A Two-Time-Scale Neural Controller for the Tracking Control of Rigid Manipulators

Weiying Cheng and John Ting-Yung Wen, Senior Member, IEEE

Abstract—In this paper, a two-time-scale neural controller applied to the tracking control of rigid manipulators is introduced. Several fast learning rules and slow learning strategies are proposed. The stability properties of the closed loop system with the proposed two-time-scale neural controller are analyzed. The results show that the tracking error will be uniformly bounded and converge to a bounded region. If a sufficiently large leakage term is used in the fast learning rule, the ultimate bound of the tracking error depends only on the accuracy of the slow learning. Moreover, the feasibility of the proposed neural controller is demonstrated through the simulation of a two-link rigid robot manipulator.

I. INTRODUCTION

COMMANDING the state of a nonlinear system to follow a desired trajectory is an important problem that has application in many fields. In most of the control strategies proposed for this problem, a highly model dependent feedforward control is typically needed. With the well known property of a neural network as a universal function approximator [1], [2], it is natural to try to apply a neural network to approximate this feedforward. Indeed, many papers have appeared to address this problem. Most of the past approaches can be classified into the following categories.

1) Direct Learning of Inverse Dynamics [3]–[6]: The inverse dynamics of the nonlinear system is learned by a neural network through supervised training. After training is finished, the trained network is used as a controller to drive the plant tracking the desired trajectory.

2) Learning Based on Output Tracking Error: This approach tries to update the weights of the neural controller by minimizing an objective function that measures the tracking error. In order to use back propagation as the learning rule, the gradient of the output of the plant with the weights of the neural controller is generally required. However, as the model is assumed unknown or partially known, this gradient is not available. To address this problem, one method is to train another neural network to approximate the forward dynamics of the plant, then the gradient can be obtained on this network [7], [8]. Another method is to use perturbation method to obtain this gradient [9], [4]. Requirement of the gradient can also be avoided by using the feedback torque to approximate the torque error signal for the training of the neural control [10].

Both of these approaches have the following drawbacks.

1) Large Number of Training Samples: The teacher for training the inverse dynamics neural network is the desired control commands. However, these desired control commands are not available. Some of the past approaches involves exciting the plant by using random input and then, together with the corresponding output, the net is trained. This usually leads to an unnecessarily large set of learning samples in order to cover the desired control commands. The learning of the forward dynamics has the same problem.

2) Difficulty in Approximating the Gradient: The gradient of the plant output with respect to the weights of the neural controller has to be approximated for the neural controllers that learn the feedforward based on the output tracking error. However, the current approximation of this gradient ignores the fact that the plant is a dynamical system.

3) One Time-Scale Learning: Most current neural controllers update all their weights in only one time-scale. Therefore, all of the weights have to be updated for any parameter change in the plant. In many cases, only linear parameters are changed, e.g., payload change of robot manipulators; then relearning all the weights is unnecessary.

In order to ameliorate the above drawbacks, we proposed a new structure of neural tracking controller: the two-time-scale neural tracking controller. This type of neural controllers consists of a slow subnet and a fast subnet. The weights in the slow subnet are updated in the slow time-scale according to the slow learning rules, while the weights in the fast subnet are updated in the fast time-scale according to fast learning rules. Moreover, only the fast weights need to be updated in order to adapt to the change of the linear parameters of the plant.

Based on this structure, several fast learning rules and slow learning strategies are developed. The stability properties of the closed loop system with the proposed two-time-scale neural tracking controller are analyzed. The results show that the closed loop system is uniformly bounded and the states of the plant converge to the region which depends only on the accuracy of the slow learning.

The rest of this paper is organized as the following sections: Section II describes the basic structure of the two-time-scale neural controller. Section III proposes fast learning rules for the fast subnet. Section IV analyzes the stability of the closed loop system with the two-time-scale neural controller based on the proposed fast learning rules. Section V introduces slow learning strategies, where a one-stage learning strategy and a two-stage learning strategy are proposed. In Section VI, some simulation results are presented to demonstrate the
feasibility of the two-time-scale neural tracking controller. Finally, conclusions with future work are presented in Section VII.

II. BASIC ARCHITECTURE OF THE TWO-TIME-SCALE NEURAL TRACKING CONTROLLER

We will consider a class of nonlinear dynamical systems which can be described by the following dynamical state equation:

$$\dot{X} = f(X) + g(X)u$$

(1)

where $X$ is the state vector of the system and can be measured directly, and $f(\cdot), g(\cdot)$ are unknown smooth nonlinear mappings.

The basic structure of the controller is a feedforward component plus a feedback component:

$$u = u_{ff} - K_e e$$

(2)

where $U_{ff}$ is the feedforward, $e = X - X_d$ is the tracking error with $X_d$ as the desired trajectory, and $K_e$ is the feedback gain. The desired feedforward component $u_{ff}$ is highly model dependent, where $u_{ff}$ can drive the plant so that the tracking error $e$ will converge to zero.

The error equation to describe the dynamical properties of the tracking error is as follows:

$$\dot{e} = f(e, X_d, \dot{X}_d, u_{ff}) + g(x)(u_{ff} - u_{ff}^d)$$

(3)

where $f, g$ are unknown nonlinear mappings. Note that the feedback component is included in $f$. Based on the definition of the $u_{ff}$, we see that if $u_{ff} = u_{ff}^d$, the error system

$$\dot{e} = f(e, X_d, \dot{X}_d, u_{ff}^d)$$

(4)

will be asymptotically stable around the equilibrium $e = 0$.

A special case for the nonlinear system described above is a rigid manipulator with $n$ degrees of freedom. The rigid manipulator can be described by the following dynamical equation:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + k(\theta) = u$$

(5)

where $\theta \in \mathbb{R}^n$ is the joint position vector, $\dot{\theta} \in \mathbb{R}^n$ is the joint velocity vector, $u \in \mathbb{R}^n$ is the input torque vector, $M(\theta) \in \mathbb{R}^{n \times n}$ is the mass-inertia matrix, $C(\theta, \dot{\theta}) \in \mathbb{R}^{n \times n}$ corresponds to the centrifugal and coriolis force, and $k(\theta) \in \mathbb{R}^n$ is the gravity force. It should be pointed out that $M(\theta), C(\theta, \dot{\theta}), k(\theta)$ are assumed unknown. Let $e = [\theta - \theta_d, \dot{\theta} - \dot{\theta}_d]^T$, then we can obtain the error equation as (3).

Most of the proposed desired feedforward components in (3) have the same form as the inverse dynamics of the plant:

$$u_{ff}^d(Z) = M(z_1)z_4 + C(z_1, z_2)z_3 + k(z_1)$$

(6)

where $Z = \{z_1, z_2, z_3, z_4\}$ is chosen from $(X, X_d, \dot{X}_d)$. Examples are as follows.

1) In [11], $z_1 = \theta, z_2 = \dot{\theta}, z_3 = \ddot{\theta}, z_4 = \dddot{\theta}$, and $z_4 = \dddot{\theta} - \Delta \dddot{\theta}$.

2) In [12], $z_1 = \theta_d, z_2 = \dot{\theta}_d, z_3 = \ddot{\theta}_d,$ and $z_4 = \dddot{\theta}_d$.

Fig. 1. Closed loop system with two-time-scale neural controller.

One of the following feedforward in [12] depends only on the desired trajectory, and we shall use it in the proposed controller. Therefore, we consider

$$u_{ff}^d(Z) = M(z_1^d)z_4^d + C(z_1^d, z_2^d)z_3^d + k(z_1^d)$$

(7)

where $Z^d = \{z_1^d, z_2^d, z_3^d, z_4^d\}$ is chosen from $(X_d, \dot{X}_d)$.

Since it is difficult to obtain this inverse dynamics accurately, it is natural to apply neural networks to learn the feedforward component (7). As seen in (7), the feedforward component is a static nonlinear mapping from the desired trajectory set to the feedforward set, and therefore it can be represented as a linear combination of some basis functions. Based on this fact, we propose a new type of neural controller: the two-time-scale neural controller.

The proposed two-time-scale neural controller consists a slow learning subnet cascaded to a fast learning subnet. The output of the slow learning subnet is $\dot{\theta}$, which is fed to the output of the fast learning subnet. The slow learning subnet is a common multilayer feedforward neural network with either sigmoidal neural functions or radial basis functions (RBF) [13]. The weight of the slow learning subnet is called the slow weight and denoted by $\dot{W}_s \in \mathbb{R}^{N_s}$, is the number of the branches in the slow subnet. The learning rule for updating the slow weight is called the slow learning rule. The objective for the slow learning subnet is to learn the basis functions corresponding to the structural information of the feedforward in the slow-time scale. The fast learning subnet consists of only one layer of neurons with linear neuron functions. The weight of the fast learning subnet is called the fast weight and denoted as $\dot{W}_f \in \mathbb{R}^{N_f}$, where $N_f$ is the number of branches in the fast subnet. The learning rule for updating the fast weight is the fast learning rule. The fast learning subnet is used to learn the parameters in the linear combination of learned nonlinear mapping in the fast-time scale, these parameters are corresponding to the parameter information of the feedforward component. The block diagram of the closed loop system with the proposed two-time-scale neural controller is shown in Fig. 1.

III. FAST LEARNING RULES

The first nominal fast learning rule is given in the following theorem.

Theorem 1: Suppose there exists a $C^1$ positive definite function $V_i(t, e)$, a positive definite matrix $Q$, a matrix $R$,
and a positive constant $\alpha$ such that
\[
\begin{align*}
V_1(t,e) &\leq \alpha ||e||^2 \\
\dot{V}_1(t,e) &\leq -e^T Q e + e^T R (u_{ff} - u_{ff}^d)
\end{align*}
\] (8)
where $V_1(t,e)$ denotes the derivative of $V_1$ along the trajectories of (3). Let $H(Z_d)$ be a basis function vector of $u_{ff}$ in the sense that there exists a constant vector $W_f$ such that
\[
u_{ff}^d = H(Z_d) W_f.
\] (9)
Assume that $Z_d$ is uniformly bounded in $t$. Let the feedforward control be
\[
u_{ff} = \tilde{H}(W_f^*, Z_d) W_f
\] (10)
where $W_f^*$ is such that $R(H(Z_d)) \subset R(\tilde{H}(W_f^*, Z_d))$ (R(·) denotes the range space). Then, $e(t) \to 0$ as $t \to \infty$ and $\tilde{W}_f \in L_{\infty}$ if the following fast learning rule is applied:
\[
\dot{\tilde{W}}_f = -\Gamma \tilde{H}(W_f^*, Z_d) R^T e
\] (11)
where $\Gamma$ is a chosen positive definite matrix.

\textbf{Proof:} For simplicity, we denote $\tilde{H}(W_f^*, Z_d)$ as $\tilde{H}(W_f^*)$ and $H(Z_d)$ as $H$, respectively. By assumption, $R(\tilde{H}) \subset R(\tilde{H}(W_f^*))$. Therefore, there exists a constant vector $W_f^d$ such that $HW_f = \tilde{H}(W_f^*) W_f^d$.

Then, we have
\[
u_{ff} = u_{ff}^d + \tilde{H}(W_f^*) (\tilde{W}_f - W_f^d)
\]
(12)
where $\Phi = \tilde{W}_f - W_f^d$ is the fast weight error.

Choose the following Lyapunov function candidate:
\[V = V_1 + \frac{1}{2} \Phi^T \Gamma^{-1} \Phi\] (13)
where $V_1$ is a Lyapunov function that satisfies the condition (8). We can obtain the derivative of $V$ as follows:
\[\dot{V} = \dot{V}_1 + \Phi^T \Gamma^{-1} \Phi \leq -e^T Q e + e^T R \tilde{H}(W_f^*) \Phi + \Phi^T \Gamma^{-1} \Phi.
\] (14)
Substituting (11) into (14) and noting that $\dot{\tilde{W}}_f = \tilde{\Phi}$ since $W_f^d$ is a constant vector, we have
\[\dot{V} \leq -e^T Q e.
\] (15)
Therefore, $V$ is positive definite and $\dot{V}$ negative semidefinite.

Since $Q$ is positive definite, there is a positive constant $\lambda$ such that $e^T Q e \geq \lambda ||e||^2 = W(t)$. Then we have
\[
\int_0^t W(s) \, ds \leq V(0) - V(t)
\] (16)
where $V(t)$ is nonincreasing and bounded below. It follows that
\[
\lim_{t \to \infty} \int_0^t W(s) \, ds \leq \infty.
\] (17)
By the lemma due originally to Barbalat, it follows $e \to 0$, $\tilde{W}_f \in L_{\infty}$ follows from the fact that $V(t)$ is bounded.

Theorem 1 states that the tracking error will converge to zero and the fast weight will be bounded by implementing the fast learning rule (11) if the desired slow weights $W_f^*$ can be found such that the output of the slow network is the basis of the desired feedforward component. However, such a desired slow weight vector $W_f^*$ is generally unknown. Instead, we usually have $||H(Z_d) W_f - H(W_s, Z_d) W_f^d|| > 0$, where $W_f^d$ is a fast weight vector such that $||H(Z_d) W_f - H(W_s, Z_d) W_f^d||_{Z_d \in \Omega_{Z_d}}$ is minimized over a compact set $\Omega_{Z_d}$ that contains all possible desired trajectories. Denote $\delta H(W_s, Z_d) = H(Z_d) W_f - H(W_s, Z_d) W_f^d \neq 0$. The goal of the slow learning is to choose the slow weight $\tilde{W}_s$ such that the norm of $||\delta H(W_s, Z_d)||$ over $\Omega_{Z_d}$ attains its minimum.

Theorem 1 states that $\tilde{W}_f \in L_{\infty}$. But if the assumption $R(H(Z_d)) \subset R(\tilde{H}(W_f^*, Z_d))$ is not satisfied, $e$ may not converge to zero. Consequently, $\tilde{W}_f$ may not be bounded. In order to solve this problem, we borrow the idea from the robust adaptive control [14] and modify the fast learning rule as follows:
\[
\dot{\tilde{W}}_f = -\Gamma \tilde{H}(W_f^*, Z_d) R^T e - \gamma \tilde{W}_f
\] (18)
where
\[
\gamma = \begin{cases} 
0 & ||\tilde{W}_f|| \geq \tilde{W}_{f_0} \\
2 & \text{otherwise.}
\end{cases}
\] (19)
$\gamma \tilde{W}_f$ in (18) is called the leakage term. If $\gamma$ satisfies the condition (19), $\gamma \tilde{W}_f$ is called the switching leakage term. The choice of $\tilde{W}_{f_0}$ will be described

IV. STABILITY ANALYSIS

In order to analyze the stability properties of the closed loop system, we need the following lemma [14].

\textbf{Lemma 1:} Let $\Phi = \tilde{W}_f - W_f^d$. Suppose that $\gamma$ satisfies the switching condition (19), then
\[
-\gamma \Phi^T \tilde{W}_f \leq -\frac{\gamma}{2} ||\Phi||^2 + \frac{\gamma}{2} \left(\tilde{W}_{f_0} + ||W_f^d||\right)^2
\] (20)
furthermore, if $\tilde{W}_{f_0} \geq ||W_f^d||$, then
\[
-\gamma \Phi^T \tilde{W}_f \geq 0
\] (21)
where $\gamma$ and $\gamma_0$ are.

The stability properties of the closed loop system with the two-time-scale neural controller are given by the following theorems.

\textbf{Theorem 2:} Suppose that the control law (2) and the fast learning rule (18) are applied, then all solutions $\Phi, e$ of (3) which start from any compact set with sufficiently small initial tracking error in $(\Phi, e)$ are uniformly bounded and converges to the region:
\[
\Omega_r = \left\{e, \Phi : V(e, \Phi) \leq \frac{1}{\rho_0} \frac{\lambda_0}{2} \left(\tilde{W}_{f_0} + ||W_f^d||\right)^2 + \frac{\eta^2}{2\kappa} d_0 \right\}
\] (22)
where $V$ is defined in (13); and the tracking error will be bounded and converge to the region:

$$
\Omega_0 = \left\{ e : \lim_{T \to \infty} \sup_{T > 0} \frac{1}{T} \int_{t_0}^{t_0 + T} |e(t)|^2 \, dt \leq \frac{1}{\rho_0} \left( \frac{\eta^2}{2} d_0^2 + \frac{\lambda_0}{2} (\bar{W}_{f_0} + ||W_f||^2) \right), \quad \forall t_0 \geq 0 \right\} \quad (23)
$$

where $\rho_0, \lambda_0, \eta, \xi$ are positive constants. $d_0^2(\bar{W}_s) = \sup_{Z_s \in \Omega_{Z_s}} (||\delta H(\bar{W}_s, Z_s)||^2)$ is a measure of the accuracy of the slow learning rule.

**Remark:**
- This theorem guarantees that the closed loop system is locally Lagrange stable, i.e., if the tracking error starts sufficiently closely to the origin, all states of the closed loop system will be uniformly bounded. The local nature of the result is a consequence of the assumed local stability of the ideal error system (4).
- The uniform bound depends not only on $d_0^2$ but also on $\bar{W}_{f_0}$ and $||W_f||$. This means that Theorem 2 cannot guarantee that the tracking error will be arbitrarily small even though $d_0^2$ is. But this problem can be solved by choosing sufficiently large threshold $\bar{W}_{f_0}$ in the switching leakage condition, which will be stated in Theorem 3.

**Proof:** Choose the following Lyapunov function candidate:

$$
V = V_1 + \frac{1}{\rho} \Phi^T \Gamma^{-1} \Phi
$$

where $V_1$ is any Lyapunov function that satisfies the condition in (8).

Applying the fast learning rule (18), we can obtain the derivative of $V$ as follows:

$$
\dot{V} = \dot{V}_1 + \Phi^T \Gamma^{-1} \Phi
$$

$$
\leq -\varepsilon e^T \gamma e + \varepsilon^2 R(u - u_0) + \Phi^T \Gamma^{-1} \Phi
$$

$$
\leq -\varepsilon e^T \gamma e + \varepsilon^2 R H(\bar{W}_s) \bar{W}_f - H(\bar{W}_s) W_f^T
$$

$$
+ H(\bar{W}_s) W_f^T - H W_f + \Phi^T \Gamma^{-1} \Phi
$$

$$
\leq -\varepsilon e^T \gamma e + \varepsilon^2 R \delta H(\bar{W}_s, Z_s) + e^T \delta R H(\bar{W}_s) \Phi
$$

$$
+ \Phi^T \Gamma^{-1} \Phi
$$

$$
\leq -\varepsilon e^T \gamma e + \varepsilon^2 R \delta H(\bar{W}_s, Z_s) - \gamma \Phi^T \Gamma^{-1} \Phi_f. \quad (25)
$$

Let $\xi = \sigma_{\min}(Q), \eta = ||R||, \Gamma^{-1} = \gamma I, \lambda = \gamma \gamma$, and $\gamma \gamma$, then apply Lemma 1; we have:

$$
\dot{V} \leq -\xi ||e||^2 + \eta ||e|| ||\delta H(\bar{W}_s, Z_s)|| - \lambda \Phi^T \Phi_f
$$

$$
\leq -\xi ||e||^2 - \frac{\lambda_0}{2} + \eta ||e|| ||\delta H(\bar{W}_s, Z_s)|| + \frac{\lambda_0}{2} (\bar{W}_{f_0} + ||W_f||^2). \quad (26)
$$

Note $\alpha b \leq (a^2 + b^2 \gamma^2 / 2)$ and let $a = \sqrt{\xi} ||e||$ and $b = \eta ||\delta H(\bar{W}_s, Z_s)|| / \sqrt{\xi}$; then we have

$$
\dot{V} \leq -\frac{\xi}{2} ||e||^2 - \frac{\lambda_0}{2} ||\Phi||^2 + \frac{\eta^2}{2} \xi ||\delta H(\bar{W}_s, Z_s)||^2 + \frac{\lambda_0}{2} (\bar{W}_{f_0} + ||W_f||^2)^2
$$

$$
\leq -\rho ||e||^2 + ||\Phi||^2 + \frac{\eta^2}{2} \xi ||\delta H(\bar{W}_s, Z_s)||^2 + \frac{\lambda_0}{2} (\bar{W}_{f_0} + ||W_f||^2)^2. \quad (27)
$$

where $\rho = \min(\xi, \lambda_0)/2$. Since $V = V_1 + 1/2 \Phi^T \Gamma^{-1} \Phi$ and $V_1$ satisfies the condition in Theorem 1. There exists a positive number $\alpha = \max(\alpha_1, \sigma_{\max}(\Gamma^{-1} / 2))$ such that $V \leq \alpha (||e||^2 + ||\Phi||^2)$. Therefore we have

$$
\dot{V} \leq -\rho_0 V + \frac{\eta^2}{2} \xi ||\delta H(\bar{W}_s, Z_s)||^2 + \frac{\lambda_0}{2} (\bar{W}_{f_0} + ||W_f||^2)^2. \quad (28)
$$

where $\rho_0 = \rho / \alpha$. Since $\delta H(\bar{W}_s, Z_s)$ is a bounded function over the compact set $\Omega_{Z_s}$, we can define

$$
d_0^2(\bar{W}_s) = \sup_{Z_s \in \Omega_{Z_s}} (||\delta H(\bar{W}_s, Z_s)||^2). \quad (29)
$$

Then, we have

$$
\dot{V} \leq -\rho_0 V + K_0
$$

$$
(30)
$$

where $K_0 = (\eta^2 / 2 \xi) d_0^2 + (\lambda_0 / 2) (\bar{W}_{f_0} + ||W_f||^2)^2$ is a constant.

Integrating (30), we then have

$$
V(t) \leq \left( V_0 - \frac{K_0}{\rho_0} \right) e^{-\rho_0 t} + \frac{K_0}{\rho_0}. \quad (31)
$$

Then, we have the result (22). Since $\dot{V} \leq -\rho_0 ||e||^2 + K_0$, we have

$$
\rho_0 \int_{t_0}^{t_0 + T} ||e(t)||^2 \, dt \leq \frac{V(t_0) - V(t_0 + T) + K_0}{T}. \quad (32)
$$

Noting that $V(t)$ is bounded for all $t \geq 0$, then the second result follows by letting $T \to \infty$.

From this theorem, we see that the ultimate bound depends not only on $d_0^2$ but also on $(\bar{W}_{f_0} + ||W_f||^2)^2$. Therefore, this theorem cannot guarantee that the tracking error is arbitrarily small even though $d_0^2$ is. However, we can solve this problem by choosing sufficiently large threshold $\bar{W}_{f_0}$ in (19), which will be described in the following theorem.

**Theorem 3:** If $\bar{W}_{f_0} \geq ||W_f||^2$ in (19), then in addition to the results in Theorem 2, the tracking error will converge to the region that depends only on $d_0^2$:

$$
\Omega_s = \left\{ e : \lim_{T \to \infty} \sup_{T > 0} \frac{1}{T} \int_{t_0}^{t_0 + T} ||e(t)||^2 \, dt \leq \frac{\eta^2}{2 \xi \rho_2} d_0^2, \quad \forall t_0 \geq 0 \right\}. \quad (33)
$$
Proof: Choosing the same Lyapunov function as in Theorem 2 and applying the fast learning rule (18), we can obtain the derivative of the chosen Lyapunov function as follows:

\[
\dot{V} \leq -e^T Q e - e^T R \dot{H}(\hat{W}_s, Z_d) + e^T \Phi \dot{\Phi} + \Phi^T \Gamma^{-1} \Phi \\
\leq -e^T Q e - e^T R \delta H(\hat{W}_s, Z_d) - \gamma \Phi^T \Gamma^{-1} \Phi_f.
\]

(34)

Since \( \hat{W}_{f0} \geq \|W_f\| \), then we have \( \gamma \hat{W}_f^T \Gamma \Phi \geq 0 \) by Lemma 1 and therefore

\[
\dot{V} \leq -e^T Q e - e^T R \delta H(\hat{W}_s, Z_d) \\
\leq -\xi \|e\|^2 + \eta \|e\| \|\delta H(\hat{W}_s, Z_d)\|
\]

(35)

where \( \xi = \sigma_{\min}(Q), \eta = |R| \). Note \( ab \leq (a^2 + b^2/2) \) and let \( a = \sqrt{\xi} \|e\| \) and \( b = \eta \|\delta H(\hat{Z}_s, \hat{W}_s)\|/\sqrt{\xi} \); then we have

\[
\dot{V} \leq -\frac{\xi}{2} \|e\|^2 + \frac{\eta^2}{2\xi} \|\delta H(\hat{W}_s, Z_d)\|^2 \\
\leq -p_1 \|e\|^2 + K_1
\]

(36)

where \( K_1 = (\eta^2/2\xi) d^2 \). By the same reasoning as in the proof of Theorem 2, we have

\[
\frac{p_1}{T} \int_{t_0}^{t_0+T} \|e(t)\|^2 dt \leq \frac{V(t_0) - V(t_0 + T)}{2} + K_1.
\]

(37)

Since \( V(t) \) is bounded for all \( t \geq 0 \), then the theorem follows by letting \( T \to \infty \).

\[ \square \]

V. SLOW LEARNING STRATEGIES

A. One-Stage Learning Strategy

As seen in the stability analysis, the tracking error will converge to the region that is bounded by the approximation error \( \|\delta H(\hat{W}_s, Z_d)\| \) of slow subnet. The slow subnet is a feedforward neural net whose approximation error depends on the slow weights when its structure is specified. The slow weights cannot be updated on the well known backpropagation learning rule since no teacher is available. However, we still can update the slow weight indirectly based on the available tracking error. The basic principle is as follows. After updating the fast weights on-line for a sufficiently long period, if the tracking error is still not within a satisfactory level, it means that the tracking error cannot be decreased by only updating the fast weights alone. Then, the slow weights should be updated to further reduce the tracking error. We call this approach the one-stage learning strategy.

In this approach, the fast net updates its weights according to the fast learning rule on a fast-time scale. After each fast learning period is finished, the norm of the tracking error is checked. If this norm is not within the tolerance, which means that this norm cannot be minimized by only updating the fast weights, the slow weights are then updated to reduce this norm.

We choose a quadratic norm of the tracking error for the slow subnet. Therefore, the slow weight vector \( \hat{W}_s \) is updated to minimize the following objective function:

\[
J(\hat{W}_s) = \min_{\hat{W}_s \in \mathbb{R}^{n_s}} \{ J(\hat{W}_s) \} = \min_{\hat{W}_s \in \mathbb{R}^{n_s}} \int_{t_0}^{t_0+T} e^T Q e \, dt
\]

(38)

where \( Q \) is a weighting matrix.

To avoid the difficulty of computing the gradient \( \partial J/\partial \hat{W}_s \) and the local minimum problem associated with gradient-based learning rules, we propose to use stochastic optimization methods for the slow time-scale learning.

Some of the candidates for the slow learning rules in one-stage learning are: classical simulated annealing (CSA) [15], stochastic approximation with function smoothing (SAFS) [16], and fast simulated annealing (FSA) [17].

In the fast time-scale, the slow weight vector \( \hat{W}_s \) is a constant vector. Only the fast weight vector \( W_f \) is updated according to the fast learning rule. During fast learning, the sample points of the tracking error are collected for the slow learning. After the fast learning is finished at each learning cycle, the slow weight vector \( \hat{W}_s \) is updated according to the slow learning rule.

The algorithm for one-stage learning strategy is summarized as follows.

For \( k = 1, 2, 3, \ldots, N \) do the following:

1) Apply the control law (2) and fast learning rule (18) to drive the plant to track the desired trajectory \( X_d \) in the interval \( [t_k, t_k + T] \), and the sample points of the tracking error are collected.

2) Compute \( J(\hat{W}_s) \) in (38). Is \( J(\hat{W}_s) < \) the desired accuracy? If yes, stop. If not, continue to the next step.

3) Update \( \hat{W}_s \) based on a slow learning rule to reduce \( J(\hat{W}_s) \).

B. Two-Stage Learning Strategy

Although the one-stage learning strategy directly updates slow weights in one stage, its convergence rate is typically very slow. The reason is that some type of random search is used as there is no teacher available. An alternate approach is to obtain the desired feedforward commands corresponding to a desire output trajectory in the first stage, then this desired-trajectory/feedback-forward-command pair is encoded in the neural network in the second stage. This leads to the two-stage learning strategy.

In the first-stage learning, we simply choose the pulse basis. In other words,

\[
u_{df}(t) = \hat{W}_f(t).
\]

(39)

Then the fast learning rule is

\[
\dot{\hat{W}}_f = -\Gamma R^T e - \gamma \hat{W}_f
\]

(40)

where

\[
\gamma = \begin{cases} 
\gamma_0 & \text{if } \|\hat{W}_f\| \geq \hat{W}_{f0} \\
0 & \text{otherwise.}
\end{cases}
\]

(41)

For the actual implementation, the desired trajectory is sampled so that \( \hat{W}_f \) is a finite vector for a finite desired trajectory.
Since the number of the basis function is the sample number of the desired trajectory, a large number of basis functions is required as the number of the desired trajectories increases. However, since the desired control commands have been obtained in the first stage, we can use a feedforward neural network or a recurrent neural network of much smaller size to memorize these commands. This is the second-stage learning.

In the second-stage learning, the desired control commands obtained in the first-stage learning are used as targets of the neural network with the desired trajectory as the input. Since the teacher for the neural network is available, back propagation or conjugate back propagation can be used as the learning rules. It should be pointed out that the second-stage learning is implemented off-line, which does not require the operation of the plant.

VI. SIMULATION

In order to demonstrate the feasibility of the proposed two-time-scale neural controller, a two link rigid planar manipulator is used for the simulation. All models and parameters for the simulation are shown in the Appendices. Both the one-stage learning and the two-stage learning are simulated. The results show that good tracking performance can be obtained in both cases. However, the convergence rate of the two-stage learning is much faster than that for the one-stage learning for a given desired trajectory. Moreover, the amount of actual operation of the plant is much less for the two-stage learning.

A. Simulation for One-Stage Learning

In the one-stage learning, CSA is chosen as the slow learning rule. We try to show that the proposed neural controller is feasible for the slow learning rule with the slowest convergence rate. Therefore, this neural controller is certainly feasible for other slow learning rules. The simulation results are shown in Figs. 2–9.

Fig. 2 shows the value of the objective function (38) with respect to the slow learning cycles. The slow learning period $T$ is 4 seconds. We see that the trend of the objective function (38) decreases as the slow learning cycles increases. However, the convergence rate is slow since CSA is used. After the slow learning is finished, the best slow weights are obtained. The tracking performance after the learning is shown in Fig. 3 for the first link and in Fig. 4 for the second link. From these two figures, we see that the tracking of the joint 1 position $X_1$ and joint 2 in position $X_2$ is very good, but the tracking of the velocities $X_3$ and $X_4$ needs to be improved. This can be solved by choosing suitable parameters in the weighting matrix $Q$ in (38).

To see the effect of the leakage term, we suppose that the slow learning has already converged, so only the effect of the fast subnet is investigated.

Fig. 7 shows position error $e_2$ for the first link when $\gamma = 50$ (solid line) and $\gamma = 0$ (dashed line). It is obvious that $\gamma$ affects the tracking accuracy when $\|\delta H\| = 0$. Indeed, the tracking performance for the learning rule without leakage term is better.
Next, we choose a bad basis function $H_0$ as the output of the slow learning subnet to demonstrate the effect of leakage term on the Lagrange stability of the closed loop system.

Fig. 8 shows states and fast weights of the closed loop system when $\gamma = 0$. It can be seen that one of the fast weights diverges. However, if we let the leakage term $\gamma = 5$, then all states and the fast weights are bounded (see Fig. 9). Therefore, the leakage term is necessary in the two-time-scale neural controller for the boundedness of the closed loop system.

B. Simulation for Two-Stage Learning Strategy

The simulation for two-stage consists of two steps: the first step obtains the teacher and the second step encodes the knowledge. The conjugate backpropagation learning rule is chosen as the slow learning rule in the second stage.

The simulation results for the first stage learning are shown in Figs. 10–14, where PD represents the proportional-derivative controller and NC represents a neural controller together with the PD controller. Fast learning rule (40) is used.

We can observe the following based on the simulation.

1) The tracking performance of the neural controller is much better than that when only a PD controller is used.

2) For a given desired trajectory, two-stage learning converges much faster than the one-stage learning (see Figs. 2 and 14). The teacher for the slow subnet can be obtained within a few iterations.

3) The choice of $\Gamma$ is crucial. An ill-advised choice of $\Gamma$ may lead to slow convergence rate or oscillation.

VII. CONCLUSION

A two-time-scale neural controller is proposed for the tracking control of a class of nonlinear dynamical systems.
\[ M(\theta) = \begin{pmatrix} 2l_1 \cos \theta_2 + l_2 \cos \theta_2 + l_1^2 (m_1 + m_2) & l_1^2 m_2 + l_1 l_2 \cos \theta_2 m_2 \\ l_2^2 m_2 + l_1 l_2 \cos \theta_2 m_2 & l_2^2 m_2 \end{pmatrix} \]
\[ C(\theta, \dot{\theta}) = \begin{pmatrix} -2l_1 l_2 m_2 \sin \theta_2 \dot{\theta}_2 & -l_1 l_2 m_2 \sin \theta_2 \dot{\theta}_2 \\ l_1 l_2 m_2 \sin \theta_2 \dot{\theta}_2 & 0 \end{pmatrix} \]
\[ K(\theta) = \begin{pmatrix} g(m_2 l_2 \cos(\theta_1 + \theta_2) + (m_1 + m_2) l_1 \cos \theta_1) \\ m_2 l_2 g \cos(\theta_1 + \theta_2) \end{pmatrix} \]

Fig. 9. State for \( \delta H \neq 0 \), with leakage.

Fig. 10. Position tracking for the first link.

Fig. 11. Position tracking for the second link.

Fig. 12. Velocity tracking for the first link.

The closed loop system with the two-time-scale neural tracking controller is Lagrange stable if the leakage term is used in the fast learning rules. Furthermore, the tracking error will only depend on the accuracy of the slow learning if the fast learning rule with sufficiently large switching leakage term is used. The simulation results of a two-link rigid arm demonstrate the feasibility of the proposed two-time-scale neural controller.
Our future work will focus on the efficient training of the slow subnet in the two-stage learning and the generalization issue of neural networks.

APPENDIX A:

IX. DYNAMIC MODEL FOR A TWO-LINK ARM
The dynamics of the two-link rigid arm can be described by

\[ M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + K(\theta) = \tau \]  \hspace{1cm} (42)

where \( M(\theta), C(\theta, \dot{\theta}), \) and \( K(\theta) \) are as follows (see top of page 998).

The parameters of the arm are as follows: \( l_1 = 0.7 \text{ M}; l_2 = 0.5 \text{ M}; m_1 = 10 \text{ kg}; m_2 = 5 \text{ kg}, g = 9.8 \text{ M/s}^2. \)

APPENDIX B:

XI. PARAMETERS FOR THE SIMULATION
The parameters for the simulation of the one-stage learning are as follows.

- **Size of the Slow Learning Net**: The number of the neurons per hidden layer is 8 and there are two hidden layers. The number of inputs and outputs of is 8, respectively. All neuron functions are symmetric sigmoidal functions.

- **Parameters of CSA**: For the schedule of CSA, we use \( T_e = r_1 / \log(1 + \sigma e) \), where \( r_1 \) and \( \sigma \) are chosen as 25 and 0.01, respectively.
Fig. 17. Position tracking for the first link.

Fig. 18. Position tracking for the second link.

- **Parameter for the fast learning rule (18):** we chose \( \Gamma = \text{diag}(0.2, 0.2), R = \text{diag}(1.5, 1), \gamma_0 = 2, \bar{w}_{f0} = 50 \), respectively.

The parameters for the simulation of the two-stage learning are: \( \Gamma = \text{diag}(400, 100), \gamma_0 = 0.01 \), and \( \bar{w}_{f0} = 5000 \).

REFERENCES


Weiyieng Cheng was born in Wuhan, China. He received the B.S.E.E. and M.S.E.E. degrees in hydraulic and electric engineering from Wuhan University in 1982 and 1985, respectively.

He is currently pursuing the Ph.D. degree in electrical engineering at Rensselaer Polytechnic Institute, Troy, NY. His research interests include neural network theory and its applications to control, signal processing, and power systems.

John Ting-Yung Wen (SM’93) received the B.Eng. degree from McGill University in 1979, the M.S. degree from the University of Illinois in 1981, and the Ph.D. degree from Rensselaer Polytechnic Institute in 1985, all in electrical engineering.

From 1981 to 1983, he was a System Engineer with Fisher Control Company, Marshalltown, IA, where he worked on the coordination control of a pulp and paper plant. From 1985 to 1988, he was a member of Technical Staff at the Jet Propulsion Laboratory, Pasadena, CA, where he worked on the modeling and control of large flexible structures and multiple-arm control. Since 1988, he has been with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, where he is currently an Associate Professor. He is also a member of the New York Center for Advanced Technology in Robotics and Automation. His current research interests are in the area of modeling and control of multibody mechanical systems, including flexible structures, robots, and vehicles.

Dr. Wen is a member of SIAM and AIAA.