Neural Networks, Vol. 1, pp. 251–265, 1988
Printed in the USA. All rights reserved.

0893-6080/88 $3.00 + .00

ORIGINAL CONTRIBUTION

Feedback-Error-Learning Neural Network for Trajectory Control of a Robotic Manipulator

HIROYUKI MIYAMOTO,* MITSUO KAWATO,† TOHRU SETOYAMA, AND RYOJI SUZUKI‡

Faculty of Engineering Science, Osaka University

(*Received and accepted February* 1988)

Abstract—*A neural network model for generation of motor command was proposed in our earlier paper. This model contains a feedback loop (transcortical loop), an internal neural model of motor system (spinocerebellum and magnocellular part of the red nucleus) and an internal neural model of inverse dynamics of the motor system (cerebrocerebellum and parvocellular part of the red nucleus). The inverse-dynamics model is acquired by heterosynaptic plasticity using feedback motor command (torque) as an error signal. In this paper, we apply hierarchical arrangement of the transcortical loop and the inverse-dynamics model for learning trajectory control of an industrial robotic manipulator. Although neither strict modeling of the manipulator nor precise parameter estimation was required, the control performance by the neural-network model improved gradually during 30 minutes of learning. Once the neural-network model learned to control some movement, it could control quite different and faster movements. That is, the neural-network model has capability to generalize learned movements. Advantages of the neural-network control over conventional control methods are discussed.*

Keywords—Motor control, Inverse-dynamics model, Feedback-error-learning.

1. INTRODUCTION

It has long been believed that nerve connections in the brain are stable and rigid after their formation at an early developmental stage. This rigidity of the nerve connections probably provides an important basis for instinctive behaviors such as reflexes or autonomous movements as the built-in generator of fixed action pattern. However, it cannot readily account for any kinds of adaptive behaviors, such as learning and memory. In this context, it has recently been recognized that some of the nerve connections of the brain have plasticity. The synaptic plasticity must play the most important roles in information processing for voluntary movements, since most of skilled voluntary movements are learned movements.

Marr (1982) pointed out that an information processing device (brain) must be understood at the following different levels before one can be said to have understood it completely: (a) computational theory; (b) representation and algorithm; and (c) hardware implementation. We proposed a computational model of voluntary movement in Figure 1 (Kawato, 1988; Kawato, Furukawa, & Suzuki, 1987; Kawato, Uno, Isobe, & Suzuki, 1988), which accounts for Marr's first level. Several lines of experimental evidence suggest that the information in Figure 1 is internally represented in the brain. First, Flash and Hogan (1985) provided strong evidence to indicate that movement is planned at the task-oriented coordinates (visual coordinates) rather than at the joint or muscle level. Second, the presence of the transcortical loop (i.e., the negative feedback loop via the cerebral cortex [Evarts, 1981]) indicates that the desired trajectory must be represented also in body coordinates, since signals from proprioceptors are expressed in body coordinates. Finally, Cheney and Fetz (1980) showed that discharge frequencies of primate corticomotoneuronal cells in the motor cortex were fairly proportional to active forces (torque). Consequently, the central nervous system (CNS) must adopt, at least partly, the step-by-step computation strategy for the control of voluntary movement.

Several works have been done regarding the step-by-step information processing shown by the three

* Present address: Sumitomo electric industries, Ltd., Osaka 554, Japan.

† Present address: ATR Auditory and Visual Perception Research Laboratories, Osaka 540, Japan.

‡ Present address: Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, Tokyo, 113 Japan.

Requests for reprints should be sent to Mitsuo Kawato, Cognitive Processes Department, ATR Auditory and Visual Perception Research Laboratories, Twin 21 Bldg., Higashi-ku, Osaka 540, Japan.

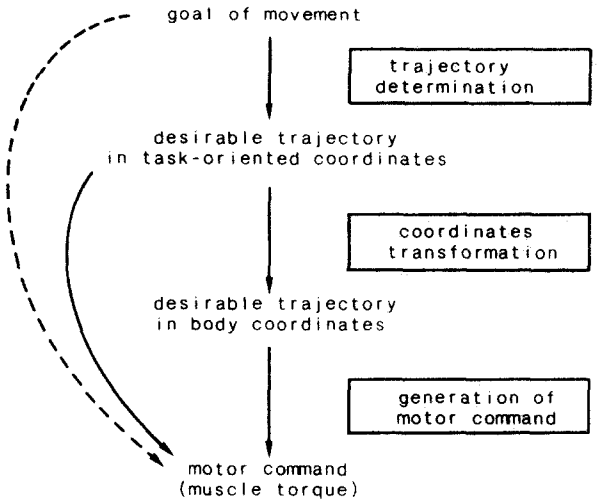

FIGURE 1. A computational model for voluntary movement.

straight arrows in Figure 1. For example, Flash and Hogan (1985) proposed a minimum-jerk model which explains computation of a desired trajectory in the task-oriented coordinates based on some performance index (goal of movement). Grossberg and Kuperstein (1986), Rumelhart (in press), and Psaltis, Sideris, and Yamamura (1987) proposed neural networks which execute coordinates transformation. However, we do not adhere to the hypothesis of the step-by-step information processing. Rather, Uno, Kawato, and Suzuki (1987) proposed a learning algorithm which calculates the motor command directly from a goal of the movement represented as minimization of the sum of the square of the rate of change of torque integrated over the entire movement (broken and curved arrow in Figure 1). Further, as shown by the solid curved arrow in Figure 1, we showed that motor command can be obtained directly from the desired trajectory represented in the task-oriented coordinates by an iterative learning algorithm. In this respect, our model differs from the three-level hierarchical movement plan proposed by Hollerbach (1982). However, it is still certain that the CNS must solve the problem of generation of motor command for a large class of voluntary movements, as evident from experimental results cited above.

Let us consider the problem of motor control in a computational framework. There exists causal relation between the motor command and the resulting movement pattern. Let $T(t)$ denote time history of the motor command (torque) and $\theta(t)$ denote time history of the movement trajectory in a finite time interval $t \in [0, t_f]$. The causal relation between T and θ can be written as $G(T(\cdot)) = \theta(\cdot)$ using a functional G. If a desired movement pattern is denoted by $\theta_d(t)$, the movement error is defined as $\theta_d - G(T) = \theta_d - \theta$. The problem to generate a motor command T_d, which realizes the desired movement pattern θ_d, is equivalent to find an inverse of the functional G. Once the inverse of G is acquired, the required motor command can be calculated as $T_d = G^{-1}(\theta_d)$ since $\theta = G(T_d) = G[G^{-1}(\theta_d)] = \theta_d$ holds.

Because controlled objects in motor control have many degrees of freedom and nonlinear dynamics, the functional G is also nonlinear. Furthermore, since the exact dynamics of the controlled objects (arms, limbs, robotic manipulators) are generally unknown, we do not know exact form of G. Consequently, it is practically very difficult to calculate the inverse of G. We proposed a neural-network model and the feedback-error-learning rule, by which an internal neural model of inverse dynamics of the motor system (i.e., G^{-1}) is acquired during execution of movement (Kawato, 1988; Kawato et al., 1987; Kawato, Uno et al., 1988). In this neural-network model, real-time learning with heterosynaptic plasticity takes place in the cerebellum and the red nucleus with using the feedback motor command as the error signal. In this paper, we apply the neural network model for learning control of an industrial robotic manipulator.

2. A HIERARCHICAL NEURAL-NETWORK MODEL

In learning a movement, we first execute the movement very slowly because it cannot be adequately preprogramed. With practice, a greater amount of the movement can be preprogramed and the movement can be executed more rapidly. Ito (1970) proposed that the cerebrocerebellar communication loop is used as a reference model for the open-loop control of voluntary movement. Allen and Tsukahara (1974) proposed a comprehensive model, which accounts for the functional roles of several brain regions. Tsukahara and Kawato (1982) proposed a theoretical model of the cerebro-cerebello-rubral learning system based on recent experimental findings of the synaptic plasticity. Expanding on these previous models, we proposed a neural-network model for the control of and learning of voluntary movement, shown in Figure 2 (Kawato, 1988; Kawato et al., 1987; Kawato, Isobe, Maeda, & Suzuki, in press; Kawato, Uno et al., 1988).

In our model, the association cortex sends the desired motor pattern θ_d expressed in body coordinates, to the motor cortex, where the motor command, that is torque T to be generated by muscles, is then somehow computed. The actual motor pattern θ is measured by proprioceptors and sent back to the motor cortex via the transcortical loop. Then, feedback control can be performed utilizing error in the movement trajectory $\theta_d - \theta$. However, feedback delays and small gains both limit controllable speeds of motions.

The spinocerebellum-magnocellular part of the red nucleus system receives information about the results of the movement θ as afferent input from the proprioceptors, as well as an efference copy of the motor command T. As shown in Figure 2, axon collateral of the

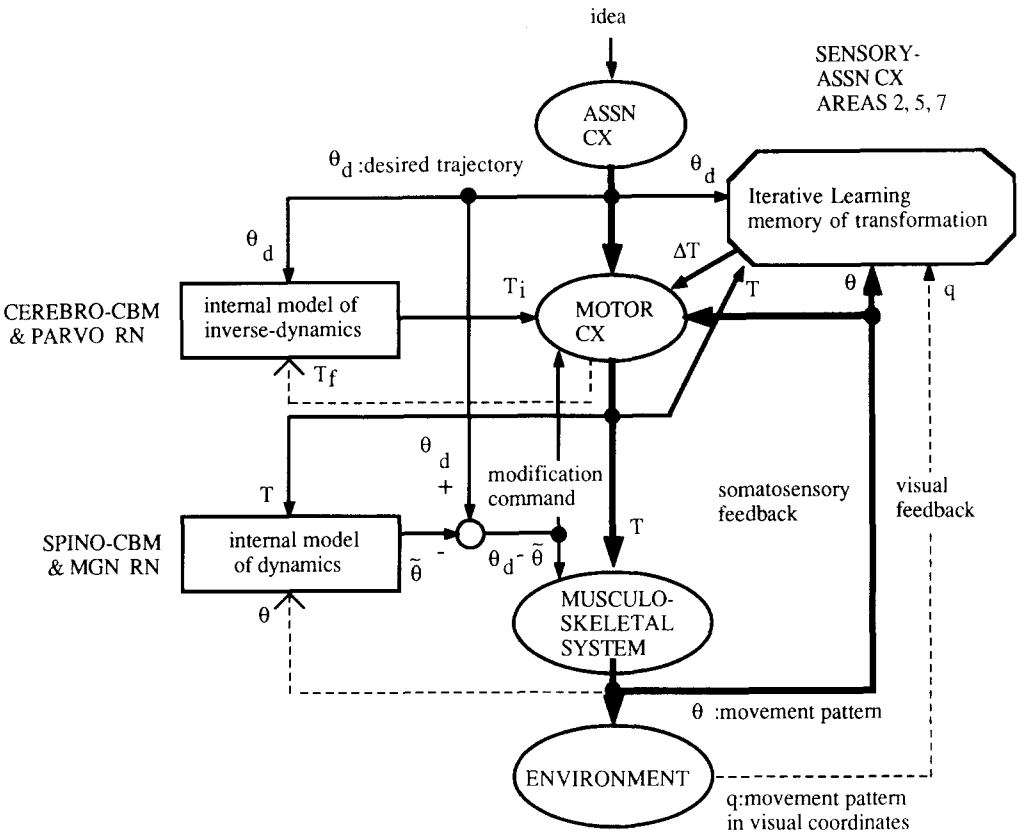

FIGURE 2. A hierarchical neural network model for control and learning of voluntary movement. The model is composed of the following four parts: (a) the main descending pathway and the transcortical loop designated by heavy lines, (b) the spinocerebellum-magnocellular red nucleus system (SPINO-CBM & MGN RN) as an internal neural model of dynamics of the musculoskeletal system, (c) the cerebrocerebellum-parvocellular red nucleus system (CEREBRO-CBM & PARVO RN) as an internal neural model of inverse-dynamics of the musculoskeletal system, (d) the sensory-association cortex (SENSORY-ASSN CX, AREAS 2, 5, 7) as an associative memory for iterative learning control (Kawato, Isobe *et al.*, in press). These subsystems of the central nervous system are phylogenically and ontogenetically older in this order. For the dynamics model and the inverse-dynamics model of the musculoskeletal system, the inputs used for calculating nonlinear input–output relation are designated by solid arrows and lines (T for dynamics model and θ_d for inverse-dynamics model) and the inputs for synaptic modification (i.e., teaching or error signals) are designated by dotted line and thin arrow (θ for dynamics model and T_f for inverse-dynamics model). ASSN CX = association cortex; MOTOR CX = motor cortex.

main descending pathway carries the information of the efference copy (T) of the motor command. Within this system, an internal neural model of the musculoskeletal system is acquired. Once the internal model is formed by motor learning, it can provide an approximated prediction $\tilde{\theta}$ of the actual movement of θ when it receives the motor command T. A predicted possible movement error $\theta_d - \tilde{\theta}$ is transmitted to the motor cortex via the ascending pathway and to muscles via the rubrospinal tract, which are shown by thin solid lines parallel with the main descending pathway, as the modification motor command. Since the loop time of the cerebrocerebellar communication loop (10–20 ms) is shorter than that of the supraspinal loop (40–60 ms), the performance of the internal feedback loop with the internal neural model is better than that of long-loop sensory feedback.

The cerebrocerebellum-parvocellular part of the red nucleus system, which develops extensively in primates, especially in man, receives its input from wide areas of the cerebral cortex and does not receive peripheral sensory input. That is, it monitors both the desired trajectory θ_d and the feedback motor command T_f but it does not receive information about the actual movement θ. Within the cerebrocerebellum-parvocellular red nucleus system, an internal neural model of the inverse-dynamics of the musculoskeletal system is acquired, with practice, by making use of the synaptic plasticity. The inverse-dynamics of the musculoskeletal system is defined as the nonlinear system whose input and output are inverted (trajectory θ is the input and motor command T is the output). Note that the spinocerebellum-magnocellular red nucleus system provides a model of the dynamics of the musculoskeletal system. The inverse-dynamics model is not a model of the external world; rather it is a model of the information processing done in other brain regions such as the motor cortex and the spinocerebellum, which com-

putes the motor command from the desired trajectory. Once the inverse-dynamics model is acquired by motor learning, it can compute a good motor command T_i directly from the desired trajectory θ_d.

Kawato, Isobe et al. (in press) proposed that some parts of sensory association cortex (areas 2, 5, and 7) solve the problems of coordinates transformation and generation of motor command simultaneously by an iterative learning algorithm (the motor command ΔT produced by the sensory association cortex in Figure 2). In this paper, this iterative learning is not discussed further.

2.1. Internal Neural Model with Heterosynaptic Plasticity

Let us consider a neural model as an identifier of an unknown nonlinear system (Figure 3). It approximates the output $z(t)$ of the unknown nonlinear system by monitoring both the input $u(t)$ and the output $z(t)$ of this system. The input $u(t)$ to the unknown nonlinear system is also fed to n subsystems and is nonlinearly transformed into n different inputs $x_k(t)(k = 1, \ldots, n)$ to the neuron with plasticity. Let w_k denote a synaptic weight of the kth input. The output signal of the neuron $y(t)$ is the sum of n postsynaptic potential: $y(t) = W^T X(t)$.

The second synaptic input to the neuron is an error signal, and is given as an error between the output of the neuron $y(t)$ and the output from the unknown nonlinear system $z(t)$: $s(t) = z(t) - y(t)$. Based on the physiological information about the heterosynaptic plasticity of the red nucleus neurons (Tsukahara, Oda, & Notsu, 1981) and the Purkinje cells (Ito, 1984), we assume that the kth synaptic weight w_k changes when the conjunction of the kth input $x_k(t)$ and the error signal $s(t)$ occurs:

$$\tau dW(t)/dt = X(t)s(t) = X(t)[z(t) - X(t)^T W(t)],$$
$$X = (x_1, x_2, \ldots, x_n)^T,$$
$$W = (w_1, w_2, \ldots, w_n)^T. \quad (1)$$

Here, τ is a time constant of change of the synaptic weight. It has been proved that the synaptic weights converge to optimum values for approximating the nonlinear system if the time constant of synaptic modification is very long (Kawato et al., 1987). Because the time constants of physiologically known synaptic plasticities are sufficiently long (from a few hours to a few weeks) compared with temporal patterns of movement (several hundreds ms), the assumption of the theorem is satisfied. It is worthwhile to note that the averaged equation of Equation (1) gives the steepest descent

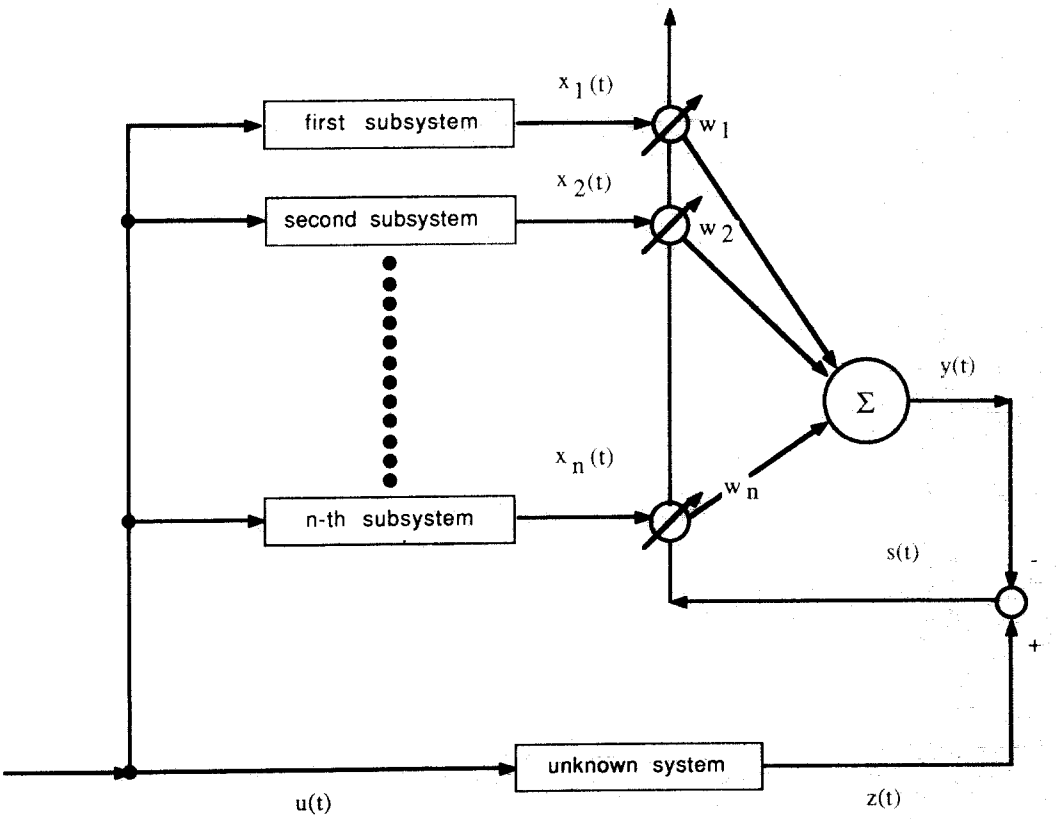

FIGURE 3. A neural identifier of an unknown nonlinear system comprises many nonlinear subsystems and a neuron with heterosynaptic plasticity.

method and the convergence is global. The basic organization of the neural identifier shown in Figure 3 is similar to the LMS adaptive filter of Widrow, McCool, Larimore, & Johnson, 1976) or the adaptive filter model of Fujita (1982), although they dealt with a linear system and the proof of convergence was different.

2.2. Acquisition of Inverse-Dynamics Model by Feedback-Error-Learning

In this paper, for simplicity we consider a three degree of freedom manipulator as a controlled object. Although it is much simpler than musculoskeletal systems such as the human arm, they both have several essential features such as nonlinear dynamics and interactions between multiple degrees of freedoms in common.

The simplest block diagram for acquiring the inverse-dynamics model of a controlled object by the heterosynaptic learning rule is shown in Figure 4a. As shown in Figure 4a, the manipulator receives the torque input $T(t)$ and outputs the resulting trajectory $\theta(t)$. The inverse dynamics model is set in the opposite input–output direction to that of the manipulator, as shown by the arrow. That is, it receives the trajectory as an input and outputs the torque $T_i(t)$. The error signal $s(t)$ is given as the difference between the real torque and the estimated torque: $s(t) = T(t) - T_i(t)$. However, this architecture does not seem to be used in the CNS because of the following three reasons. First, after the in-

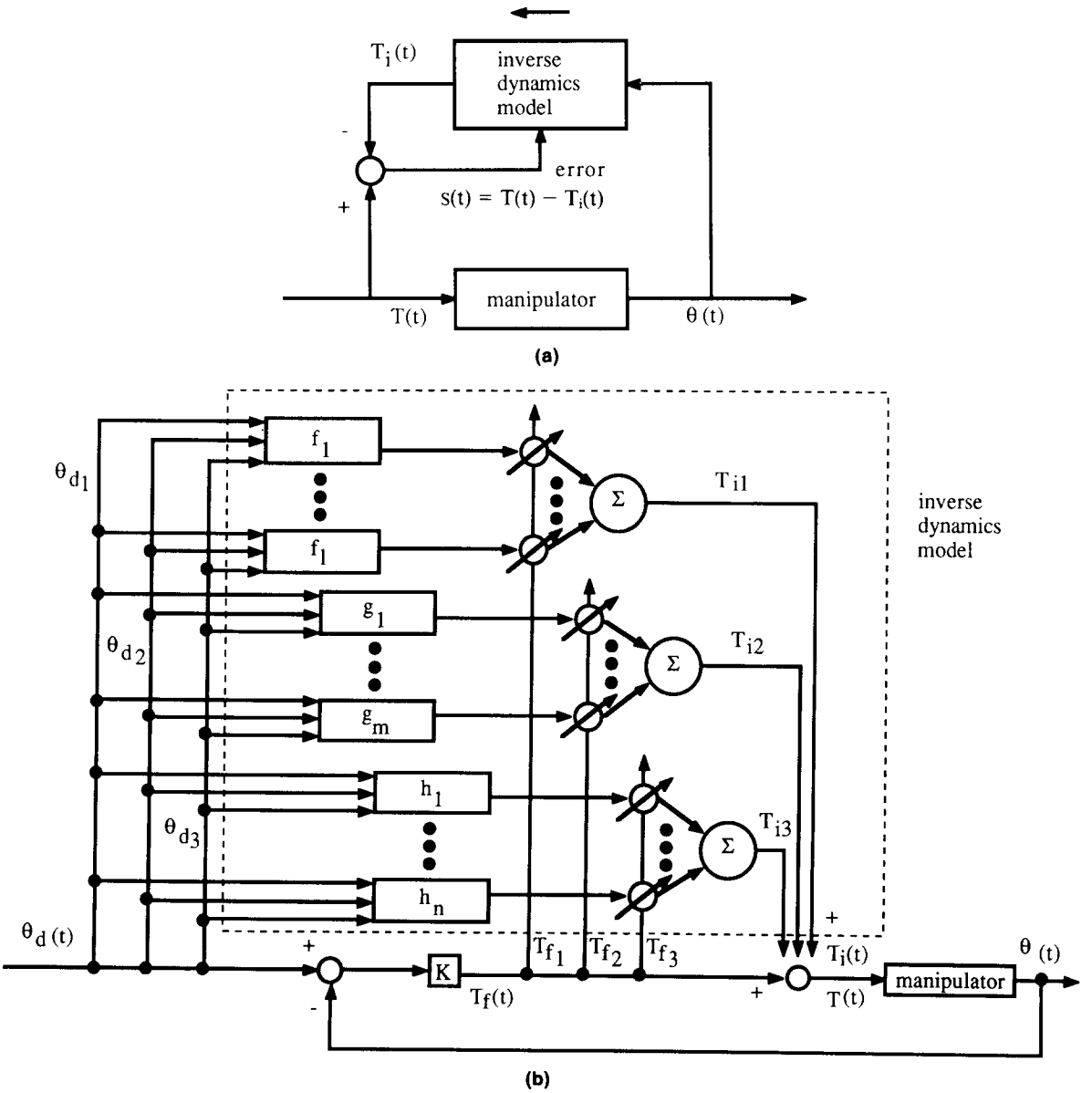

FIGURE 4. Learning schemes of the inverse-dynamics model of the controlled system. (a) The simplest learning method. The arrow shows the direction of signal flow in the inverse-dynamics model. (b) The feedback-error-learning scheme and internal structure of the inverse-dynamics model for a three degree of freedom robotic manipulator.

verse-dynamics model is acquired, large scale connection change must be done for its input from the actual trajectory to the desired trajectory, so that it can be used in feedforward control. This change of connections must be very precise so that the kth component of the actual trajectory corresponds to the same kth component of the desired trajectory. It is very hard to imagine that such large scale connection changes occur in the CNS while preserving the minute one-to-one correspondence. Second, we need another supervising neural network which determines when the connection change should be done, in other words when the learning is regarded as complete. Third, this method which separates the learning and control modes cannot cope with aging changes of the manipulator dynamics or a sudden change of a payload.

Figure 4b shows a detailed structure of the inverse-dynamics model and the block diagram which illustrates arrangement of the inverse-dynamics model, the controlled object (manipulator) and the feedback loop. This block diagram corresponds to a subsystem of our hierarchical neural-network model shown in Figure 2, which includes the motor cortex, the transcortical loop and the cerebrocerebellum-parvocellular red nucleus system. Please note that the arrangement and structure of the model shown in Figure 4b is completely different from those of the model shown in Figure 4a although both models serve as the inverse-dynamics model of the controlled object. First, the input used for calculating the output is different in the two models: It is θ in Figure 4a while it is θ_d in Figure 4b. Second, the error signal for synaptic modification is different. It is the error $T - T_i$ in Figure 4a but it is the feedback torque T_f in Figure 4b. Third, the directions of the information flow within the inverse-dynamics model are opposite between these two models. Although it is opposite to the direction of the information flow in the controlled object for the model in Figure 4a as shown by the arrow, it is the same for the model in Figure 4b.

The most important assumption of the model shown in Figure 4b is that the torque fed to the manipulator is the sum of the feedback torque generated by the somatosensory feedback loop and the feedforward torque generated by the inverse-dynamics model. That is, the total torque $T(t)$ fed to an actuator of the manipulator is a sum of the feedback torque $T_f(t)$ and the feedforward torque $T_i(t)$, which is calculated by the inverse-dynamics model. The feedback torque is calculated from the trajectory error $\theta_d - \theta$ multiplied by the feedback gain K. As shown in Figure 2, the feedback signal θ is sent back to the motor cortex by the transcortical loop, and there feedback motor command T_f is calculated. Although the total torque is a simple sum of the feedback torque and the feedforward torque, these two play totally different roles in motor command generation. The feedback torque is used for clumsy but robust control at an early stage of motor learning. It is also essential as the error signal for synaptic plasticity. The feedforward torque is necessary for smooth control of fast movements and it must be learned from experiences.

The inverse-dynamics model receives the desired trajectory θ_{dj}, ($j = 1, 2, 3$) represented as the three joint angles as input $u(t)$, and monitors the feedback torque $T_f(t)$ as the error signal $s(t)$. In Figure 4b, θ_d, θ, T_f, T_i, and T are three-dimensional vectors. θ_{dj}, T_{fj}, and T_{ij} represent the jth component ($j = 1, 2, 3$) of the vectors θ_d, T_f, and T_i. The jth component of the feedback torque T_{fj} is used to modify the synaptic weights of the jth output neuron as shown by the three upward arrows in Figure 4b.

We explain the correspondence between the internal model of the inverse-dynamics in Figure 2 and the inverse-dynamics model in Figure 4b. The input used for calculating the output are denoted by θ_d in both figures. The outputs of the inverse-dynamics model are denoted as T_i in both figures. The error signal used for synaptic modification is denoted as the feedback motor command T_f in both figures.

Let w_{kj} denote the kth synaptic weight of the jth neuron ($j = 1, 2, 3$). The learning equation for the synaptic weights of the three neurons contained in the inverse-dynamics model of Figure 4b can be written as follows:

$$\tau dw_{k1}/dt = f_k(\theta_{d1}(t), \theta_{d2}(t), \theta_{d3}(t)) \times T_{f1}(t), \quad (k = 1, \ldots, l),$$

$$\tau dw_{k2}/dt = g_k(\theta_{d1}(t), \theta_{d2}(t), \theta_{d3}(t)) \times T_{f2}(t), \quad (k = 1, \ldots, m),$$

$$\tau dw_{k3}/dt = h_k(\theta_{d1}(t), \theta_{d2}(t), \theta_{d3}(t)) \times T_{f3}(t), \quad (k = 1, \ldots, n), \quad (2)$$

here nonlinear transformation of subsystems for the first, second, and third neurons is denoted by f_k, g_k and h_k, respectively (see Figure 4b), and T_{fj}($j = 1, 2, 3$) is the jth component of the feedback torque which is fed to the jth motor of the manipulator. This architecture for learning of the inverse-dynamics model has several advantages over the learning scheme, such as that shown in Figure 4a or those of Rumelhart (in press) and Psaltis, Sideris, and Yamamura (1987). First, the teaching signal or the desired output for the neural model is not required. Instead, the feedback torque is used as the error signal. Second, the control and learning are done simultaneously. Third, back-propagation of the error signal through the controlled object (Psaltis et al., 1987) or through the model of the controlled object (Rumelhart, in press) is not necessary at all.

Because the feedback torque is chosen as the error signal for the heterosynaptic learning rule, the feedback torque is expected to decrease with learning. This implies that the error between the desired trajectory and the realized trajectory tends to zero as learning pro-

ceeds. We call this learning scheme as "feedback-error-learning" emphasizing the importance of using the feedback torque (motor command) as the error signal fo the heterosynaptic learning. The potential of the feedback-error-learning to control a complex nonlinear object was first demonstrated by computer simulations (Kawato et al., 1987). In the study of the neural network which learns to output appropriate motor commands from various input–output examples, there always exists the difficult and essential problem about how a teacher for the neural network is obtained. In the feedback-error-learning this problem is essentially solved based on the hierarchical structure of the CNS. The fundamental and general idea that the output of the lower neural network is used as an error signal for learning of the upper overlaying neural network may be utilized for various applications of adaptive neural networks and may be observed in various parts of the real brain.

3. APPLICATION TO INDUSTRIAL ROBOTIC MANIPULATOR

The neural network shown in Figure 4b was applied to learning trajectory control of an industrial robotic manipulator (Kawasaki-Unimate PUMA 260). The neural-network model was implemented in a microcomputer (Hewlett Packard 9000-300-320). Although the manipulator has six degrees of freedom, for simplicity, only the basal three degrees of freedom were controlled in the present experiment. Figure 5 shows a schematic mechanical model for the basal three degrees of freedom structure of the manipulator. The dynamics equation of this mechanical model is written as follows:

$$(A + R(\theta))\ddot{\theta} + F(\theta, \dot{\theta}) + B\dot{\theta} + G(\theta) = C\mathbf{V}, \quad (3)$$

here $\theta = (\theta_1, \theta_2, \theta_3)^T$ are the three joint angles (see Figure 5) and $\mathbf{V} = (V_1, V_2, V_3)^T$ represents voltage input to the three actuators of the manipulator. A, B, C are the 3×3 diagonal matrix with positive elements. R is an inertia matrix of the manipulator, which is positive definite. A represents inertia of the reduction gears and motors. The first, second, third, and fourth terms of the left-hand-side of the dynamics equation (3) represent the inertia term, the centripetal and Coriolis force, the frictional force and the gravitational force, respectively. As can be seen from the form of (3), there is interaction between different joints, and the dynamical property of the manipulator depends on its posture. That is, the dynamics equation (3) has nonlinearity.

3.1. Experimental Methods

The inverse-dynamics equation of the manipulator can be derived from corresponding dynamics equation (3) as follows:

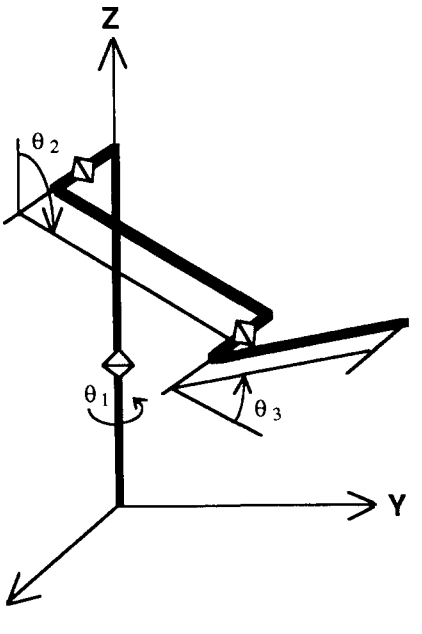

FIGURE 5. A schematic mechanical model of a manipulator with three joints and three degrees of freedom.

$$\mathbf{V} = C^{-1}\{(A + R(\theta))\ddot{\theta} + F(\theta, \dot{\theta}) + B\dot{\theta} + G(\theta)\}. \quad (4)$$

This equation can be regarded as defining nonlinear transformation from θ to $\mathbf{V}$, which gives required voltage input $V(t)$ so that the desired time history of the joint angles $\theta(t)$ is realized. This nonlinear transformation is called inverse dynamics of the controlled object in the robotics literature. In Equation (3), the voltage is represented as a linear summation of various nonlinear terms. We chose these nonlinear terms as the nonlinear transformation of subsystems in Figure 4b. Table 1 shows 43 nonlinear subsystems used in the control experiment. Numbers of subsystems for the first, second, and third neurons are 17, 16, and 10, respectively. The 1–7 subsystems of the first joint are inertia forces, the 8–15 subsystems of the first joint represent centripetal and Coriolis forces and the 16–17 subsystems of the first joint represent the frictional forces. The 15–16 subsystems of the second joint and the 10th subsystem of the third joint represent the gravitational force. Synaptic weights w_{kj} in the inverse-dynamics model of Figure 4b correspond to the coefficients of these nonlinear terms in the inverse-dynamics equation (4). In principle, these coefficients can be calculated from manipulator mechanical parameters such as length of joints, mass of joints, position of center of mass, inertia moments of joints and motors, frictional coefficients, reduction ratios of gears, etc., but they are difficult to be precisely estimated in practice.

The feedback torque (voltage input to the motors) was chosen as a sum of proportional and derivative terms:

$$V_{fj} = K_p(\theta_{dj} - \theta_j) + K_d(\dot{\theta}_{dj} - \dot{\theta}_j), \quad (j = 1, 2, 3), \quad (5)$$

TABLE 1
Nonlinear Transformation of 43 Subsystems Which are Chosen From the Manipulator Dynamics Equation (3)

	Joint 1	Joint 2	Joint 3
1	$\ddot{\theta}_{d1}$	$\cos\theta_{d2}\ddot{\theta}_{d1}$	$\cos(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d1}$
2	$\sin\theta_{d2}\sin(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d1}$	$\cos(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d1}$	$\ddot{\theta}_{d2}$
3	$\sin^2\theta_{d2}\ddot{\theta}_{d1}$	$\ddot{\theta}_{d2}$	$\cos\theta_{d3}\ddot{\theta}_{d2}$
4	$\sin^2(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d1}$	$\cos\theta_{d3}\ddot{\theta}_{d2}$	$\ddot{\theta}_{d3}$
5	$\cos\theta_{d2}\ddot{\theta}_{d2}$	$\cos\theta_{d3}\ddot{\theta}_{d3}$	$-\sin(\theta_{d2}+\theta_{d3})\cos(\theta_{d2}+\theta_{d3})\dot{\theta}^2_{d1}$
6	$\cos(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d2}$	$\ddot{\theta}_{d3}$	$-\sin\theta_{d2}\cos(\theta_{d2}+\theta_{d3})\dot{\theta}^2_{d1}$
7	$\cos(\theta_{d2}+\theta_{d3})\ddot{\theta}_{d3}$	$-\sin\theta_{d2}\cos\theta_{d2}\dot{\theta}^2_{d1}$	$\sin\theta_{d3}\dot{\theta}^2_{d2}$
8	$\sin\theta_{d2}\cos\theta_{d2}\dot{\theta}_{d1}\dot{\theta}_{d2}$	$-\sin(\theta_{d2}+\theta_{d3})\cos(\theta_{d2}+\theta_{d3})\dot{\theta}^2_{d1}$	$\dot{\theta}_{d3}$
9	$\sin\theta_{d2}\cos(\theta_{d2}+\theta_{d3})\dot{\theta}_{d1}\dot{\theta}_{d2}$	$-\sin\theta_{d2}\cos(\theta_{d2}+\theta_{d3})\dot{\theta}^2_{d1}$	$\mathrm{sig}(\dot{\theta}_{d3})$
10	$\sin(\theta_{d2}+\theta_{d3})\cos\theta_{d2}\dot{\theta}_{d1}\dot{\theta}_{d2}$	$-\sin(\theta_{d2}+\theta_{d3})\cos\theta_{d2}\dot{\theta}^2_{d1}$	$-\sin(\theta_{d2}+\theta_{d3})$
11	$\sin\theta_{d2}\cos(\theta_{d2}+\theta_{d3})\dot{\theta}_{d1}\dot{\theta}_{d3}$	$-\sin\theta_{d3}\dot{\theta}_{d2}\dot{\theta}_{d3}$	
12	$\sin(\theta_{d2}+\theta_{d3})\cos(\theta_{d2}+\theta_{d3})\dot{\theta}_{d1}\dot{\theta}_{d2}$	$-\sin\theta_{d3}\dot{\theta}^2_{d3}$	
13	$\sin(\theta_{d2}+\theta_{d3})\cos(\theta_{d2}+\theta_{d3})\dot{\theta}_{d1}\dot{\theta}_{d3}$	$\dot{\theta}_{d2}$	
14	$-\sin\theta_{d2}\dot{\theta}^2_{d2}$	$\mathrm{sig}(\dot{\theta}_{d2})$	
15	$-\sin(\theta_{d2}+\theta_{d3})(\dot{\theta}_{d2}+\dot{\theta}_{d3})^2$	$-\sin\theta_{d2}$	
16	$\dot{\theta}_{d1}$	$-\sin(\theta_{d2}+\theta_{d3})$	
17	$\mathrm{sig}(\dot{\theta}_{d1})$		

here V_{fj} is the feedback voltage input to the jth motor, θ_{dj} is the jth joint angle of the desired trajectory, and θ_j is the jth joint angle of the realized trajectory.

Figure 6 schematically shows experimental setup of the digital computer, input–output data processor, control unit, and manipulator. Joint angles were measured by photo encoders and were sampled at 10 ms time intervals. The feedback (V_f) and the feedforward (V_i) voltages were computed with the same 10 ms time intervals. Derivatives of joint angles necessary for calculation of subsystems output in Table 1 ($\dot{\theta}_d$, $\dot{\theta}_d$), and for calculation of the feedback torque ($\dot{\theta}$) were done numerically. The program for neural network learning and control of manipulator was written in Pascal and the learning of synaptic weights was done in real time. The initial values of all the synaptic weights at the beginning of the experiment were set at 0. That is, at the beginning of the learning experiment, control of the manipulator was executed only by the feedback voltage: Equation (5). The input–output data processor was fabricated by ourselves.

4. EXPERIMENTAL RESULTS

Experiment I: In the first experiment the learning time constants τ of all the synaptic weights were set 1000 s. A desired trajectory lasting for 6 s, which is shown by chain curves in Figure 8a, were given 300 times to the control system repeatedly. The learning time was 30 min. The feedback gains were set as $K_p = 60$ and $K_d = 1.2$. As the learning proceeded, both the mean square of the feedback voltage $\overline{V_f^2}$ and the mean square error of the trajectory $\overline{(\theta_d - \theta)^2}$ decreased. Figure 7a plots the change of the mean square error of

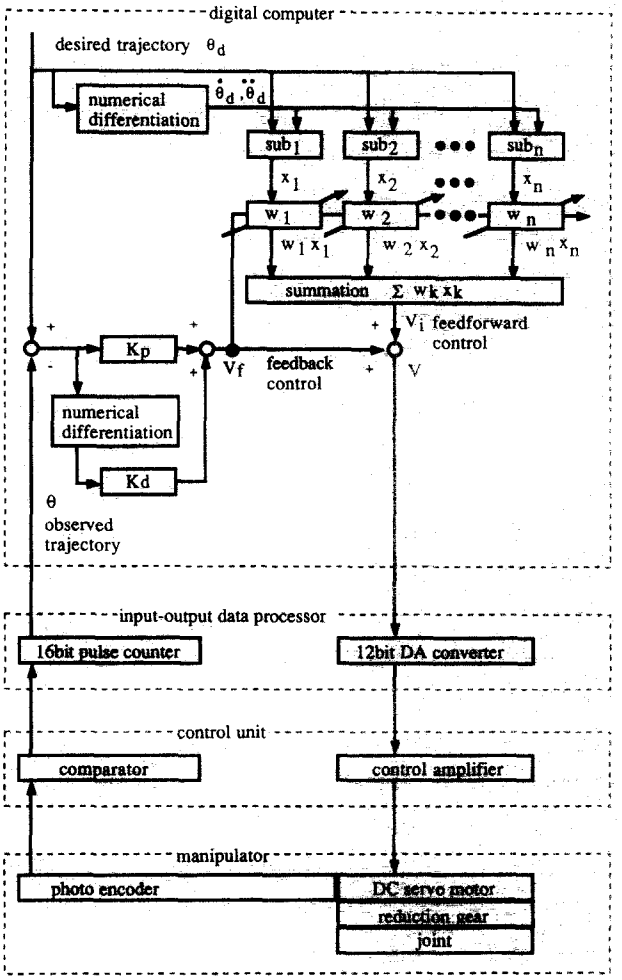

FIGURE 6. Schematic diagram of an experimental system for feedback-error-learning of inverse dynamics of an industrial robotic manipulator.

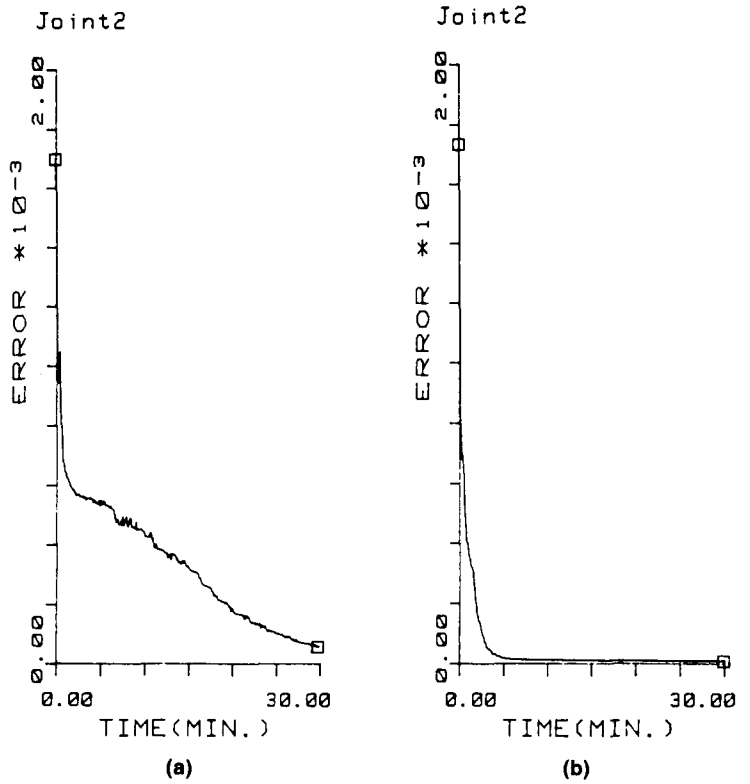

FIGURE 7. Change of the mean square error of the second joint angle during 30 minutes of learning. The unit of the ordinate is radian^2. (a) In Experiment I the time constants for synaptic modification of all synaptic weights were equal to 1000 s. (b) In Experiment II the time constants for synaptic modifiçation were different for different subsystems.

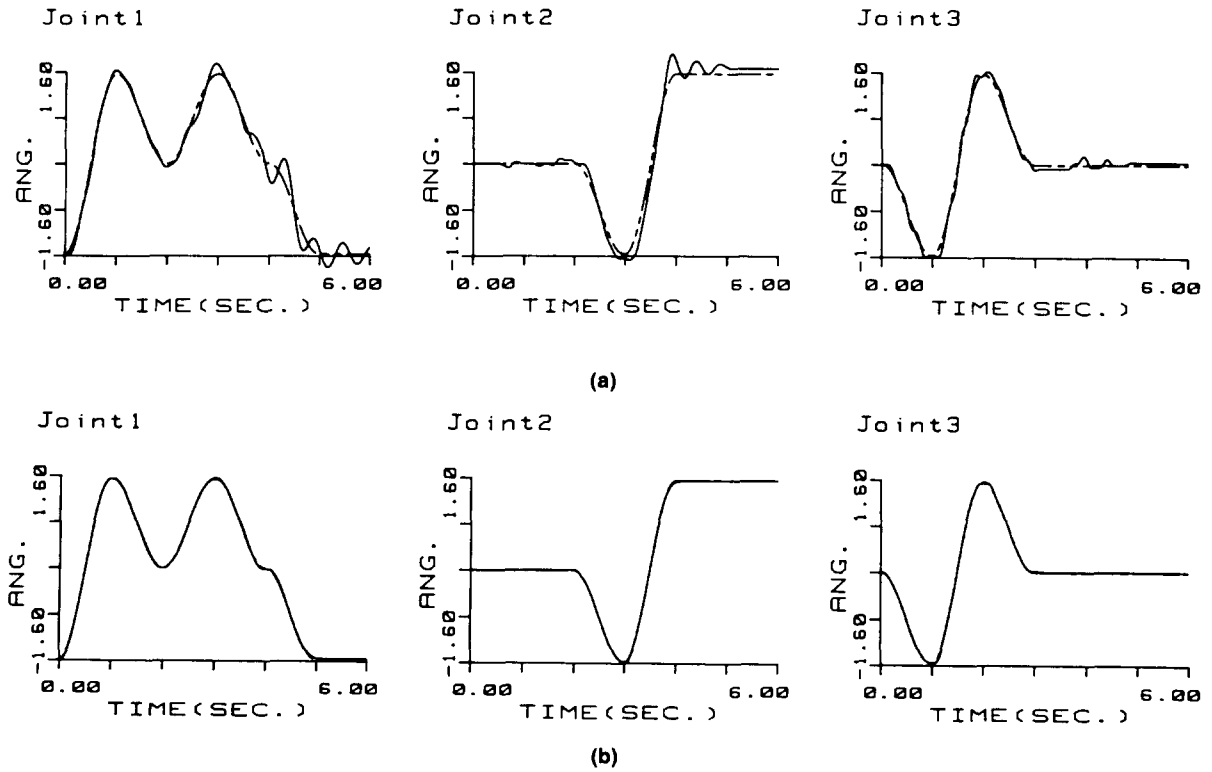

FIGURE 8. (a) Time courses of the three joint angles during 6 s of a single repetition of the training movement pattern before learning. Desired trajectories are shown by chain curves and realized trajectories by solid curves. The unit of the ordinate is radian. (b) Same as (a) but after 30 minutes of learning.

the second joint angle, $\overline{(\theta_{d2} - \theta_2)^2}$ during 30 minutes of learning. The time average was taken over 6 s of the single repetition of the training pattern. In this experiment, learning speed of 43 synaptic weights were wildly different with each other. The reason for this is that the magnitude of outputs from the different subsystems shown in Table 1 were very different and the learning speed of the synaptic weight is proportional to the output of the corresponding subsystem (see Equation 2).

Experiment II: The time constants of change of synaptic weights were chosen as in Table 2 so that the learning speeds of different synaptic weights became more similar and that the overall learning speed of the system improved. Each value of the time constant was pragmatically chosen based on the observed change of the corresponding synaptic weight in Experiment I. If a change of some synaptic weight was slow, the corresponding time constant was decreased. If synaptic modification was too rapid and unstable, the corresponding time constant was increased. Figure 7b shows the change of mean square error of the second joint angle, $\overline{(\theta_{d2} - \theta_2)^2}$ during 30-min learning. As can be clearly seen from comparison of Figure 7a and b, the overall learning speed dramatically improved by modification of time constants for synaptic modification. Similar effects are expected by more systematic approach. That is, if we compute the magnitude of outputs of various subsystems for expected range of input patterns (desired trajectory) before the learning experiment and normalize the outputs from the subsystems by their magnitudes, then the change of various synaptic weights are expected to be almost uniform even with the common time constant for the synaptic change.

Experiment III: A learning experiment with a poorer feedback control was done (i.e., with smaller feedback gains). The feedback gains were set $K_p = 20$ and $K_d = 0.4$, and the other experimental conditions were the same as in Experiment II. Figure 8 compares control performance before (a) and after (b) 30 minutes of learning. The three joint angles of the desired movement pattern (chain curve, $\theta_{d1}(t)$, $\theta_{d2}(t)$, $\theta_{d3}(t)$) and the realized movement pattern (solid curve, $\theta_1(t)$, $\theta_2(t)$, $\theta_3(t)$) during 6 s of a single repetition of the training movement pattern are shown. As can be seen, when control depended only on the feedback, overshoots and oscillation of the realized trajectory were observed (a), but after 30 minutes of learning the desired trajectory and the realized trajectory can not be separately seen (b). It was shown that the neural network can learn even with a poor feedback signal. This is important from the neuroscience point of view as well as that of engineering, since loop time of the transcortical reflex is long and its feedback gain is low.

TABLE 2
Time Constants (s) for Synaptic Modification for 43 Synaptic Weights in Experiment II

	Joint 1	Joint 2	Joint 3
1	1000	1000	1000
2	1000	1000	1000
3	1000	1000	2000
4	3000	1000	1000
5	3000	1000	1000
6	3000	1000	1000
7	3000	1000	1000
8	700	1000	700
9	700	1000	500
10	700	1000	500
11	700	1000	
12	1000	1000	
13	2000	700	
14	2000	100	
15	2000	100	
16	700	100	
17	700		

Figure 9 compares the feedforward voltage (V_f: top), feedback voltage (V_f: middle) and the total voltage time course (V: bottom) fed to the second motor of the manipulator before (a) and after (b) 30 minutes of learning. As learning proceeded, the feedback motor command decreased (middle) and the feedforward command increased (top). That is, the main controller shifted from the feedback mode to the feedforward mode. At the same time, the amplitude of the total voltage V decreased considerably and its time course became smoother (bottom). Consequently, the motor command became smoother and excessive energy dissipation was suppressed by motor learning. This point will be later dealt with in detail.

We then examined whether the neural network model after 30 minutes of learning a single pattern (Figure 8 chain curve) could control a quite different movement pattern (Figure 10 chain curve) which was about twice as fast as the training pattern. Figure 10 compares control performance of this test movement before (a) and after (b) 30 minutes of learning. The three joint angles of the desired trajectory during a 6 s test pattern are designated by chain curves, and those of the realized trajectory by solid curves. Before learning (a), delays and overshoots were often observed, and the realized trajectory deviated considerably from the desired trajectory. However, after learning (b), the actual trajectory almost coincided with the desired trajectory. This control capability for quite different and faster movements than the training pattern is one of the most outstanding features of our neural network model. The neural-network model has the capability to generalize learned movements.

Experiment IV: Dynamical properties of controlled objects often change due to various reasons such as aging effects or grasping a payload. In order to investigate adaptability of the network to such changes of dynamics, a payload with 315 g weight was attached to the hand of the manipulator 15 min after the start of the experiment during 30 minutes of learning. Other experimental conditions were the same as Experiment II. Figure 11a shows the change of the mean square error of the second joint angle $\overline{(\theta_{d2} - \theta_2)^2}$ during 30

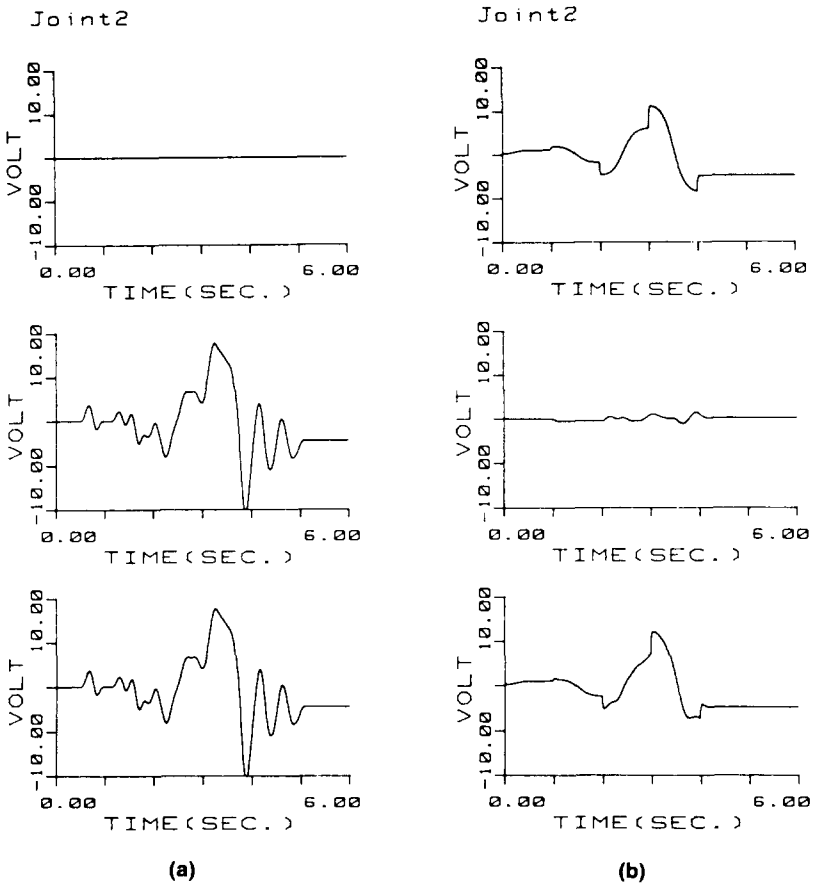

FIGURE 9. The feedforward (top), the feedback (middle) and the total voltage (bottom) fed to the second motor of manipulator before (a) and after (b) the 30 minutes of learning.

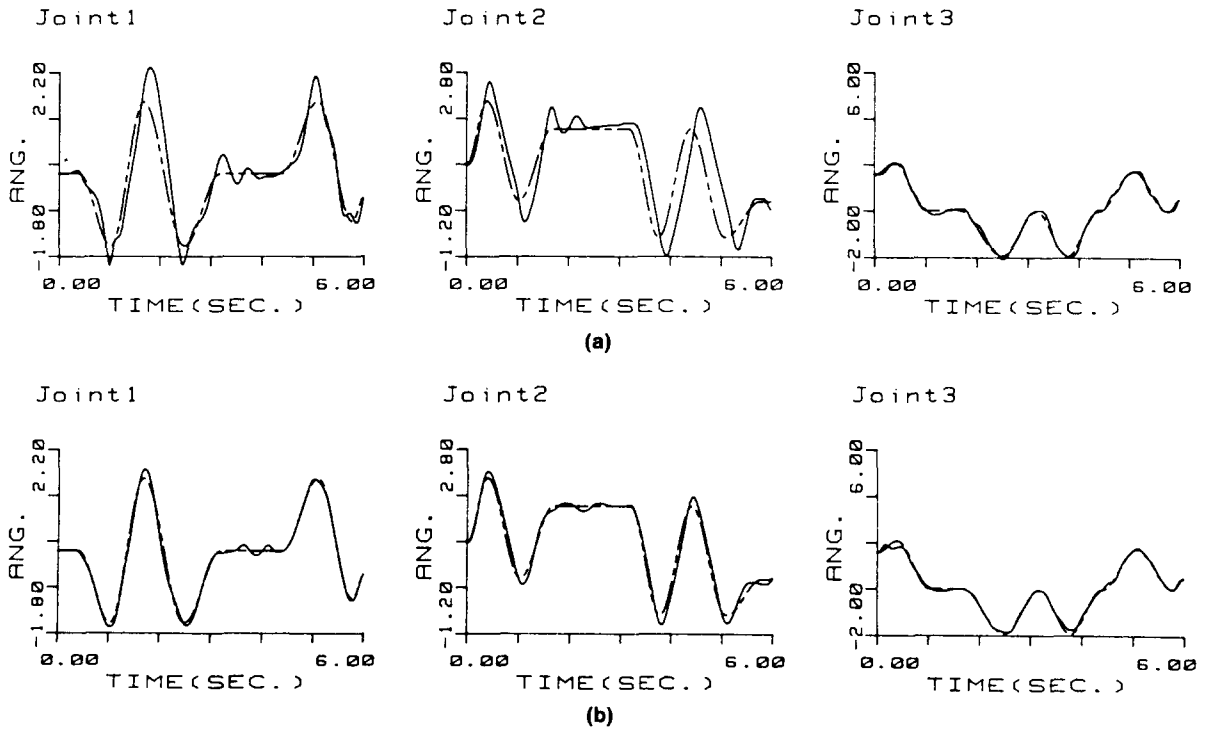

FIGURE 10. Control performance of a faster and different movement than the training pattern, before (a) and after (b) 30 minutes of learning. Desired trajectories for the three joint angles are shown by chain curves, and the realized trajectories by solid curves. The unit of the ordinate is radian.

min. The mean square error increased temporarily just after the attachment of the payload, but rapidly decreased after relearning. Consequently, the neural network possesses good adaptability to changes of dynamical characteristics of the controlled object.

If we carefully examine the manipulator dynamics equation (3), we notice that all coefficients of nonlinear terms except the frictional force $B\dot{\theta}$ change when the weight of the payload changes. Figure 11b shows changes of the 16 synaptic weights for the 16 subsystems during 30 minutes of learning, which constitute the voltage input to the second motor of the second joint. As expected, all the synaptic weights suddenly changed their directions of changes after the payload was attached, except the 13th and 14th synaptic weights which correspond to the unaffected frictional force (see Table 1). This experimental result provides additional evidence that the neural network properly acquires the inverse-dynamics model of the controlled object under the proposed learning rule.

4.1. Summary of Results

We applied the neural network model for learning trajectory control of the industrial robotic manipulator with the following results.

1. The inverse-dynamics model was acquired by repetitively experiencing a single motor pattern during 30 minutes of learning. The mean square error of the trajectory decreased, that is, the control performance improved considerably during learning.
2. Figure 12a shows mean squares of the feedforward torque (top: $\overline{V_f^2}$), feedback (middle: $\overline{V_f^2}$) and the total torque (bottom: $\overline{V^2}$) during 30 minutes of learning. The feedforward torque increased (top) while the feedback torque decreased considerably (bottom). This implies that as motor learning proceeded, the inverse-dynamics model gradually took the place of the feedback loop as a main controller. Furthermore, one may notice that the mean square of the total income torque significantly decreased during 30 minutes of learning (bottom). This might be related to the fact that we can gracefully and smoothly control very accustomed movement without excessive forces.
3. Once the neural network model learned some movement, it could control quite different and faster movements. It has capability to generalize learned movements.
4. The model had adaptability to a sudden change in dynamics of the manipulator.

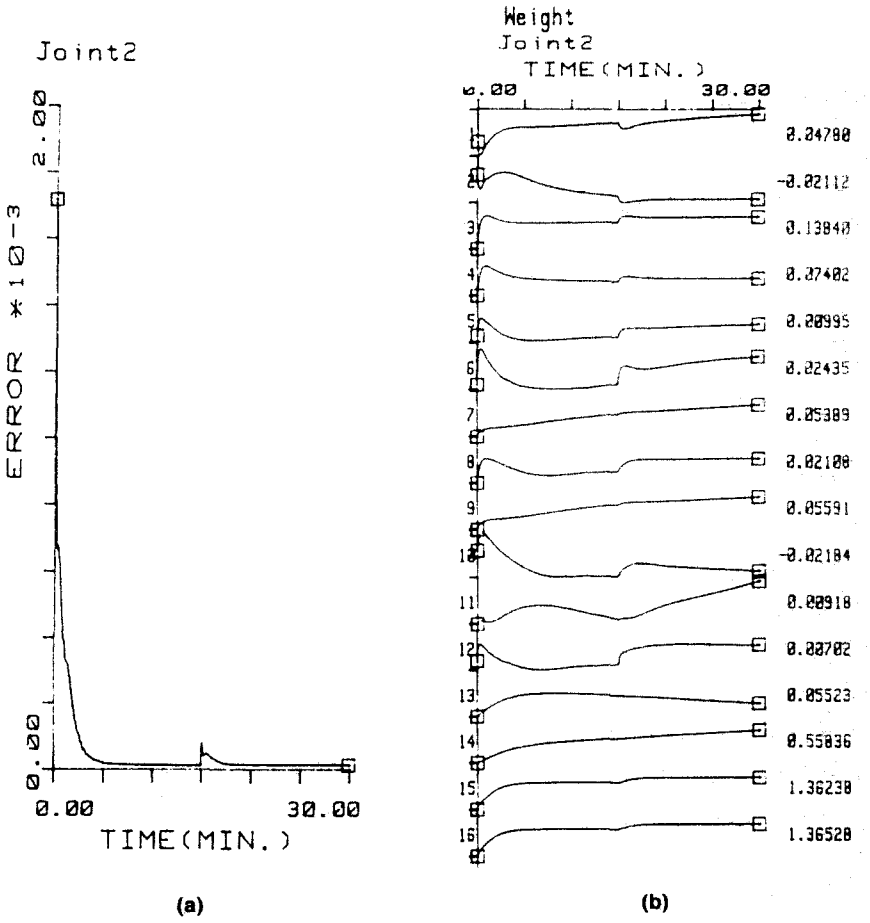

FIGURE 11. (a) Change of the mean square error of the second joint angle during Experiment IV. The unit of the ordinate is radian^2. (b) Time courses of 16 synaptic weights for the second motor during 30 minutes of learning in Experiment IV.

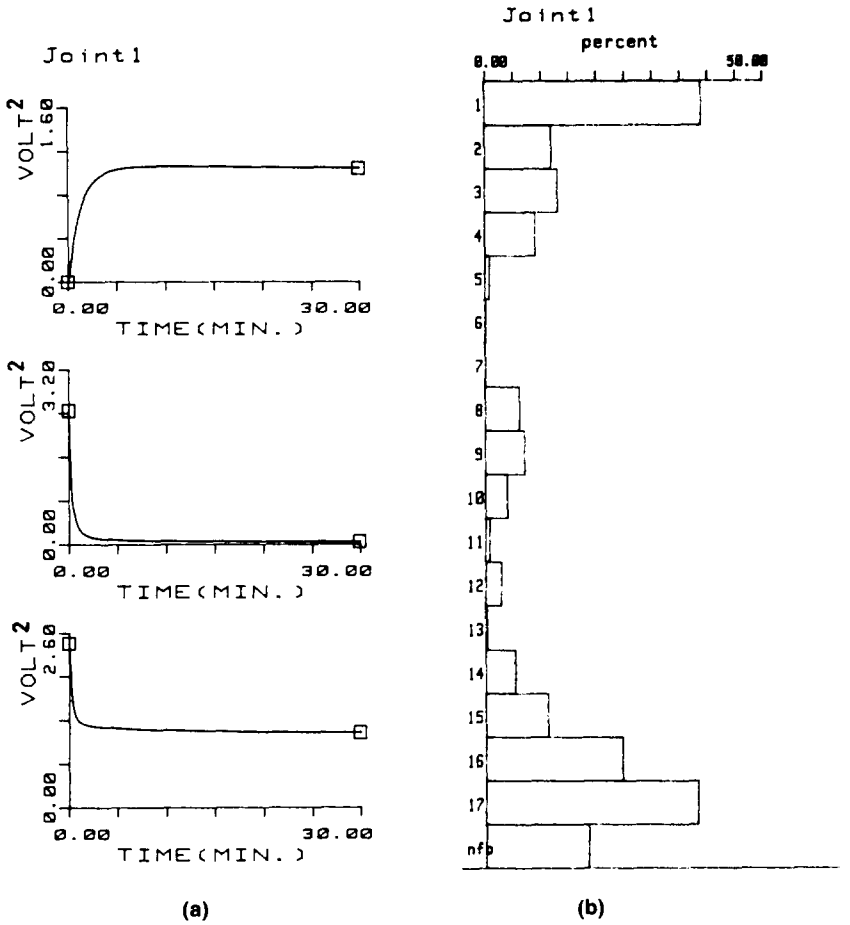

FIGURE 12. (a) Change of the mean square of the feedforward (top), feedback (middle) and the total voltage (bottom) fed to the first motor during 30 min learning in the Experiment II. (b) Percentage of contribution of 17 subsystems to the total input fed to the first motor after 30 minutes of learning in Experiment II.

Figure 12b shows the percentage of mean magnitude of weighted output of 17 subsystems ($|w_{k1}f_k|$) and the mean magnitude of the feedback torque (nf: $|T_{f1}|$) in the total torque (T_1) applied to the first joint after 30 minutes of learning. This figure is to compare relative contribution of the 17 subsystems to the total motor command. As expected, contributions of the main inertia term (subsystem 1) and the frictional forces (subsystems 16, 17) occupy a large proportion. As can be seen from this figure and Table 1, the nonlinearity in the dynamics of the manipulator is considerable (subsystems 2–4, 8–10, 14–15) although the reduction ratio of gears at joints were high (30–100) in the PUMA manipulator. It must be noted that the merit of the neural network control proposed in this paper is significantly magnified for a direct drive manipulator (reduction ratio = 1), whose dynamics has strong nonlinearity and which is considered as future major robotic manipulators.

5. DISCUSSION

There are two possibilities about how the CNS computes nonlinear transformations of the subsystems. One is that they are computed by nonlinear information processing within the dendrites of neurons (Kawato, Hamaguchi, Murakami, & Tsukahara, 1984; Poggio & Torre; 1981). The other possibility is that they are realized by neural circuits. Recently we succeeded in learning control of the robotic manipulator by an inverse-dynamics model made of a three-layer neural network (Setoyama, Kawato, & Suzuki, unpublished observation). We used a modification of the back-propagation learning rule (Rumelhart, Hinton, & Williams, 1986) while still using the feedback torque command as the error signal. So, in the new experiment we successfully expanded the feedback-error-learning to a multilayer neural network. These results are quite interesting since both the Purkinje cells and the red nucleus neurons have heterosynaptic plasticity in our neural-network model (see Figure 2).

In this multilayer neural network, the subsystems performing nonlinear transformation were not used. That is, we did not use any a priori knowledge about the dynamical structure of the controlled object. Application of the neural network to control of constrained movements for robotic manipulators such as object manipulation is very appealing. The reason is that un-

derstanding or modeling of interactions between controlled objects (manipulators) and their environments is conceptually very difficult. The multilayer neural network does not require any a priori knowledge about these interactions and all we need to do is to feed the neural-network model necessary information (desired trajectory, desired force, desired impedance, feedback motor command) and to let it learn by examples. Furthermore, application of the feedback-error-learning in combination with the multilayer neural network is very promising for feedforward control of almost all kinds of large-scale complex systems whose fundamental dynamical structure is not known.

Finally, we assess the present model from an engineering point of view. Although feedback controls have been nearly adequate for the slow control of usual industrial robotic manipulators, new control methods have been demanded for direct-drive manipulators. Several learning and adaptive control schemes were proposed (model reference adaptive control [Dubowsky & Des Forges, 1979], betterment process [Arimoto, Kawamura, & Miyazaki, 1984], table look-up method [Albus, 1975; Barto, Sutton, & Anderson, 1983; Raibert, 1978]), but they were at most perturbation-learning schemes. That is, experiences obtained during learning cannot be used for the execution of a quite different movement. The method of computed torque (Luh, Walker, & Paul, 1980) requires both strict modeling of the manipulator dynamics and the precise estimation of physical parameters, which are difficult in practice. In contrast, the present method requires neither an accurate model nor parameter estimation. Further, it possesses a great ability to generalize learning.

We must also emphasize that the present model can be easily implemented in a parallel distributed processing machine, since both the nonlinear transformations in subsystems and the synaptic modifications are essentially parallel. This merit is magnified when the degrees of freedom of the controlled object increase. In order to apply the proposed neural-network model to trajectory control of the full six degrees of freedom of the PUMA robot, we first need to calculate nonlinear transformation of the subsystems which appear in Equation (4). However, it is very tedious and almost impractical to calculate manually the nonlinear transformation for six degrees of freedom since the number of the subsystems for n degrees of freedom manipulator increases in the order of n^4. We calculated the nonlinear transformation of the PUMA robot for the full six degrees of freedom based on the Lagrangian equation using REDUCE, a programming language for explicit handling of mathematical formulas. The total number of subsystems was 942. The numbers of subsystems for the first to the sixth joints were 302, 178, 151, 119, 110, and 80, respectively. Numbers of subsystems contained in the inertia force, the centripetal force, the Coriolis force and the gravitational force were 222, 205, 505 and 10, respectively. We then compared computational time required to calculate the six torques at one time point by the recursive formulation of the Newton-Euler dynamics (Luh et al., 1980) and by the neural-network model proposed in this paper based on nonlinear transformation calculated by REDUCE. The program was written in HP-Pascal and was run on HP-9000-300-320 microcomputer. The torque computed by the two different methods perfectly coincided. So, the programming in REDUCE was verified. The computation time of the recursive Newton-Euler method (Luh et al., 1980) was 0.04 s, which was regarded fastest in robotics literature. The computation time of the present neural-network model was 0.08 s and twice longer than the computed torque method. However, the neural network has the strong parallel structure with about 1000 subsystems which can run completely in parallel. So if the present neural network is implemented on a parallel machine with at least 1000 processor nodes, the computation time is expected to be shorter than at least the order of 1/100. On the other hand, the recursive Newton-Euler method (Luh et al., 1980), or recursive Lagrangian method proposed in robotics literature cannot fully benefit from parallel-distributed-processing because of their inherent serial (recursive) structures of the algorithms. This advantage of the neural network over conventional control algorithms increases in order of n^4 which is the number of synapses and subsystems for the n degrees of freedom manipulator. Consequently, the neural-network control will play very important roles in control of objects with many degrees of freedom (e.g., manipulators with redundancy).

Furthermore, we emphasize that the present neural-network model does not require the enormous memory size of the table look-up method (only 942 synaptic weights are necessary for a six degree-of-freedom manipulator). In summary, the present method is one of the promising schemes for the future control of not only a direct drive manipulator, but also of a large-scale complex system, whose dynamics is only partially known.

REFERENCES

Albus, J. S. (1975). A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement and Control*, **97**, 270–277.

Allen, G. I., & Tsukahara, N. (1974). Cerebrocerebellar communication systems. *Physiological Reviews*, **54**, 957–1006.

Arimoto, S., Kawamura, S., & Miyazaki, F. (1984). Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems. *Proceedings of the 23rd IEEE Conference on Decision and Control*, **2**, 1064–1069.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, Cybernetics*, **SMC-13**, 834–846.

Cheney, P. D., & Fetz, E. E. (1980). Functional classes of primate

corticomotoneuronal cells and their relation to active force. *Journal of Neurophysiology,* **44,** 773–791.

Dubowsky, S., & DesForges, D. T. (1979). The application of model reference adaptive control to robotic manipulators. *Journal of Dynamic Systems, Measurement and Control,* **101,** 193–200.

Evarts, E. V. (1981). Role of motor cortex in voluntary movements in primates. In V. B. Brooks (Ed.), *Handbook of physiology.* (Vol. II, p. 2, pp. 1083–1120). Bethesda: American Physiological Society.

Flash, T., & Hogan, N. (1985). The coordination of arm movements; An experimentally confirmed mathematical model. *Journal of Neuroscience,* **5,** 1688–1703.

Fujita, M. (1982). Adaptive filter model of the cerebellum. *Biological Cybernetics,* **45,** 195–206.

Grossberg, S., & Kuperstein, M. (1986). *Neural dynamics of adaptive sensory-motor control: Ballistic eye movements.* Amsterdam: North-Holland.

Hollerbach, J. M. (1982). Computers, brains and the control of movement. *Trends Neuroscience,* **5,** 189–192.

Ito, M. (1970). Neurophysiological aspects of the cerebellar motor control system. *International Journal of Neurology* **7,** 162–176.

Ito, M. (1984). *The cerebellum and neural control.* New York: Raven Press.

Kawato, M. (1988). Adaptation and learning in control of voluntary movement by the central nervous system. *Advanced Robotics,* **2.**

Kawato, M., Isobe, M., Maeda, Y., & Suzuki, R. (1988). Coordinates transformation and learning control for visually-guided voluntary movement with iteration: A Newton-like method in a function space. *Biol. Cybern.* **59.**

Kawato, M., Furukawa, K., & Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. *Biol. Cybern.* **57,** 169–185.

Kawato, M., Hamaguchi, T., Murakami, F., & Tsukahara, N. (1984). Quantitative analysis of electrical properties of dendritic spines. *Biol. Cybern.* **50,** 447–454.

Kawato, M., Uno, Y., Isobe, M., & Suzuki, R. (1988). A hierarchical neural network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine,* **8,** 8–16.

Luh, J. Y. S., Walker, M. W., & Paul, R. P. C. (1980). On-line computational scheme for mechanical manipulations. *Journal of Dynamic Systems, Measurement Control,* **102,** 69–76.

Marr, D. (1982). *Vision.* New York: Freeman.

Poggio, T., & Torre, V. (1981). A theory of synaptic interactions. In W. E. Reichardt and T. Poggio (Eds.), *Theoretical approaches in neurobiology* (pp. 28–46). Cambridge: MIT Press.

Psaltis, D., Sideris, A., & Yamamura, A. (1987). Neural controllers. *Proceedings of the IEEE 1st International Conference on Neural Networks,* **IV,** 551–558.

Raibert, M. H. (1978). A model for sensorimotor control and learning. *Biological Cybernetics,* **29,** 29–36.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature,* **323,** 533–536.

Rumelhart, D. E. (in press). Learning sensorimotor programs in parallel distributed processing systems. *Proceedings of US-Japan Joint Seminar on Competition and Cooperation in Neural Nets II.* New York: Springer.

Tsukahara, N., & Kawato, M. (1982). Dynamic and plastic properties of the brain stem neuronal networks as the possible neuronal basis of learning and memory. In S. Amari and M. A. Arbib (Eds.), *Competition and cooperation in neural nets* (pp. 430–441). New York: Springer.

Tsukahara, N., Oda, Y., & Notsu, T. (1981). Classical conditioning mediated by the red nucleus in the cat. *Journal of Neuroscience,* **1,** 72–79.

Uno, Y., Kawato, M., & Suzuki, R. (1987). Formation of optimum trajectory in control of arm movement—minimum torque-change model. *Japan IEICE Technical Report,* **MBE86-79,** 9–16.

Widrow, B., McCool, J. M., Larimore, M. G., & Johnson, C. R. (1976). Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE,* **64,** 1151–1162.