Parameter Uncertainty Compensation in Robot Trajectory Tracking: a Neural Network Approach

Peter C.Y. Chen
Department of
Mechanical Engineering
University of Toronto
5 King's College Road
Toronto, Canada M5S 1A4

James K. Mills
Department of
Mechanical Engineering
University of Toronto
5 King's College Road
Toronto, Canada M5S 1A4

Kenneth C. Smith
Department of Electrical
and Computer Engineering
University of Toronto
10 King's College Road
Toronto, Canada M5S 1A4

Abstract **An approach employing a multilayer feedforward neural network (with the error-backpropagation algorithm) for uncertainty compensation in the problem of robot trajectory tracking is proposed. It is proved, and verified through computer simulation, that the resulting closed-loop system (with a neural network as the uncertainty compensator) is stable in the sense that all signals in the closed-loop system are bounded, while the performance of the closed-loop system improves as the neural network learning process iterates.**

I. Introduction

To control a robotic system such that the system behaves exactly as desired requires precise knowledge of the system. The complexity of robotic systems, however, renders the acquisition of such precise knowledge impossible. It is thus conventional (and has been demonstrated effective [8]) in practice to derive control strategies based on an approximate mathematical model representing the physical system. The discrepancy between the model and the actual system is referred to as the uncertainty associated with the robotic system.

For the robot trajectory tracking problem, this concept of uncertainty can be quantified in the context of the feedback linearization control technique [8]. In general, the dynamics of a unconstrained serial manipulator with n joints can be described by a set of nonlinear differential equations, compactly expressed in the form

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

where $q \in R^n, \dot{q} \in R^n$, and $\ddot{q} \in R^n$ are respectively the joint position, joint velocity, and joint acceleration vectors, $M(\cdot) \in R^{n\times n}$ is the inertia matrix, $h(\cdot) \in R^n$ is a vector containing the Coriolis and gravitation terms, and $\tau \in R^n$ is the input torque vector. Let $\tau = \hat{M}(\ddot{q}^d + u) + \hat{h}$, and $u = K_v\dot{e}_1 + K_p e_1 + v$, where $e_1 = q^d - q$, q^d is the desired trajectory, $\hat{M}$ and $\hat{h}$ are the estimates of M and h respectively, K_v and K_p are constant gains, and v is the compensating signal to be determined, then the error dynamics of the closed-loop system can be expressed as

$$\dot{e} = Ae + B\Delta v, \tag{2}$$

where $e = \begin{bmatrix} e_1 \\ \dot{e}_1 \end{bmatrix}, A = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}, B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, and $\Delta v = \eta(q,\dot{q},\ddot{q}) - v$ is referred to as the *control error*, with $\eta = (\hat{M}^{-1}M - I)\ddot{q} + \hat{M}^{-1}(h - \hat{h})$ being referred to as the uncertainty. The problem is to generate the appropriate siganl v so that $\Delta v \to 0$. Figure 1 illustrates this idea of uncertainty compensation.

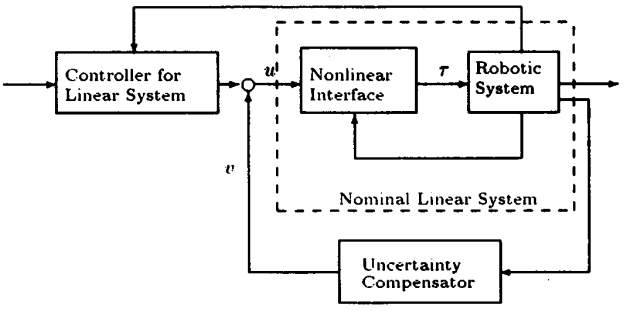

Figure 1 Uncertainty Compensation.

In recent years, there have been reports on applying neural networks for robotic control, e.g. [3, 5]. The most common approach appears to be using neural networks to model the inverse dynamics of a manipulator, e.g. [5]. An approach using a neural network for uncertainty compensation is proposed in [7], where the effectiveness of the

Manuscript received July 1, 1993. This work was financially supported by IRIS, NSERC, and OGS.

neural network as a uncertainty compensator is demonstrated only through computer simulation. Among the reported studies on the application of neural networks for robotic control, one element that is consistently missing is theoretical analyses which address the stability and performance aspects of the closed-loop system embedded with a neural network.

In this paper, we propose an approach using a multilayer feedforward neural network for uncertainty compensation in the context of the robot trajectory following problem. Using a multi-loop version of the small gain theorem [2], we prove that the closed-loop system (with the neural network learning on-line) is stable in sense that all signals in the closed-loop system are bounded. We further prove that under certain assumption, the performance of the closed-loop system improves as the neural network learning process iterates.

This paper is organized as follows. Section II presents the proposed approach. Section III contains the stability proof. Section IV contains the performance analysis. Section V presents simulation results which substantiate the theoretical results of Sections III and IV. Section VI concludes the paper.

II. Approach

A. A Function Approximation Problem

Note that $\Delta v = 0$ implies $v = \eta(q, \dot{q}, \ddot{q})$. Although the structure of the function $\eta(\cdot)$ is known, the exact values of the parameters of this function are not explicitly known. Consequently, it is not possible to predict the exact output of this function. An ideal compensator N is a function whose output v exactly equals that of the function $\eta(\cdot)$ so that $\Delta v = 0$. Based on such a premise, the problem of designing N can then be considered as a function approximation problem.

A multilayer feedforward neural network (with the error-backpropagation learning algorithm) represents an attractive mechanism for dealing with such a function approximation problem, mainly because of its ability to learn [4].

B. Neural Network Structure

A multilayer feedforward neural network consists of a collection of processing elements (or units) arranged in a layer structure as shown in Figure 2.

For the neural network with two hidden layers (Figure 2), the network output is generated according to: $v_i = g_1\left(\sum_{j=1}^{J_n} W_{ij} g_2\left(\sum_{k=1}^{K_n} R_{jk} g_3\left(\sum_{l=1}^{L_n} S_{kl} z_l\right)\right)\right)$, where $g_m(x) = c_m \tanh(\gamma_m x)$. For convenience we define a *generalized* weight vector Θ as: $\Theta = [W_1, ..., W_{I_n}, R_1, ..., R_{J_n}, S_1, ..., S_{K_n}] \in R^{c_\theta}$, where $(\cdot)_i$ represents the i^{th} row of the matrix $(\cdot)$, and $c_\theta = I_n \times J_n + J_n \times K_n + K_n \times L_n$, then the mapping realized by the network can be compactly expressed as: $v = g(Z, \Theta)$, where Z is the input vector, i.e. $Z = [z_1, z_2, ...z_l, ...z_{L_n}]^T$.

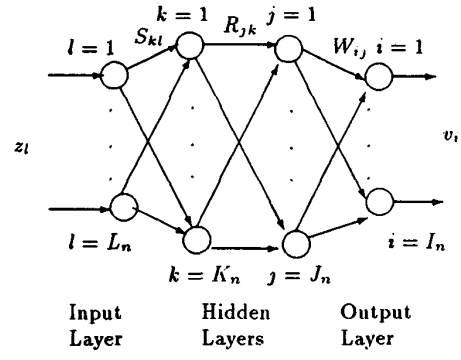

Figure 2 Neural Network Structure.

C. Neural Network for Uncertainty Compensation

To approximate the function $\eta(q, \dot{q}, \ddot{q})$, the neural network takes $q, \dot{q}$, and $\ddot{q}$ as its input, and produce an output v. Thus for a n-joint manipulator, the numbers of units in the input layer and the output layer are respectively $3n$ and n, i.e. $L_n = 3n, I_n = n$.

Note that the control error Δv represents the difference between the output of the uncertainty function $\eta(\cdot)$ and the output of the neural network $N(\cdot)$. The objective of neural network learning is then to effectively adjust the weights of the neural network to minimize the control error Δv. The error-backpropagation algorithm [4, 6] is an effective algorithm for neural network learning.

Let the cost function to be minimized be: $J_{\Delta v} = \frac{1}{2}\Delta v^T \Delta v$. Now proper application of the error-backpropagation algorithm yields the weight update rule: $\dot{\Theta} = -\lambda_n \Delta v^T \frac{\partial \Delta v}{\partial \Theta}$, where λ_n is the learning rate. Since $\Delta v = v^d - v$ and $\frac{\partial v^d}{\partial \Theta} = 0$, where v^d is the desired output of the network, the 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 as: $\dot{W}_{ij} = \lambda_n \Gamma_i v_j$, $\dot{R}_{jk} = \lambda_n \Gamma_j \bar{v}_k$, $\dot{S}_{kl} = \lambda_n \bar{\Gamma}_k z_k$, where $\Gamma_i = \Delta v_i g_1'$, $\Gamma_j = g_2' \sum_{i=1}^{I_n} \Gamma_i W_{ij}$, $\bar{\Gamma}_k = g_3' \sum_{j=1}^{J_n} \Gamma_j R_{jk}$, with $g_m'(\cdot) = \frac{\partial g_m(\cdot)}{\partial(\cdot)}$.

It is noted that this proposed learning scheme is similar in form to that presented in [7]. The difference is that in [7], the error signal for neural network learning is derived using the nominal model of the manipulator, which results in more computational overhead than the approach we proposed here. In our proposed approach, the error signal for neural network learning (i.e. the control error Δv) is constructed from (2), i.e.

$$\Delta v = \eta - v = \dot{e}_1 + K_v e_1 + K_p e_1. \qquad (3)$$

Note that (3) contains the joint acceleration vector $\ddot{q}$, which, though not directly available from the robotic sys-

tem output, can be estimated by using appropriate filtering techniques.

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

$$\begin{cases} \dot{e} = Ae + B\Delta v(q,\dot{q},\ddot{q},\Theta) \\ \dot{\Theta} = -\lambda_n \Delta v^T(q,\dot{q},\ddot{q},\Theta)\frac{\partial \Delta v(q,\dot{q},\ddot{q},\Theta)}{\partial \Theta}. \end{cases} \quad (4)$$

It is generally accepted that, for the error-backpropagation to function properly, the learning rate λ_n must be small [6]. Consequently, the learning process must be conducted in a repetitive manner (i.e. through a series of trials). In other words, the robot executes a task repetitively while the neural network learns on-line to generate the signal v to counteract the uncertainty η.

III. Stability Analysis

A. Preliminaries

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 denotes the eigenvalue of the matrix A. The L_∞^n-norm of a Lebesque integrable function $f(t) : R_+ \to R^{n\times n}$, denoted by $\|f\|_\infty$, is defined as: $\|f\|_\infty = ess\,sup_{t\in[0,\infty)}\|f(t)\| < \infty$. The *extended* L_∞^n-space (for the truncated L_∞^n-norm), denoted by $L_{\infty e}^n$, is defined as: $L_{\infty e}^n = \{f : R_+ \to R^{n\times n} | f_T \in L_\infty^n, \forall T < \infty\}$, where $f_T(t) = \begin{cases} f(t) & \text{for } t \in [0,T] \\ 0 & \text{otherwise} \end{cases}$. For convenience, we use $\|f\|_{T\infty}$ to denote $\|f_T\|_\infty$. The L_∞^n-norm of a transfer matrix P is defined as: $\|P\|_\infty = sup_{x\in L_\infty^n - 0}\frac{\|Px\|_\infty}{\|x\|_\infty}$. In other words, $\|P\|_\infty$ is the L_∞-gain of P. Let β denote the norm $\|P\|_\infty$, then $\|Px\|_{T\infty} \le \beta\|x\|_{T\infty}$. A system with input x and output y is said to be bounded-input bounded-output (BIBO) stable if for every $x \in L_\infty$, $y \in L_\infty$.

To prove the stability of the closed-loop system, we utilize the method presented in [9]. Through algebraic operations, the error dynamics of the closed-loop system can alternatively be expressed as

$$\begin{cases} \dot{\bar{e}} = \bar{A}\bar{e} + B(\bar{\eta} + u) \\ \bar{\eta} = \eta_0 + Eu \\ \eta_0 = E\ddot{q}^d + M^{-1}\Delta h \\ u = K\bar{e} + v \end{cases} \quad (5)$$

where $E = M^{-1}\dot{M} - I$, $\bar{e}_1 = q - q^d$, $\Delta h = h - \hat{h}$, $\bar{e} = \begin{bmatrix} \bar{e}_1 \\ \dot{\bar{e}}_1 \end{bmatrix}$, $\bar{A} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ I \end{bmatrix}$, $K = \begin{bmatrix} -K_p & 0 \\ 0 & -K_v \end{bmatrix}$, and v is the neural network output.

We make the following assumptions regarding the nominal model of the manipulator, as in [9].

Assumption 1: For the inertia matrix M of the manipulator (1), there exist constants M_1 and M_2 such that $M_1 \le \|M(q)^{-1}\| \le M_2 < \infty$.

Assumption 2: There exists a nonnegative constant $\alpha < 1$ such that $\|M(q)^{-1}\hat{M}(q) - I\| \le \alpha$.

Assumption 3: There exist nonnegative constants δ and ρ such that $\|\hat{h}(q,\dot{q}) - h(q,\dot{q})\| \le \delta\|x\| + \rho$, where $x = [q^T, \dot{q}^T]^T$.

B. Main Results

Theorem 1: There exist constant real matrices K_v and K_p such that the closed-loop system (4) remains BIBO stable for a bounded reference trajectory $[q^d(t), \dot{q}^d(t), \ddot{q}^d(t)]$ for all $t \in [0,T]$.

Proof: Let $G(s) = (sI - \bar{A})^{-1}B$. Then (5) can be expressed as: $\bar{e} = G\epsilon$, $\epsilon = \bar{\eta} + u$, $u = K\bar{e} + v$, and $\bar{\eta} = \eta_0 + Eu$. Through algebraic operations, we obtain $\bar{e} = (I - GK)^{-1}G\bar{\eta} + (I - GK)^{-1}Gv$, and $u = K(I - GK)^{-1}G\bar{\eta} + (K(I - GK)^{-1}G + I)v$. Let $P_1 = (I - GK)^{-1}G$, $P_2 = K(I - GK)^{-1}G$, and $P_3 = K(I - GK)^{-1} + I$, then $\bar{e} = P_1\bar{\eta} + P_1 v$, and $u = P_2\bar{\eta} + P_3 v$. Thus, from the definitions in Section III-A, we have

$$\|\bar{e}\|_{T\infty} \le \beta_1\|\bar{\eta}\|_{T\infty} + \beta_1\|v\|_{T\infty}, \quad (6)$$

$$\|u\|_{T\infty} \le \beta_2\|\bar{\eta}\|_{T\infty} + \beta_3\|v\|_{T\infty}. \quad (7)$$

where $\beta_1 = \|(I - GK)^{-1}G\|_{T\infty}$, $\beta_2 = \|K(I - GK)^{-1}G\|_{T\infty}$, and $\beta_3 = \|K(I - GK)^{-1}G + I\|_{T\infty}$. From (5) and the modeling assumptions, we have $\|\eta_0\|_{T\infty} \le M_2\delta\|\bar{e}\|_{T\infty} + b$, where $b = \alpha\|\ddot{q}^d\|_{T\infty} + M_2\delta\|x^d\|_{T\infty} + M_2\rho$. It follows that

$$\|\bar{\eta}\|_{T\infty} \le M_2\delta\|\bar{e}\|_{T\infty} + \alpha\|u\|_{T\infty} + b. \quad (8)$$

Since the output of the neural network is bounded by construction (due to the logistic activation function of the output units), let $\phi = \|v\|_{T\infty}$. Substituting (6) into (8) yields

$$\begin{bmatrix} \|\bar{\eta}\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} \le \begin{bmatrix} M_2\delta\beta_1 & \alpha \\ \beta_2 & 0 \end{bmatrix}\begin{bmatrix} \|\eta_2\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} + \begin{bmatrix} M_2\delta\beta_1\phi + b \\ \beta_3\phi \end{bmatrix}. \quad (9)$$

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

$$\begin{bmatrix} \|\bar{\eta}\|_{T\infty} \\ \|u\|_{T\infty} \end{bmatrix} \le \frac{1}{\Delta}\begin{bmatrix} 1 & \alpha \\ \beta_2 & 1 - M_2\delta\beta_1 \end{bmatrix}\begin{bmatrix} M_2\delta\beta_1\phi + b \\ \beta_3\phi \end{bmatrix}. \quad (10)$$

From (6) and (10), we obtain

$$\|\bar{e}\|_{T\infty} \le \frac{\beta_1 b}{\Delta} + \frac{\beta_1\phi}{\Delta}(\beta_1 M_2\delta + \alpha\beta_3). \quad (11)$$

Therefore, if the condition $\Delta > 0$ is satisfied, then the uncertainty $\bar{\eta}$, the control signal u, and the trajectory tracking error $\bar{e}$ are bounded during each trial. It is thus clear that a sufficient condition for BIBO stability of the closed-loop system is $\Delta > 0$.

Remark: 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 $\Delta > 0$ can be satisfied by selecting sufficiently large values for K such that β_1 approaches zero and β_2 approaches unity simultaneously.

Here we see that the selection of K has nothing to do with the network output v. Thus the closed-loop system can first be made BIBO stable by selecting appropriate values for K, and then its performance can be improved by properly training the neural network to generate the correct compensating signal.

Corollary 1: The joint acceleration $\ddot{q}(t)$ is bounded for all $t \in [0, T]$.

Theorem 2: The state of the neural network (i.e. the weights) remains bounded for all $t \in [0, T]$.

Remark: The proofs for Corollary 1 and Theorem 2 follow from Theorem 1, and are omitted here due to space limitation.

IV. Performance Analysis

A. Preliminaries

Let t represents 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[1], i.e. $0 \leq \xi \leq T$. With slight abuse of notation, let $x(p, \xi)$ denote the value of the variable x at the ξ^{th} second of the p^{th} trial. Then $x(p, 0)$ and $x(p, T)$ represent the value of the variable x at the beginning and the end of the p^{th} trial respectively. Note that $x(p, 0) = x(p-1, T)$.

Let $C[0, T]$ denote the family of Lebesgue integrable functions $f_i(\xi)$ for all $\xi \in [0, T]$. We define 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 $[0, T]$ as: $\|f(\xi)\|_{2T} = \left(\int_0^T f^T(\xi) f(\xi) d\xi\right)^{\frac{1}{2}}$. For convenience, we shall use $\|\cdot\|$ instead of $\|\cdot\|_{2T}$ to denote this norm in the subsequent analysis.

Assumption 4: For the system (4) with $\lambda_n << 1$, the following approximation holds: $\int_0^\xi \dot{\Theta}(p+1, \sigma) d\sigma \approx \int_0^\xi \dot{\Theta}(p, \sigma) d\sigma$, $\forall \xi \in [0, T]$.

[1] 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.

Remark: This assumption (illustrated in Figure 3) means that, for $\lambda_n << 1$, the difference between the weight change of any two *successive* trials is considered negligible.

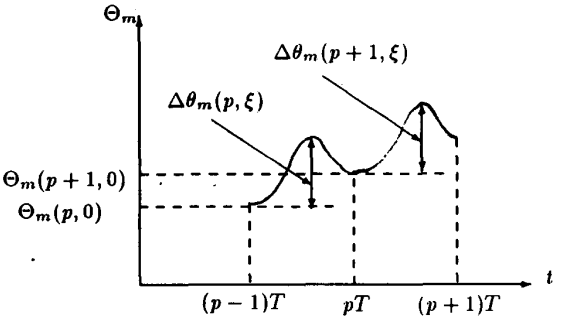

Figure 3 Weight and Weight Change.

Let Θ_m be an element of Θ, and let $\Delta\theta_m(p+1, \xi)$ represent the amount of weight *change* during the first ξ seconds of the $(p+1)^{th}$ trial, and $\Delta\theta_m(p, \xi)$ represent the amount of weight *change* during the first ξ seconds of the p^{th} trial. Then $\Delta\theta_m(p+1, \xi) = \int_0^\xi \dot{\Theta}_m(p+1, \sigma) d\sigma$, and $\Delta\theta_m(p, \xi) = \int_0^\xi \dot{\Theta}_m(p, \sigma) d\sigma$. Although the difference between the initial and final values of Θ_m, i.e. $\Theta_m(p+1, 0) - \Theta_m(p, 0)$, of any trial could be significant, the difference between the *change* in Θ_m of any two successive trials can be considered negligible, i.e. $\Delta\theta_m(p+1, \xi) \approx \Delta\theta_m(p, \xi)$.

B. Main Results

Theorem 3: Under Assumption 4, the L_2-norm of the control error Δv decreases as the number of trials p increases, i.e. $\|\Delta v(p+1, \xi)\| < \|\Delta v(p, \xi)\|$.

Proof: Recall that the cost function to be minimized by the neural network learning process is $J_{\Delta v}(p, \xi) = \frac{1}{2}\Delta v^T(p, \xi)\Delta v(p, \xi)$. From the definition of the L_2-norm, we see that $\|\Delta v(p, \xi)\|^2 = 2\int_0^T J_{\Delta v}(p, \xi) d\xi$. Now $\|\Delta v(p+1, \xi)\|^2 - \|\Delta v(p, \xi)\|^2 = 2\int_0^T (J_{\Delta v}(p+1, \xi) - J_{\Delta v}(p, \xi)) d\xi$. 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 $J_{\Delta v}(p, \xi)$ to obtain: $J_{\Delta v}(p+1, \xi) - J_{\Delta v}(p, \xi) \approx \sum_{m=1}^{c_\theta} \left[\frac{\partial J_{\Delta v}}{\partial \Theta_m} (\Theta_m(p+1, \xi) - \Theta_m(p, \xi))\right]$.

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 $\Theta_m(p+1, 0)$ represents the weight value at the *end* of the p^{th} trial, $\Theta_m(p, 0)$ represents the weight value at the *beginning* of the p^{th} trial. $\Delta\theta_m(p+1, \xi)$ 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. Now since $\Delta\theta_m(p+1,\xi) = \int_0^\xi \dot{\Theta}_m(p+1,\sigma)d\sigma$, and $\Delta\theta_m(p,\xi) = \int_0^\xi \dot{\Theta}_m(p,\sigma)d\sigma$, from Assumption 4, we have $\Delta\theta_m(p+1,\xi) \approx \Delta\theta_m(p,\xi)$. Therefore, with $\frac{\partial J_{\Delta v}(p,\xi)}{\partial \Theta_m(p,\xi)} = \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}$, we have

$$\begin{aligned}
&\|\Delta v(p+1,\xi)\|^2 - \|\Delta v(p,\xi)\|^2 \\
&= 2\sum_{m=1}^{c_\theta}[\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)} \\
&\quad \times(\Theta_m(p+1,0) - \Theta_m(p,0))d\xi] \\
&= 2\sum_{m=1}^{c_\theta}[(\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi) \\
&\quad \times(\Theta_m(p+1,0) - \Theta_m(p,0))]. \qquad (12)
\end{aligned}$$

We note that $\Theta_m(p+1,0) - \Theta_m(p,0)$ represents the *total* weight change during the p^{th} trial, therefore, $\Theta_m(p+1,0) - \Theta_m(p,0) = \int_0^T \dot{\Theta}_m(p,\xi)d\xi$. But from the neural network weight update rule, $\dot{\Theta}_m(p,\xi) = -\lambda_n \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}$. So

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

Now $\|\Delta v(p+1,\xi)\|^2 - \|\Delta v(p,\xi)\|^2 = 0$ if $\int_0^T \Delta v^T(p,\xi)\frac{\partial \Delta v(p,\xi)}{\partial \Theta_m(p,\xi)}d\xi = 0$. But this implies that $\int_0^T \dot{\Theta}_m d\xi = 0$, which in turn implies 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 $\|\Delta v(p+1,\xi)\|^2 - \|\Delta v(p,\xi)\|^2 < 0$, that is, $\|\Delta v(p+1,\xi)\| < \|\Delta v(p,\xi)\|$.

V. Simulation

Computer simulation has been conducted for the trajectory following problem to verify the theoretical results discussed in Sections III and IV. A simple two-link planar robot (Figure 4) [1] was used in the simulation.

A control system (equivalent to that shown in Figure 1) was setup in the *EASY5* environment[2]. An error of 10% was introduced in the mass of the first link, and an error of 12.5% was introduced in the mass of the second link. As a result, $\eta \neq 0$. The neural network used in the simulation has six input units, two hidden layers with twenty units in each hidden layer, and two output units. The learning rate λ_n was set at 0.0005. The gains in for the PD controller were set to be $K_v = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, K_p = \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix}$. The desired trajectory for each joint was generated by a fifth-order polynomial. The initial weights of the neural network for the first trial were set to arbitrary values in the order of 10^{-4}. A series of simulation runs (i.e. trials) were carried out.

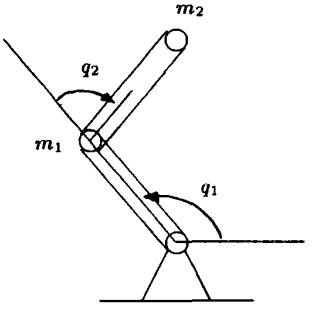

Figure 4 A Two-link Planar Manipulator.

Figure 5 shows the dynamics of one connection weight, $W_{(10,1)}$, during the 10^{th} trial (thin line) and the 11^{th} trial (thick line). Figure 6 shows the dynamics of the same connection weight during the 20^{th} (thin line) trial and the 21^{st} trial (thick line). It can be seen that the *difference* of the weight change between two *successive* trials (i.e. 10^{th} and 11^{th}, 20^{th} and 21^{st}) is negligible. Comparing these figures to Figure 3 (which illustrates the assumption that the difference of the weight change between any two successive trials is negligible), it is evident that the neural network does indeed possess the dynamic behavior as assumed.

Figure 7 shows the L_2-norm of the control error Δv versus the trial number p. It can be seen that the control error Δv decreases as the number of trials p increases. This confirms the theoretical conclusion presented in Section IV. Figure 8 shows the L_2-norm of the trajectory tracking error ϵ versus the trial number p. It can be seen the reduction in the control error eventually translates to reduction in the trajectory tracking error.

VI. Summary

In this paper, an approach using a multilayer feedforward neural network with the error-backpropagation learning algorithm for uncertainty compensation in the context

[2]The *Boeing EASY5 Engineering Analysis System (EASY5)* is an interactive computer program for the modeling, analysis, and design of large complex dynamical systems defined by algebraic, differential, and/or difference equations. The software version used for the simulations in this work was *EASY5x, Version 2.0.920406*.

of the robot trajectory following problem has been proposed. It has been shown that the error-backpropagation algorithm for supervised learning can be directly applied. More significantly, it has been proved that the closed-loop system with the neural network learning on-line is stable in the sense that all signals in the closed-loop system are bounded, and that, under certain assumption, the performance of the closed-loop system improves as the number of learning trials increases. These theoretical results have been substantiated by computer simulation.

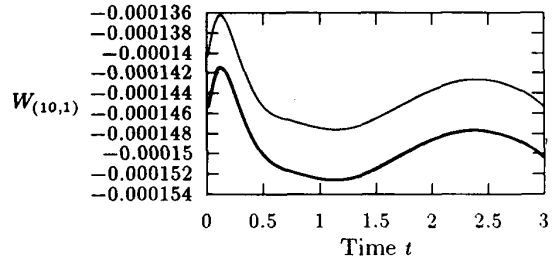

Figure 5 Weight Dynamics: Trials 10^{th} and 11^{th}.

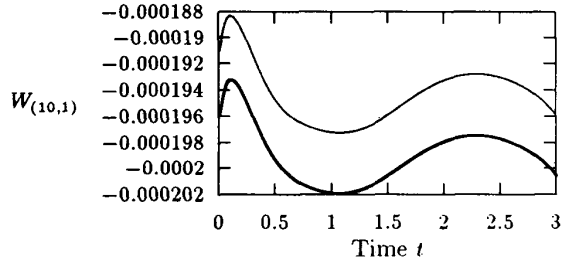

Figure 6 Weight Dynamics: Trials 20^{th} and 21^{st}.

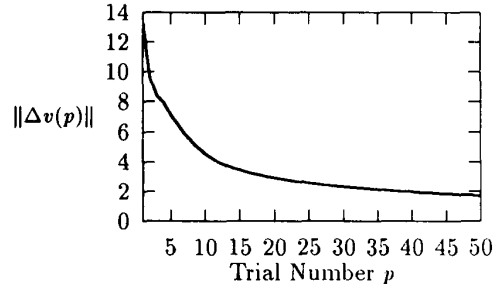

Figure 7 Control Error v.s. Trial Number.

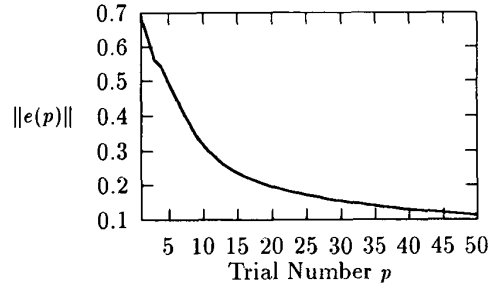

Figure 8 Tracking Error v.s. Trial Number.

References

[1] Craig, J., *Introduction to Robotics: Mechanics and Control.* Reading, MA: Addison-Wesley, 1986.

[2] Desoer, C.A. and M. Vidyasagar, *Feedback Systems: Input-Output Properties.* New York: Academic, 1975.

[3] Fukuda, T. and T. Shibata, Neuromorphic Control for Robotic Manipulators: Position, Force and Impact Control. *Proceedings of the 5th IEEE International Symposium on Intelligent Control,* (Philadelphia, Pennsylvania), 310-315, 1990.

[4] Hertz, J., A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation.* Addison-Wesley: Redwood City, California, 1991.

[5] Kawato, M., Computational Schemes and Neural Network Models for Formation and Control of Multijoint Arm Trajectory. In *Neural Networks for Control,* ed. W.T, Miller, III, R.S. Sutton, and P.J. Werbos, Cambridge, Massachusetts: MIT Press, 197-228, 1990.

[6] McClelland, J.L. and D.E. Rumelhart (eds), *Parallel Distributed Processing,* Vol. 1, Cambridge, Massachusetts: MIT Press, 1986.

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

[8] Spong, M.W. and M. Vidyasagar, *Robot Dynamics and Control.* New York: John Wiley & Sons, 1989.

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