.

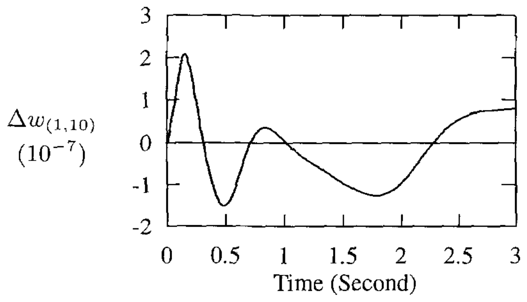

Figure 14. Difference between Change of Weight $W_{(1,10)}$: 10th and 11th Trial.

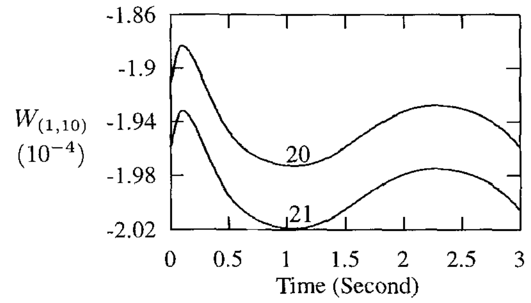

Figure 15. Dynamics of Weight $W_{(1,10)}$: 20^{th} and 21^{st} Trial.

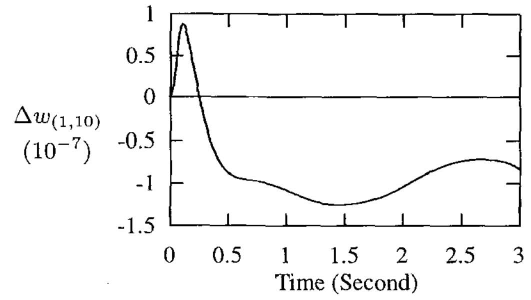

Figure 16. Difference between Change of Weight $W_{(1,10)}$: 20^{th} and 21^{st} Trial.

Comparing Figures 9, 11, 13, and 15 with Figure 6 (which illustrates the proposition that the difference between the weight change of any two successive trials is negligible), it is evident that, when the learning rate is small, the neural network does indeed possess the dynamical behavior as predicted.

Recall that the analytical conclusion in Section 5 based on Proposition 1 is that the L_2-norm of the control error Δv decreases as the number of trials p increases. Figure 17 shows the L_2-norm of the control error Δv versus the trial number p for this simulation.

It can be seen that the control error Δv indeed decreases as the number of trials p increases. This confirms the theoretical conclusion presented in Section 5. It is conjectured that reduction in the control error Δv eventually results in reduction in the trajectory tracking error. Figures 18 and 19 show the joint errors without compensation (i.e., $v = 0$) and during the 300^{th} trial. It can be seen that the joint errors are reduced as the learning of the network progresses.

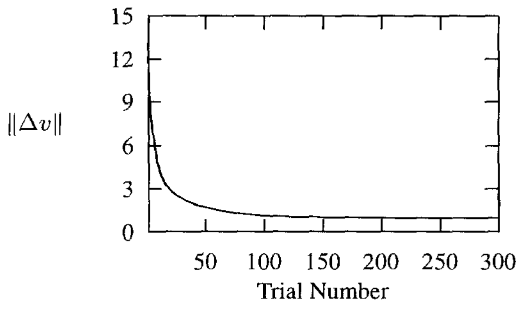

Figure 17. Control Error.

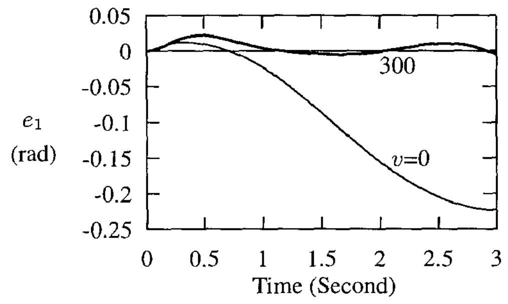

Figure 18. Error in q_1.

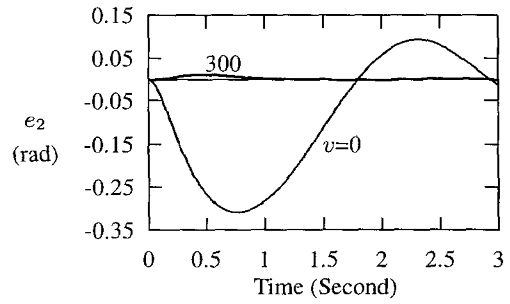

Figure 19. Error in q_2.

From the results of the simulation, it can be seen that both the proposition (that with a small learning rate, the weight change between two successive trials can be considered negligible) and the analytical conclusion (that the L_2-norm of the control error decreases as the number of trials p increases) are valid. It can also be seen that reduction in the control error Δv indeed results in reduction in the trajectory tracking error. A laboratory experiment conducted to demonstrate the effectiveness of the proposed approach is described next.

7. Experiment

The robot used for implementing the proposed approach is of a five-bar parallel link configuration operating in the horizontal plane (Lokhorst, 1990) as shown in Figure 20.

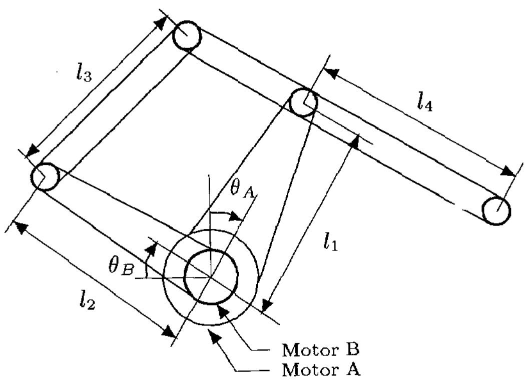

Figure 20. A Laboratory Manipulator.

High torque, brushless direct current motors are used to drive the manipulator without gear reduction. The motors used are manufactured by Yokogawa Corporation. S[illegible]ations of the motors are listed in Table 1.

Table 1. Direct-drive Motor Specification.

Specification	*Motor A*	*Motor B*
Manufacturer	Yokogawa	Yokogawa
Model	DMA 12000	DMB 1045
Maximum Torque (Nm)	200	45
Maximum Speed (rev/s)	1.2	2.4
Encoder Resolution $(pulse/rev)$	1024000	655360
Diameter (mm)	264	160
Length (mm)	188	143
Weight (kg)	29	9.5

The dynamics of this manipulator can be formulated as (Lokhorst, 1990)

$$\begin{bmatrix} d_1 & d_3 S_{BA} \\ d_3 S_{BA} & d_2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_A \\ \ddot{\theta}_B \end{bmatrix} + \begin{bmatrix} -d_3 C_{BA} \dot{\theta}_B^2 \\ d_3 C_{BA} \dot{\theta}_A^2 \end{bmatrix} + \begin{bmatrix} b_1 \dot{\theta}_A + b_3 sgn(\dot{\theta}_A) \\ b_2 \dot{\theta}_B + b_4 sgn(\dot{\theta}_B) \end{bmatrix} = \begin{bmatrix} \tau_A \\ \tau_B \end{bmatrix};$$

where $S_{BA} = \sin(\theta_B - \theta_A)$, $C_{BA} = \cos(\theta_B - \theta_A)$, θ_A and θ_B are the angles of the two motors with the corresponding torques τ_A and τ_B, $d_1 = m_1 l_{c1}^2 + m_3 l_{c3}^2 + m_4 l_1^2 + I_1 + I_3$, $d_2 = m_2 l_{c2}^2 + m_3 l_2^2 + m_4 l_{c4}^2 + I_2 + I_4$, and $d_3 = m_3 l_2 l_{c3} - m_4 l_1 l_{c4}$. Here m_i is the mass of the i-th link, l_i is the length of the i-th link, l_{ci} is the distance from the joint to the center of mass of the i-th link, and I_i is the moment of inertia of the i-th link. The lengths of the links are: $l_1 = l_3 = l_4 = 0.5\ m$, and $l_2 = 0.3\ m$. The parameters b_1 and b_2 are the coefficients of viscous friction and the parameters b_3 and b_4 are the coefficients of static friction. The values of the parameters have been experimentally determined (Lawryshyn, 1993) and are shown in Table 2 under the heading "True".

Table 2. True and Estimated Parameter Values for Robot Trajectory Tracking Experiment.

Parameter	*True*	*Estimated*
d_1	$3.4\, kgm^2$	$1.2\, kgm^2$
d_2	$-0.1\, kgm^2$	$-0.511\, kgm^2$
d_3	$1.2\, kgm^2$	$0.772\, kgm^2$
b_1	$8.3\, kgm^2/s$	$8.3\, kgm^2/s$
b_2	$1.6\, kgm^2/s$	$1.6\, kgm^2/s$
b_3	$2.3\, Nm$	$2.3\, Nm$
b_4	$0.46\, Nm$	$0.46\, Nm$

A neural network with six input units, two output units, and ten units in each of its two hidden layers was implemented in the form of a program written in the C programming language for the experiment. Due to the limitation of hardware and computational resources — the control laws were executed on a 386 microcomputer with a clock speed of $25MHz$, the learning process of the neural network was implemented off-line. This means that during each trial, only the static mapping of the network was activated, while the weights of the network were fixed. Data required for network learning were recorded during the trial. Once a trial was completed, the weights of the network were updated using the recorded data. Because network learning was conducted off-line, Theorem 2 does not apply to this experiment. The significance of the experiment lies in demonstrating the effectiveness of the proposed approach in the execution of realistic robotic tasks.

The weights of the neural network were randomly initialized to be in the order of 10^{-3}. The learning rate of the neural network, initially set at 1×10^{-3}, was gradually reduced to 5×10^{-5} as the learning process was iterated. The input signals to the neural network were $q, \dot{q}$, and $\ddot{q}$. The joint acceleration $\ddot{q}$ was obtained by filtering the the joint velocity signal $\dot{q}$ on-line using a Kalman filter (Yanovski, 1991). The feedback gains were set to be: $K_v = \text{diag}[15, 15]$ and $K_p = \text{diag}[50, 50]$. The control law was executed at a sampling frequency of $86Hz$.

A task trajectory for the tip of the manipulator as shown in Figure 21 was planned in this experiment. This trajectory was generated off-line and stored in memory, and retrieved during real-time task execution.

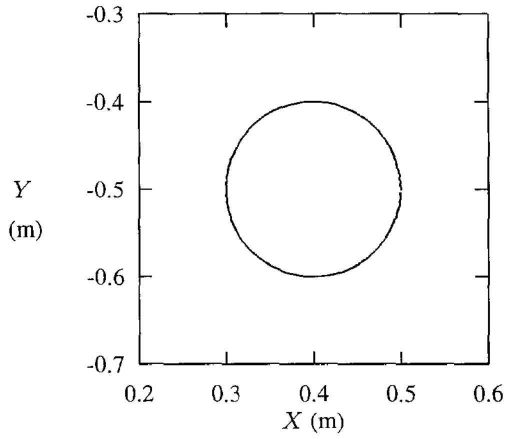

Figure 21. Desired Task Trajectory.

To establish a basis for comparison, a baseline experiment was first conducted using the "true" parameters of the manipulator as shown in Table 2. Figure 22 shows the results of this experiment. This baseline experiment shows the performance of the system when there is presumably no uncertainty in the system parameters.

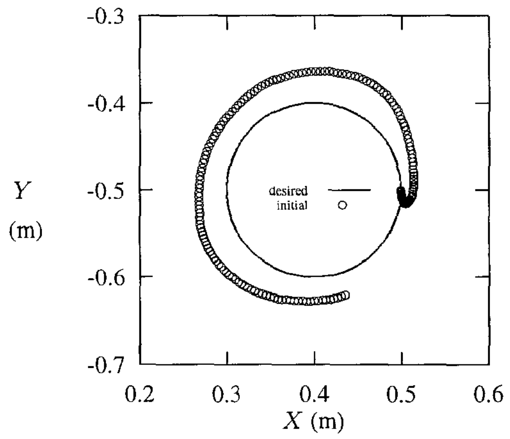

Figure 22. Desired Trajectory and Baseline Trajectory.

It is noted that even when the "true" parameter values were used in generating the control signal, the trajectory of the tip of the manipulator still deviates from the desired trajectory quite significantly. Such deviation can be accounted for by considering (i) the effect of time-delay in executing the control law with a sampling frequency of only 86 Hz, and (ii) the fact that the control gains K_v and K_p were set at low values. The reason for using such low gains in this experiment was to allow a larger initial tracking error (when uncertainty but no compensation was introduced) so that the effect of neural network learning in improving the performance of the manipulator could be more clearly seen. Since the key objective in this experiment was to show performance improvement, the emphasis was on showing reduction in the "size" of the tracking error when a neural network was used as a compensator, as compared to the case where no compensator was used.

Upon obtaining the baseline trajectory, uncertainty in the system parameters was introduced. The values of the three parameters, namely, d_1, d_2, and d_3, of the manipulator were altered, as summarized in Table 2 under the heading "Estimated". Another trial was executed with the uncertainty introduced in the manipulator parameters and without any compensating signal incorporated into the control law. Figure 23 shows the results of this trial (which is referred to as the initial trajectory) together with the baseline trajectory.

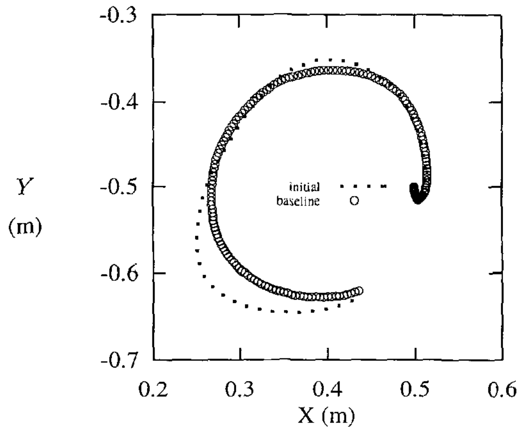

Figure 23. Baseline Trajectory and Initial Trajectory.

A series of trials was then executed with the network weights being adjusted off-line after each trial. Figures 24 shows the trajectory of the manipulator tip at the 20th trial and at the 30th trial. It can be seen that the incorporation of the neural network as an uncertainty compensator indeed improved system performance in the sense that the actual trajectory of the manipulator approached the baseline trajectory as the learning process of the neural network was iterated.

8. Discussion

8.1. Implication of Results

We have proposed an approach to improving the performance of uncertain robotic systems using neural networks. It has been shown that this approach is applicable to repetitive continuous-path robot operation. In this approach, uncertainty in the robotic system is

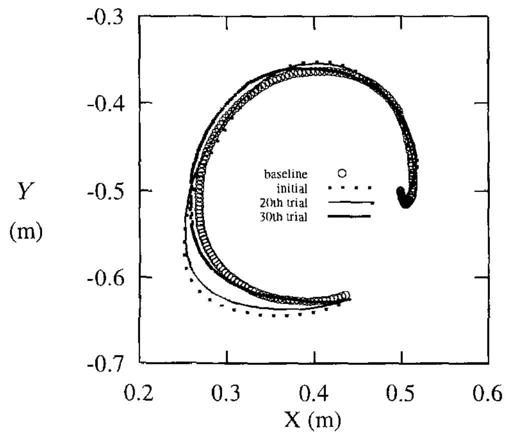

Figure 24. Task Trajectory: 20th and 30th Trial.

quantified and a neural network is used to "nullify" the uncertainty so that performance improvement can be achieved.

Using techniques from nonlinear system theory, closed-loop stability of the robotic system (incorporated with a neural network) has been analyzed. Results of the analysis confirm that the closed-loop system is stable in the sense that all signals in the system are bounded. Stability of closed-loop systems embedded with neural networks is a key issue that has not been adequately addressed in the literature. The stability analysis presented in this article offers a possible solution in resolving this issue.

A method for analyzing the performance of the robotic system (incorporated with a neural network) has been developed. Using this method, the effect of the dynamics of the neural network on the performance of the manipulator has been revealed. It has further been shown that the performance of the robotic system is improved as the learning process of the neural network is iterated.

The results of these analyses provide theoretical justification for the use of multilayer feedforward neural networks with the error-backpropagation algorithm in a feedback control system in which the dynamics of the network is coupled with that of the controlled plant. The error-backpropagation algorithm has been one of the most commonly used learning rules for neural networks. Various schemes have been proposed in the literature on the application of neural networks to robot control using this algorithm, but the important question of how the dynamics of the neural network affects the performance of the robotic system has not been addressed rigorously in the literature. The performance analysis presented in this article provides a plausible answer. The analysis confirms that, with a sufficiently small learning rate, the error-backpropagation algorithm will not destabilize the closed-loop system and will improve the performance of the robotic system as the learning process is iterated. Numerical simulations have been conducted. Results of the simulation have confirmed the conclusions of the theoretical analysis.

An experiment using a laboratory manipulator has been conducted. The results of the experiment clearly demonstrate the effectiveness of the proposed approach in improving the performance of the robotic system. The experimental implementation of the proposed approach, together with the positive experimental results, are important not only in demonstrating the effectiveness of the neural network as an uncertainty compensator, but also in demonstrating the effectiveness of a control system incorporated with a neural network for realistic robotic tasks. Many studies on the application of neural networks to system control in general and to robot control in particular rely on numerical simulation to verify the conclusions therein; very few proposed schemes have been physically implemented to verify their effectiveness. Extensive and conclusive experimental results are needed to affirm neural networks as viable engineering tools. The experiment reported in this article represents a small step in gathering such results.

8.2. Research Directions

In the context of the work presented in this article, the following directions are suggested.

Comparison with Other Approaches
An important and necessary extension of the work presented is an in-depth analysis in comparing the proposed approach with various other approaches (such as adaptive control) so that the full potential of neural networks for robot control (as discussed in Section 1) can be firmly substantiated and convincingly demonstrated.

Effect of Manipulator Link and Joint Flexibility
Throughout this work, the manipulators under consideration were treated as rigid mechanical linkages. In reality, however, link and joint flexibility exist in industrial robots. It would thus be practically meaningful to extend the proposed approach to account for the effect of link and joint flexibility of industrial robots. Such an extension will yield results that further affirm the utility of neural networks in practical robotic applications.

Reduction of the Number of Learning Trials
In analyzing the effect of neural network learning on the performance of the robotic system, i.e., Theorems 2, the condition needed to guarantee the reduction of the control error was that the difference between the weight change of two successive trials is negligible. The justification of this condition was based on the requirement that the learning rate be sufficiently small. A small learning rate, however, implies that to achieve significant performance improvement, a large number of learning trials may be required.

Since the dynamics of the network weights also depends on the gradient of the "error surface", a technique that allows variation of the learning rate based on the "steepness" of the gradient would be a potential solution to reduce the number of learning trials.

Determination of the Necessary Number of Learning Trials
The clarification of the relationship between the reduction in the control error and the reduction in the trajectory tracking error represents an important theoretical issue in under-

standing the utility of neural networks for system control in general and for robot control in particular. It is important because successful theoretical clarification of this relationship would provide a definitive answer to the question of how many trials are required in order to reduce the tracking error to within certain tolerance.

Learning Multiple Trajectories
In manufacturing operations, a robot is often required to be capable of performing more than one task. This implies that the robot controller must be able to execute more than one trajectory. The scope of the present work has been limited to the case where the robot controller is to learn to improve its performance for a single trajectory. Such a controller is clearly inadequate in a production environment. Thus, a practical extension of the present work is to develop methodologies which enable the neural-network-based controller to learn multiple trajectories and to execute these trajectories without interference.

Effective Global Convergence
Currently there is no known method that guarantees global convergence of neural network learning. This means that it is possible that the set of weights found by the error-backpropagation algorithm is optimal only locally. It is of course worthwhile to find a learning algorithm that theoretically guarantees global convergence, but it is also practically meaningful to find a way to circumvent local optimality so as to achieve effective global convergence. One plausible approach is to use multiple networks in such a way that when one network becomes "trapped" in a local optimum, another network is activated to continue the learning process. Conceptually, it can be argued that it is possible to reduce the control error to any required tolerance by using a series of networks. A method of implementing such a series of networks would significantly enhance the practicality of neural networks in robot control.

Hardware Implementation of Neural Networks for Robot Control
Currently in studies reported in the literature on neural networks for robot control, the neural networks are usually implemented in the form of software programs and executed on general-purpose digital computers. Such software implementation, however, does not take full advantage of the parallel processing capability of neural networks. Thus, in order to fully utilize the capability of neural networks for practical robotic applications, it is necessary to investigate hardware implementation of neural networks. One plausible hardware implementation of a neural network is in the form of a VLSI chip. It can be anticipated that successful synthesis of "neuro-chips" and commercial robot controllers would advance the application of industrial robots to a more sophisticated level.

Appendix A

Definitions

The following definitions are based on material presented in (Vidyasagar, 1992). The convolution of a Laplace transformable signal $f(t)$ and a transfer function matrix $M(s)$ is denoted by Mf, i.e., $Mf = (m * f)(t)$, where $*$ denotes the convolution operator. The l_2-norm of a vector $x \in R^n$, denoted by $\|x\|$, is defined as: $\|x\| = (\sum_{j=1}^{n} |x_j|^2)^{\frac{1}{2}}$. The l_2-norm of a matrix $A \in R^{n\times n}$, denoted by $\|A\|$, is defined as: $\|A\| = [\max_i \lambda_i(A^T A)]^{\frac{1}{2}}$, where λ_i denotes the eigenvalue of the matrix A. The L_∞^n-norm and the *extended* L_∞^n-space (for the truncated L_∞^n-norm) are as defined in Section 5. A system with input x and output y is said to be bounded-input bounded-output stable (or BIBO stable) if for every $x \in L_\infty$, $y \in L_\infty$. The L_∞^n-norm of a transfer matrix P is defined as: $\|P\|_\infty = sup_{x\in L_\infty^n - 0} \frac{\|Px\|_\infty}{\|x\|_\infty}$. Let β denote the norm $\|P\|_\infty$, then $\|Px\|_{T\infty} \leq \beta \|x\|_{T\infty}$.

Appendix B

Proof of Theorem 1

The method presented in (Spong & Vidyasagar, 1987) is utilized in constructing this proof. Through algebraic operations, the system (10) can alternatively be expressed as

$$\dot{\bar{e}} = \bar{A}\bar{e} + B(\bar{\eta} + u) \tag{B.1}$$

where $\bar{e} = \begin{bmatrix} q - q^d \\ \dot{q} - \dot{q}^d \end{bmatrix}$, $\bar{A} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, $\bar{\eta} = \eta_0 + Eu$, $\eta_0 = E\ddot{q}^d + M^{-1}\Delta h$, $u = K\bar{e} + v$, $E = M^{-1}\hat{M} - I$, $\Delta h = \hat{h} - h$, $K = [-K_p, -K_v]$, and v is the neural network output.

We make the following assumptions (Spong & Vidyasagar, 1987) regarding the nominal model of the manipulator. (*A1*): For the inertia matrix M of the manipulator, there exist constants M_1 and M_2 such that $M_1 \leq \|M^{-1}\|_{T\infty} \leq M_2 < \infty$. (*A2*): There exists a nonnegative constant $\alpha < 1$ such that $\|M^{-1}\hat{M} - I\|_{T\infty} \leq \alpha$. (*A3*): There exist nonnegative constants δ and ρ such that $\|\hat{h} - h\|_{T\infty} \leq \delta\|x\|_{T\infty} + \rho$, where $x = [q^T, \dot{q}^T]^T$.

Let $G(s) = (sI - \bar{A})^{-1}B$. Then (B.1) can be expressed as: $\bar{e} = G\epsilon$, with $\epsilon = \bar{\eta} + u$, $u = K\bar{e} + v$, and $\bar{\eta} = \eta_0 + Eu$. Through algebraic operations, we obtain $\bar{e} = (I - GK)^{-1}G\bar{\eta} + (I - GK)^{-1}Gv$, $u = K(I - GK)^{-1}G\bar{\eta} + (K(I - GK)^{-1}G + I)v$. Let $P_1 = (I - GK)^{-1}G$, $P_2 = K(I - GK)^{-1}G$, and $P_3 = K(I - GK)^{-1} + I$, then we can express $\bar{e}$ and u as: $\bar{e} = P_1\bar{\eta} + P_1 v$, and $u = P_2\bar{\eta} + P_3 v$. Now taking the truncated L_∞-norm yields

$$\|\bar{e}\|_{T\infty} \leq \beta_1\|\bar{\eta}\|_{T\infty} + \beta_1\|v\|_{T\infty}, \tag{B.2}$$

$$\|u\|_{T\infty} \leq \beta_2\|\bar{\eta}\|_{T\infty} + \beta_3\|v\|_{T\infty}, \tag{B.3}$$

where $\beta_1 = \|(I - GK)^{-1}G\|_{T\infty}$, $\beta_2 = \|K(I - GK)^{-1}G\|_{T\infty}$, and $\beta_3 = \|K(I - GK)^{-1}G + I\|_{T\infty}$. Since $\eta_0 = E\ddot{q}^d + M^{-1}\Delta h$, from the modeling assumptions, we have $\|\eta_0\|_{T\infty} \leq M_2\delta\|\bar{e}\|_{T\infty} + b$, where $b = \alpha\|\ddot{q}^d\|_{T\infty} + M_2\delta\|x^d\|_{T\infty} + M_2\rho$, and $x^d = [(q^d)^T, (\dot{q}^d)^T]^T$. It follows that $\|\bar{\eta}\|_{T\infty} \leq M_2\delta\|\bar{e}\|_{T\infty} + \alpha\|u\|_{T\infty} + b$. Since the output of the neural network is bounded by construction (due to the $tanh$ activation function of the output units), let $\phi = \|v\|_{T\infty}$. Hence $\begin{bmatrix} \|\bar{\eta}\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} \leq \begin{bmatrix} M_2\delta\beta_1 & \alpha \\ \beta_2 & 0 \end{bmatrix} \begin{bmatrix} \|\eta_2\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} +$

$\begin{bmatrix} M_2\delta\beta_1\phi + b \\ \beta_3\phi \end{bmatrix}$. Let $Q = \begin{bmatrix} M_2\delta\beta_1 & \alpha \\ \beta_2 & 0 \end{bmatrix}$. Then $\det(I - Q) = 1 - M_2\delta\beta_1 - \alpha\beta_2 \equiv \Delta_m$. If $\Delta_m > 0$, then

$$\begin{bmatrix} \|\bar{\eta}\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} \leq \frac{1}{\Delta_m} \begin{bmatrix} 1 & \alpha \\ \beta_2 & 1 - M_2\delta\beta_1 \end{bmatrix} \begin{bmatrix} M_2\delta\beta_1\phi + b \\ \beta_3\phi \end{bmatrix}. \tag{B.4}$$

From (B.2) and (B.4), we obtain $\|\bar{e}\|_{T\infty} \leq \frac{\beta_1 b}{\Delta_m} + \frac{\beta_1\phi}{\Delta_m}(\beta_1 M_2\delta + \alpha\beta_3)$. Therefore, if the condition $\Delta_m > 0$ is satisfied, then $\bar{\eta}$, u, and $\bar{e}$ are bounded. Now recall that $\beta_1 = \|(I - GK)^{-1}G\|_{T\infty}$, and $\beta_2 = \|K(I - GK)^{-1}G\|_{T\infty}$. It can be seen that the condition $\Delta_m > 0$ can be satisfied by selecting sufficiently large values for K such that β_1 approaches zero and β_2 approaches unity simultaneously.

Appendix C

Proof of Corollary 1

From (1) and (5), we have $M\ddot{q}+h = \hat{M}(\ddot{q}^d+u)+\hat{h}$, where $u = K_v(\dot{q}^d-\dot{q})+K_p(q^d-q)+v$. So $\ddot{q} = E(\ddot{q}^d + u) + \ddot{q}^d + u + M^{-1}\Delta h$, where E and Δh are as defined in (B.1). From the modeling assumptions $A.1$-$A.3$, we have

$$\begin{aligned} \|\ddot{q}\|_{T\infty} &\leq \|E\|_{T\infty}\|\ddot{q}^d + u\|_{T\infty} + \|\ddot{q}^d + u\|_{T\infty} + \|M^{-1}\|_{T\infty}\|\Delta h\|_{T\infty} \\ &\leq (1+\alpha)(\|\ddot{q}^d\|_{T\infty} + \|u\|_{T\infty}) + M_2\delta\|\bar{e}\|_{T\infty} + M_2\delta\|x^d\|_{T\infty} + M_2\rho, \end{aligned}$$

where $x^d = [(q^d)^T, (\dot{q}^d)^T]^T$. Since u and $\bar{e}$ have been proved to be bounded (Theorem 1), it can be concluded that $\ddot{q}$ is bounded.

Appendix D

Proof of Corollary 2

From (9), we have $\Delta v = \ddot{q}^d - \ddot{q} + K\bar{e}$, where K is as specified in (B.1). Therefore, $\|\Delta v\|_{T\infty} \leq \|\ddot{q}^d\|_{T\infty} + \|\ddot{q}\|_{T\infty} + \|K\bar{e}\|_{T\infty}$. Since $\ddot{q}$ and $\bar{e}$ have been proven to be bounded, it can be concluded that Δv is also bounded.

Appendix E

Proof of Corollary 3

As described in Section 4 (with reference to Figure 4), we have

$$\dot{W}_{ij} = \lambda_n \Delta v_i g'(v_i)\bar{v}_j, \dot{R}_{jk} = \lambda_n g'(\bar{v}_j)\sum_{i=1}^{I_n}(\Delta v_i g'(v_i)W_{ij}),$$

and $\dot{S}_{kl} = \lambda_n g'(\bar{\bar{v}}_k)\sum_{j=1}^{J_n}\left(g'(\bar{v}_j)\sum_{i=1}^{I_n}(\Delta v_i g'(v_i)W_{ij}R_{jk})\right)$, where $g'(\cdot) = \frac{\partial g(\cdot)}{\partial(\cdot)}$. Since $\dot{W}_{ij}$ depends on $\lambda_n, \Delta v_i, g'$, and $\bar{v}_j$, which are all bounded, it can be concluded that $\dot{W}_{ij}$ and consequently W_{ij} are bounded during a given trial. Similarly, it can be concluded that $\dot{R}_{jk}$, R_{jk}, $\dot{S}_{kl}$ and S_{kl} are also bounded during a given trial.

Appendix F

Proof of Proposition 1

From Section 4, the dynamics of the neural network can be expressed as: $\dot{W}_{ij} = \lambda_n \Delta v_i g'(v_i)\bar{v}_j$, $\dot{R}_{jk} = \lambda_n g'(\bar{v}_j) \sum_{i=1}^{I_n} (\Delta v_i g'(v_i) W_{ij})$, and $\dot{S}_{kl} = \lambda_n g'(\bar{\bar{v}}_k) \sum_{j=1}^{J_n} \Big(g'(\bar{v}_j) \sum_{i=1}^{I_n} (\Delta v_i g'(v_i) W_{ij} R_{jk})\Big)$. During the subsequent development, the symbols λ_w, λ_r, and λ_s are used (instead of the general notation λ_n) to indicate that these learning rates are specifically associated with W_{ij}, R_{jk}, and S_{kl} respectively. Since during a given trial p, the signals driving the weights are bounded, i.e., $\|\Delta v_i(p,\xi)\|_{T\infty} \le \gamma_p < \infty$, $\|v_i(p,\xi)\|_{T\infty} \le \tilde{c}$, $\|\bar{v}_j(p,\xi)\|_{T\infty} \le \tilde{c}$, $\|\bar{\bar{v}}_k(p,\xi)\|_{T\infty} \le \tilde{c}$, and $\|g'(\cdot)\|_{T\infty} \le \tilde{c}$, where $\tilde{c}$ is the largest scaling factor, hence $\|\dot{W}_{ij}(p,\xi)\|_{T\infty} = \|\lambda_w \Delta v_i(p,\xi) g'(v_i(p,\xi))\bar{v}_j(p,\xi)\|_{T\infty} \le \lambda_w \tilde{c}^2 \gamma_p$, and $\|W_{ij}(p,\xi)\|_{T\infty} = \left\|W_{ij}(p,0) + \int_0^{\xi} \dot{W}_{ij}(p,\sigma)d\sigma\right\|_{T\infty} \le \lambda_w \tilde{c}^2 \gamma_p \xi + \gamma_p^w \equiv \psi_p^w$. Similarly, $\|\dot{R}_{jk}(p,\xi)\|_{T\infty} \le \lambda_r \tilde{c}^2 I_n \gamma_p \psi_p^w$, $\|R_{jk}(p,\xi)\|_{T\infty} \le \lambda_r \tilde{c}^2 I_n \gamma_p \psi_p^w \xi + \gamma_p^r \equiv \psi_p^r$, and $\|\dot{S}_{kl}(p,\xi)\|_{T\infty} \le \lambda_s \tilde{c}^3 I_n J_n \gamma_p \psi_p^w \psi_p^r$. From the definitions presented in Section 5, we have $\Delta w_{ij}(p,\xi) = \int_0^{\xi} \dot{W}_{ij}(p,\sigma)d\sigma$, $\Delta r_{jk}(p,\xi) = \int_0^{\xi} \dot{R}_{jk}(p,\sigma)d\sigma$, and $\Delta s_{kl}(p,\xi) = \int_0^{\xi} \dot{S}_{kl}(p,\sigma)d\sigma$.

Now for two successive trials p and $(p+1)$, $\|\Delta w_{ij}(p+1,\xi) - \Delta w_{ij}(p,\xi)\|_{T\infty} \le \left\|\int_0^{\xi} \dot{W}_{ij}(p+1,\sigma)d\sigma\right\|_{T\infty} + \left\|\int_0^{\xi} \dot{W}_{ij}(p,\sigma)d\sigma\right\|_{T\infty} \le \lambda_w \tilde{c}^2(\gamma_{p+1} + \gamma_p)$. Therefore, to satisfy $\|\Delta w_{ij}(p+1,\xi) - \Delta w_{ij}(p,\xi)\|_{T\infty} << 1$, λ_w can be chosen such that $\lambda_w << \frac{1}{\tilde{c}^2(\gamma_{p+1}+\gamma_p)}$. Similarly, λ_r and λ_s can be chosen as: $\lambda_r << \frac{1}{\tilde{c}^2 I_n(\gamma_{p+1}\psi_{p+1}^w + \gamma_p \psi_p^w)}$, and $\lambda_s << \frac{1}{\tilde{c}^3 I_n J_n(\gamma_{p+1}\psi_{p+1}^w \psi_{p+1}^r + \gamma_p \psi_p^w \psi_p^r)}$. It can be seen that by choosing λ_n to be $\lambda_n = \min[\lambda_w, \lambda_r, \lambda_s]$, the general condition, $\|\Delta\theta_m(p+1,\xi) - \Delta\theta_m(p,\xi)\|_{T\infty} << 1$, can be met.

Acknowledgments

This work is financially supported by the Institute of Robotics and Intelligent Systems, the Ontario Graduate Scholarship, and the Natural Sciences and Engineering Research Council of Canada. The authors wish to thank Professor B. Benhabib for the use of the computing facility at the Computer Integrated Manufacturing Laboratory, Department of Mechanical Engineering, University of Toronto.

References

Arai, F., Rong, L. & Fukuda, T. (1993). Asymptotic Convergence of Feedback Error Learning Method and Improvement of Learning Speed. *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, 761-767.

Cılız, M.K.& Işık, C. (1990). Trajectory Following Control of Robotic Manipulators Using Neural Networks. *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, Philadelphia, Pennsylvania, 536-540.

Corless, M.& Leitmann, G. (1981). Continuous State Feedback Guaranteeing Uniform Ultimated Boundedness for Uncertain Dynamic Systems. *IEEE Transactions on Automatic Control*, AC-26, No. 5, 1139-1144.

Craig, J. (1986). *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley.

Cybenko, G. (1989). Approximation by Superposition of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, Vol. 2, 303-314.

Denker, J., Schwartz, D., Witter, B., Solla, S., Howard, R. & Jackel, L. (1987). Large Automatic Learning, Rule Extraction, and Generalization. *Complex Systems*, Vol. 1, 877-922.

Dorf, R.C. (1992). *Modern Control Systems.* Reading, Massachusetts: Addison-Wesley Publishing Company.

Glesner, M.& Pöchmüller, W. (1994). *Neurocomputers: An Overview of Neural Networks in VLSI.* London: Chapman & Hall.

Gomi, H.& Kawato, M. (1990). Learning Control for a Closed Loop System Using Feedback-Error-Learning. *Proceedings of the 29th Conference on Decision and Control*, Honolulu, Hawaii, 3289-3294, 1990.

Kuan, A.& Bavarian, B. (1992). Control of Universal 3DOF Robot Manipulator: A Neural Network Approach. *Proceedings of the Japan-USA Symposium on Flexible Automation*, San Francisco, CA, 249-255.

Lawryshyn, Y.A. (1993). *Robot Link and Contact Surface Dynamic Interaction: An Experimental Study of Contact Stability.* M.A.Sc. Thesis, Department of Mechanical Engineering, University of Toronto.

Lokhorst, D. M. (1990). *An Approach to Force and Position Control of Robots: Theory and Experiments.* M.A.Sc. Thesis, Department of Mechanical Engineering, University of Toronto, Toronto, Canada.

McClelland, J.L.& Rumelhart, D.E. (Eds.), (1986). *Parallel Distributed Processing.* Vol. 1, Cambridge, Massachusetts: MIT Press.

Miller, W.T., Sutton, R.S. & Werbos, P.J. (Eds.), (1990). *Neural Networks for Control.* Cambridge, Massachusetts: MIT Press.

Muller, B. & Reinhardt, J. (1990). *Neural Networks: An Introduction.* Berlin: Springer-Verlag, 1990.

Okuma, S.& Ishiguro, A. (1990). A Neural Network Compensator for Uncertainties of Robotic Manipulators. *Proceedings of the 29th Conference on Decision and Control*, Honolulu, Hawaii, 3303-3307.

Ortega, R.& Spong, M.W. (1989). Adaptive Motion Control of Rigid Robots: a Tutorial. *Automatica*, Vol.25, No.6, 877-888.

Spong, M.W.& Vidyasagar, M. (1989). *Robot Dynamics and Control.* New York; John Wiley & Sons.

Spong, M.W.& Vidyasagar, M. (1987). Robust Linear Compensator Design for Nonlinear Robotic Control. *IEEE Journal of Robotics and Automation*, Vol. RA-3, No. 4, 345-351.

Vidyasagar, M., (1992). *Nonlinear Systems Analysis.* 2nd ed., New Jersey: Prentice Hall.

Yabuta, T.& Yamada, T. (1991). Learning Control Using Neural Networks. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, 740-745.

Yanovski, G., (1991). *Experimental Analysis of a Robust Centralized/Decentralized Controller for Robotic Manipulators.* M.E. Thesis, Department of Electrical Engineering, University of Toronto.

Zomaya, A.Y.& Nahban, T.M. (1993). Centralized and Decentralized Neuro-Adaptive Robot Controllers. *Neural Networks*, Vol. 6, 223-244.