Neural Network Learning of Robot Arm Impedance in Operational Space

Toshio Tsuji, Koji Ito, *Member, IEEE*, and Pietro G. Morasso

Abstract—**Impedance control is one of the most effective control methods for the manipulators in contact with their environments. The characteristics of force and motion control, however, is determined by a desired impedance parameter of a manipulator's end-effector that should be carefully designed according to a given task and an environment. The present paper proposes a new method to regulate the impedance parameter of the end-effector through learning of neural networks. Three kinds of the feed-forward networks are prepared corresponding to position, velocity and force control loops of the end-effector before learning. First, the neural networks for position and velocity control are trained using iterative learning of the manipulator during free movements. Then, the neural network for force control is trained for contact movements. During learning of contact movements, a virtual trajectory is also modified to reduce control error. The method can regulate not only stiffness and viscosity but also inertia and virtual trajectory of the end-effector. Computer simulations show that a smooth transition from free to contact movements can be realized by regulating impedance parameters before a contact.**

I. Introduction

When a manipulator performs a task in contact with its environment, force control as well as position control are required according to constraints imposed by the environment. Impedance control can regulate an end-effector dynamics of the manipulator to the desired one, and gives us a unified approach to control force and motion of the manipulator [1]. The characteristics of force and motion control, however, is determined by desired impedance parameters of the manipulator's end-effector that should be planned according to a purpose of a given task and a characteristics of a task environment. So far, no powerful technique for appropriate planning of the impedance parameter has been developed. Learning by neural networks is one of possible approaches to adjust the impedance parameters skillfully.

Recently, several investigations that apply the neural network learning into the impedance control have been reported [2]–[7]. Gomi and Kawato [2] applied the feedback error learning scheme [8] by neural networks into nonlinear compensations of manipulator dynamics and showed that the impedance control could be implemented for unknown manipulator dynamics through learning. Venkataraman *et al.* [3] utilized a neural network for identifying environments for compliance control. The neural networks used in their methods, however, did not learn the impedance parameter of the end-effector that must be given according to the task beforehand. Maeda and Kano [4], and Cheah and Wang [5], also proposed learning control laws utilizing the Newton-like method and the iterative learning scheme, respectively, under assumptions that the desired impedance parameter is given beforehand.

On the other hand, Asada [6] showed that nonlinear viscosity of the end-effector could be realized by using the neural network model as a force feedback controller. Cohen and Flash [7] proposed a method to regulate the end-effector stiffness and viscosity of the manipulator using neural networks. The networks represented the stiffness and viscosity matrices and were trained to minimize a cost function about force and velocity control errors. Although the desired velocity trajectory for the end-effector was modified in order to improve a task performance in their method, the network could not regulate an inertia property of the end-effector and only the contact movements could be learned.

The present paper proposes a method to regulate the desired impedance of the end-effector through learning of neural networks. Three kinds of the back-propagation typed networks are prepared corresponding to the position, velocity and force control loops of the end-effector. First, the neural networks for position and velocity control are trained using iterative learning of the manipulator during free movements. Then, the neural network for force control is trained for contact movements. During learning of contact movements, the virtual trajectory is also modified to reduce the control error. The method can regulate not only stiffness and viscosity but also inertia and virtual trajectory of the end-effector, and both of the free and contact movements can be learned. Computer simulations show that a smooth transition from free to contact movements can be realized by regulating impedance parameters before a contact.

II. Impedance Control

In general, the motion equation of an n-joint manipulator in contact with an environment is given as

$$M(\theta)\ddot{\theta} + h(\theta, \dot{\theta}) + g(\theta) = \tau - J^T(\theta)F_{ext} \qquad (1)$$

where θ and $\tau \in R^n$ represent the joint angle vector and the joint torque vector, respectively; $M(\theta) \in R^{n\times n}$ is the

Manuscript received April 24, 1994; revised December 28, 1994.

T. Tsuji is with the Department of Computer Science and Systems Engineering, Faculty of Engineering, Hiroshima University, Higashi-Hiroshima 739, Japan (e-mail: tsuji@huis.hiroshima-u.ac.jp).

K. Ito is with the Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi 440, Japan, and the Bio-Mimetic Control Research Center (RIKEN), Nagoya 456, Japan.

P. G. Morasso is with the Department of Informatics, Systems and Telecommunications, University of Genova, Genova 16145, Italy.

Publisher Item Identifier S 1083-4419(96)02307-2.

nonsingular inertia tensor; $h(\theta, \dot{\theta}) \in R^n$ is the centrifugal and Colioris force vector; $g(\theta) \in R^n$ represents the gravitational torque; $J(\theta) \in R^{m \times n}$ represents the Jacobian matrix; $-F_{ext} \in R^m$ represents the external force exerted from the environment on the end-effector of the manipulator; and m is the number of degrees of freedom of the operational space. In this paper, the end-effector makes a point contact with a compliant work environment and the characteristics of the environment is expressed as [9]

$$F_{ext} = M_e\ddot{X} + B_e\dot{X} + K_e(X - X_e) \tag{2}$$

where $X \in R^m$ is the end-effector position; $X_e \in R^m$ represents the equilibrium position of the environment; and $M_e, B_e, K_e \in R^{m \times m}$ represent inertia, viscosity, and stiffness of the environment, respectively. Generally, the impedance matrices, M_e, B_e, K_e, and the equilibrium position of the environment, X_e, are unknown, while the end-effector's position, X, velocity, $\dot{X}$, and the interaction force, F_{ext} can be measured.

Using a nonlinear compensation technique such as

$$\begin{aligned}\tau = {} & h(\theta, \dot{\theta}) + g(\theta) + J^T(\theta)F_{ext} \\ & + M(\theta)J^{-1}(\theta)[F_{act} - \dot{J}(\theta)\dot{\theta}],\end{aligned} \tag{3}$$

the dynamics of the manipulator in the operational space reduces to a simplified equation [10]:

$$\ddot{X} = F_{act} \tag{4}$$

where $F_{act} \in R^m$ is the control force vector represented in the operational space.

On the other hand, a second order linear model is used as a model of the desired end-effector impedance:

$$M_d\, d\ddot{X} + B_d\, d\dot{X} + K_d\, dX = F_d - F_{ext} \tag{5}$$

where $dX \equiv X - X_d \in R^m$ is the displacement vector; X_d and $F_d \in R^m$ represent the desired hand trajectory or the virtual trajectory [11] and the desired hand force vector, respectively; and $M_d, B_d, K_d \in R^{m \times m}$ represent desired inertia, viscosity and stiffness matrices of the end-effector, respectively. From (4) and (5), the impedance control law for F_{act} is derived as follows:

$$\begin{aligned}F_{act} &= F_t + F_f + \ddot{X}_d \\ &= -M_d^{-1}(B_d\, d\dot{X} + K_d\, dX) \\ &\quad + M_d^{-1}(F_d - F_{ext}) + \ddot{X}_d\end{aligned} \tag{6}$$

where $F_t = -M_d^{-1}(B_d\, d\dot{X} + K_d\, dX)$ and $F_f = M_d^{-1}(F_d - F_{ext})$.

Fig. 1 shows the block diagram of the impedance control. During free movements, the force feedback loop in the figure does not exist because of $F_d = F_{ext} = 0$. While the manipulator is in contact with the environment, the force as well as position and velocity control loops work simultaneously. Thus, the impedance control can regulate the end-effector dynamics of the manipulator to the desired one described by (5), if the desired impedance parameters M_d, B_d, K_d are already given. However, it may be very difficult to design the desired impedance parameters according to the given task and the environment beforehand, so that M_d, B_d, K_d may be considered as unknown parameters. In this paper, the neural network model is used to represent those parameters and adjust them through iterative learning minimizing appropriate performance indices.

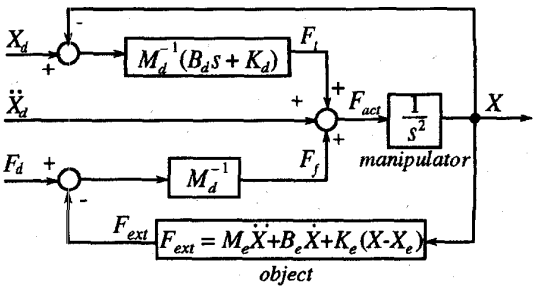


Fig. 1. Impedance control system in the operational space.

III. Iterative Learning of Impedance Parameters

A. Impedance Control Using Neural Networks

Generally, the back-propagation typed neural network model [12] has powerful learning ability and simple structure that are very attractive properties for a wide range of applications. The simple and uniform structure, however, frequently causes the local minima problem and/or very slow learning. One of the methods to improve these drawbacks is to introduce appropriate structure into the network according to each application, so that the whole problem to be learned can be divided into several parts. In this paper, the back-propagation typed neural network is structurally customized for the impedance control problem in the operational space.

Fig. 2 shows the impedance control system including two neural components. The first is the trajectory control network (TCN's) which corresponds to the impedance parameters $M_d^{-1}K_d$ and $M_d^{-1}B_d$, and the other is the force control network (FCN) which corresponds to M_d^{-1} (see Fig. 1).

Learning of the impedance networks proposed in this paper consists of two steps as shown in Fig. 3. First, the TCN's are trained using iterative learning of the manipulator during free movements to improve position control characteristics [see Fig. 3(a)]. Then, the FCN is trained for contact movements. During learning of contact movements, the TCN's are fixed and the virtual trajectory as well as the FCN are modified [see Fig. 3(b)].

B. Learning During Free Movements

Fig. 2(b) represents the structure of the TCN's. The TCN's include a position control network (PCN) and a velocity control network (VCN). Each network is of a multi-layered type with m input units and m^2 output units. The input units correspond to the end-effector position X, and the output units correspond to the impedance parameters $M_d^{-1}K_d$ for the PCN and $M_d^{-1}B_d$ for the VCN. It should be noted that the output signals of the PCN and VCN always represent the corresponding impedance parameters since the control system shown in Fig. 2(a) has the same structure as the one of Fig. 1.

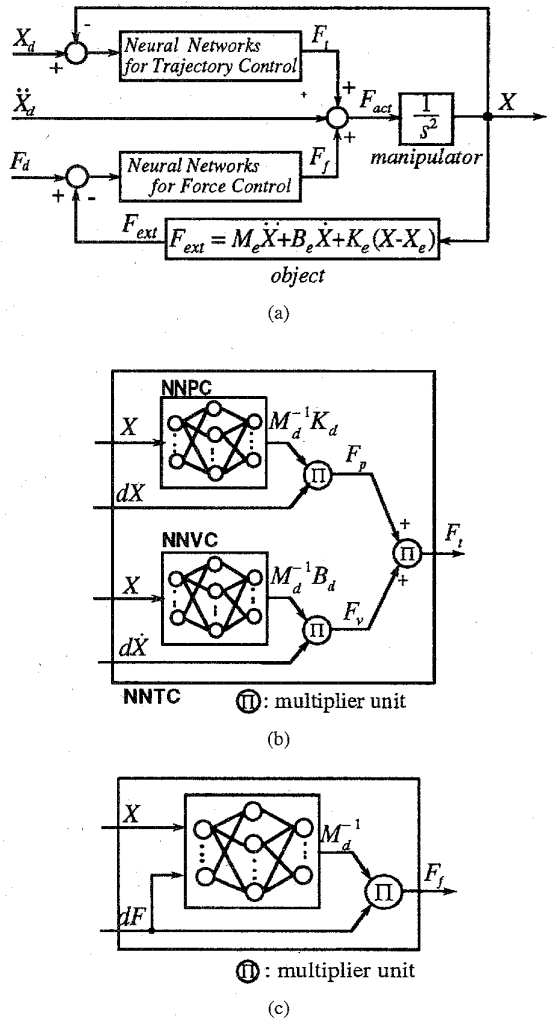


Fig. 2. Impedance control system based on the neural controller. (a) Block diagram of the control system. (b) Neural controller for trajectory control. (c) Neural controller for force control.

The sigmoidal activation functions are used for the hidden and output units, while linear functions are used for the input units, i.e.,

$$x_i = \begin{cases} I_i & \text{(input units)} \\ \sum y_i w_{ij} & \text{(hidden and output units)} \end{cases} \tag{7}$$

$$y_i = \begin{cases} x_i & \text{(input units)} \\ \dfrac{1}{1+e^{-x_i}} & \text{(hidden units)} \\ \dfrac{U}{1+e^{-x_i+\alpha}} & \text{(output units)} \end{cases} \tag{8}$$

where x_i and y_i represent input and output of unit i, respectively; w_{ij} is the synaptic weight from unit j to unit i; and U and α are the maximum value and threshold of the output units, respectively.

In this paper, the output of the PCN and VCN are denoted as vectors:

$$O_p = (o_{p1}^T, o_{p2}^T, \cdots, o_{pm}^T)^T \tag{9}$$

$$O_v = (o_{v1}^T, o_{v2}^T, \cdots, o_{vm}^T)^T \tag{10}$$

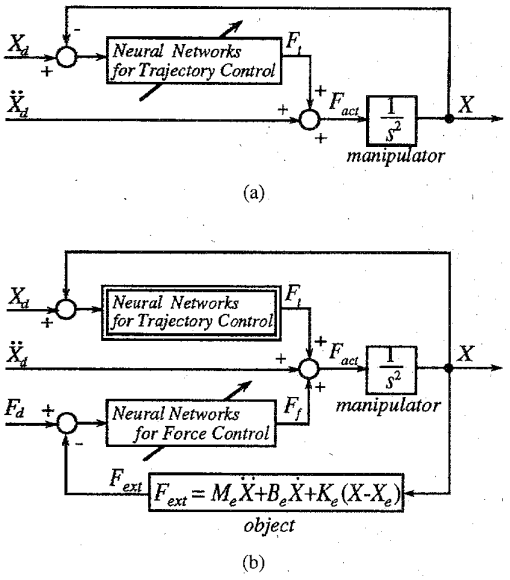


Fig. 3. Learning of impedance networks. (a) Learning during free movements. (b) Learning during contact movements.

where o_{pi} and $o_{vi} \in R^m$ are the vectors which consist of the output values of the PCN and VCN and correspond to the ith row of the matrices $M_d^{-1}K_d$ and $M_d^{-1}B_d$, respectively. Using these notations, the control force F_{act} during free movements is given as

$$\begin{aligned} F_{act} &= F_t + \ddot{X}_d = F_p + F_v + \ddot{X}_d \\ &= -\begin{bmatrix} o_{p1}^T \\ o_{p2}^T \\ \vdots \\ o_{pm}^T \end{bmatrix} dX - \begin{bmatrix} o_{v1}^T \\ o_{v2}^T \\ \vdots \\ o_{vm}^T \end{bmatrix} d\dot{X} + \ddot{X}_d \end{aligned} \tag{11}$$

where F_p and $F_v \in R^m$ are control vectors computed from the PCN and VCN, respectively (see Fig. 3).

An energy function for network learning of the TCN's is defined as

$$E_1 = \frac{1}{2} \sum_{t=0}^{N_f} \{E_p(t) + E_v(t)\} \tag{12}$$

where $E_p(t) = [X_d(t) - X(t)]^T[X_d(t) - X(t)]$ and $E_v(t) = [\dot{X}_d(t) - \dot{X}(t)]^T[\dot{X}_d(t) - \dot{X}(t)]$. $N_f = t_f/\Delta t$ denotes the number of data, where t_f is the final time of the data and Δt is a sampling interval. Then, the synaptic weights in the PCN and VCN, $w_{ij}^{(p)}$ and $w_{ij}^{(v)}$, are modified in the direction of the gradient descent as follows:

$$\begin{aligned} \Delta w_{ij}^{(p)} &= -\eta_p \frac{\partial E_1}{\partial w_{ij}^{(p)}} \\ \Delta w_{ij}^{(v)} &= -\eta_v \frac{\partial E_1}{\partial w_{ij}^{(v)}} \end{aligned} \tag{13}$$

$$\frac{\partial E_1}{\partial w_{ij}^{(p)}} = \frac{\partial E_1}{\partial F_p(t)} \frac{\partial F_p(t)}{\partial O_p(t)} \frac{\partial O_p(t)}{\partial w_{ij}^{(p)}} \tag{14}$$

$$\frac{\partial E_1}{\partial w_{ij}^{(v)}} = \frac{\partial E_1}{\partial F_v(t)} \frac{\partial F_v(t)}{\partial O_v(t)} \frac{\partial O_v(t)}{\partial w_{ij}^{(v)}} \tag{15}$$

where η_p and η_v are learning rates for the PCN and VCN, respectively. Except for the terms $\partial E_1/\partial F_p(t)$ and $\partial E_1/\partial F_v(t)$, all other terms in (14) and (15) can be computed by the back propagation learning. Since $\partial E_1/\partial F_p(t)$ and $\partial E_1/\partial F_v(t)$ cannot be obtained by back propagation learning because of the manipulator dynamics, the betterment process is used to approximate them [13].

In the betterment process, a time series of input signal to a plant is iteratively modified using an error signal between output and target signals so that the output signal at the next iteration approaches the desired one. Based on the PD-type betterment process, the control force at the $(k+1)$th iteration is defined as

$$F_{act}^{k+1}(t) = F_p^{k+1}(t) + F_v^{k+1}(t) + \ddot{X}_d(t) \tag{16}$$

$$F_p^{k+1}(t) = F_p^k(t) - \Gamma_p \, dX^k(t) \tag{17}$$

$$F_v^{k+1}(t) = F_v^k(t) - \Gamma_v \, d\dot{X}^k(t) \tag{18}$$

where Γ_p and $\Gamma_v \in R^{m \times m}$ represent gain matrices for position and velocity errors, respectively. Note that convergence of the betterment process is assured under appropriate gain matrices [13].

Paying attention to (17) and (18), it can be seen that the second terms give the directions of the control forces in order to decrease the error function E_1 at time t. Therefore, in this paper, the second terms are used as an approximation of the partial derivatives, $\partial E_1/\partial F_p(t)$ and $\partial E_1/\partial F_v(t)$:

$$\frac{\partial E_1}{\partial F_p(t)} \approx [\Gamma_p \, dX^k(t)]^T \tag{19}$$

$$\frac{\partial E_1}{\partial F_v(t)} \approx [\Gamma_v \, d\dot{X}^k(t)]^T. \tag{20}$$

The learning algorithm during free movements proposed in the paper is summarized as follows:

Step 0. Initial Values: Initial network weights, $w_{ij}^{(p)}$ for the PCN and $w_{ij}^{(v)}$ for the VCN, and a desired trajectory for the end-effector are given. The initial weights are randomly chosen and small values are recommended in order to avoid a large driving force of the manipulator in the early stage of the learning procedure.

Step 1. Impedance Control: Using neural networks, the impedance control is performed.

Step 2. Betterment Process: Using position and velocity error resulted in step 1, new control forces $F_p^{k+1}(t)$ and $F_v^{k+1}(t)$ are computed from (17) and (18). Also, error signals $dX(t)$ and $d\dot{X}(t)$ corresponding to the control input $F_{act}^{k+1}(t)$ are computed.

Step 3. Learning of the PCN and VCN: Using (13)–(15), (19), and (20), the synaptic weights $w_{ij}^{(p)}$ and $w_{ij}^{(v)}$ are modified until the learning error reaches the minimum (or a local minimum).

Until the error function E_1 reaches the minimum (usually a local minimum), the procedures from Steps 1–3 are executed iteratively. When the learning has terminated, the PCN and VCN may express the optimal impedance parameters $M_d^{-1}K_d$ and $M_d^{-1}B_d$ as the output values of the networks $O_p(t)$ and $O_v(t)$, respectively.

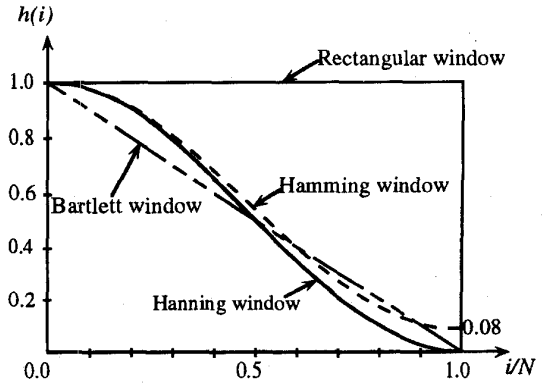


Fig. 4. Examples of the window function $h(i)$.

C. Learning During Contact Movements

After the learning of the TCN's, the network for force control is trained during contact movements. Fig. 2(c) shows the structure of the FCN. Note that the synaptic weights of the TCN's are fixed during learning of the FCN [see Fig. 3(b)].

The FCN is also a multi-layer typed network with $3m$ input units and m^2 output units. The input units correspond to the end-effector position, $X(t)$, and force control errors, $dF(t) = F_d(t) - F_{ext}(t)$ and $dF(t-1)$. The output units correspond to the impedance parameters M_d^{-1}. It should be noted that the output signal of the FCN always represents the corresponding impedance parameter since the control system shown in Fig. 2(a) has the same structure as the one of Fig. 1. The activation functions for the units are the same as (7) and (8).

The output of the FCN is denoted as a vector:

$$O_f = (o_{f1}^T, o_{f2}^T, \cdots, o_{fm}^T)^T \tag{21}$$

where $o_{fi} \in R^m$ is the vector which consists of the output values of the FCN and corresponds to the ith row of the matrix M_d^{-1}. From (6) and (11), the control force F_{act} during contact movements is given as

$$\begin{aligned} F_{act} &= F_t + F_f + \ddot{X}_d \\ &= -\begin{bmatrix} o_{p1}^T \\ o_{p2}^T \\ \vdots \\ o_{pm}^T \end{bmatrix} dX - \begin{bmatrix} o_{v1}^T \\ o_{v2}^T \\ \vdots \\ o_{vm}^T \end{bmatrix} d\dot{X} \\ &\quad + \begin{bmatrix} o_{f1}^T \\ o_{f2}^T \\ \vdots \\ o_{fm}^T \end{bmatrix} dF + \ddot{X}_d \end{aligned} \tag{22}$$

where $F_f \in R^m$ is the control vector computed from the FCN [see Fig. 2(c)] and $dF \in R^m$ is defined as $dF = F_d - F_{ext}$.

An energy function for learning of the FCN is defined as

$$E_2 = \frac{1}{2}\sum_{t=0}^{N_f} \tilde{E}_2(t) = \frac{1}{2}\sum_{t=0}^{N_f}\left\{\frac{1}{2}\sum_{i=-N}^{N} h^2(i)[E_p(t+i)+E_f(t+i)]\right\} \quad (23)$$

where $E_f(t+i) = dF(t+i)^T\, dF(t+i)$ and $E_p(t) = E_f(t) = 0$ for $t > N_f$ and $t < 0$. $h(i)$ is a data window function with the length of $2N+1$ that smoothes the time series of the error signal, and we can use one of the standard window functions [14]. Fig. 4 shows examples of $h(i)$. It should be noted that the window length $2N+1$ is chosen to be sufficiently smaller than the given total data length N_f.

The error function $\tilde{E}_2(t)$ represents a weighting sum of position and force errors from $t-N$ to $t+N$. Therefore, the error function E_2 in (23) includes the control errors within N unit time in the future from the time t, which works effectively for some tasks including a shift from free to contact motions. Generally speaking, any data in the future is not available because of the causality of the process. However, since the present paper uses an iterative learning scheme, the control errors after time t in the kth iteration can be used as the future error.

The direction of the gradient descent of the synaptic weights $w_{kl}^{(f)}$ for the error function E_2 is given as

$$\Delta w_{kl}^{(f)} = -\frac{\eta_f}{2}\sum_{i=-N}^{N} h^2(i)\frac{\partial E_3}{\partial F_{act}(t+i)} \cdot \frac{\partial F_{act}(t+i)}{\partial O_f(t+i)}\frac{\partial O_f(t+i)}{\partial w_{kl}^{(f)}} \quad (24)$$

where $E_3 = \frac{1}{2}\sum_{t=0}^{N_f}[E_p(t+i)+E_f(t+i)]$ and η_f is the learning rate. Except for the term $\partial E_3/\partial F_{act}(t+i)$, all other terms in (24) can be computed using the back propagation learning. For $\partial E_3/\partial F_{act}(t+i)$, the betterment process is used in the same way as the TCN.

Here, it is assumed that the total data length N_f is sufficiently larger than the length of the data window $2N+1$. Under this assumption, the error function E_3 in (24) can be approximated as

$$\frac{1}{2}\sum_{t=0}^{N_f}[E_p(t+i)+E_f(t+i)] \approx \frac{1}{2}\sum_{t+i=0}^{N_f}[E_p(t+i)+E_f(t+i)]. \quad (25)$$

It can be seen that the direction of the gradient descent of the right hand side of (25) is approximated by the betterment process. Based on the PD-type betterment process, the direction of the gradient descent at the $(k+1)$th iteration can be defined as

$$\frac{\partial E_3}{\partial F_{act}(t+i)} \approx \{\Gamma_p\, dX^k(t+i) - [\Gamma_f\, dF^k(t+i) + \Phi_f\, d\dot{F}^k(t+i)]\}^T \quad (26)$$

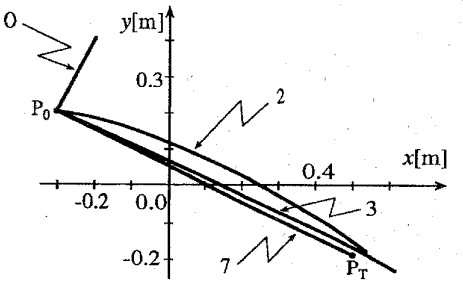


Fig. 5. End-effector trajectories of the manipulator during learning of a free movement.

where Γ_p, Γ_f, $\Phi_f \in R^{m\times m}$ represent gain matrices for position error, force error, and force error derivative, respectively.

Using (24) and (26), the FCN could minimize the error function E_2. However, since (26) contains force error derivative $d\dot{F}$, it may be very difficult to decrease this error term by only the FCN learning. Then, during learning of the contact movement, the virtual trajectory is also modified numerically based on the error back propagation method. This means that the input signals to the system, $X_d(t)$ and $\dot{X}_d(t)$, are modified to reduce an output force control error instead of learning of TCN's during learning of the contact movement. The learning rule of the virtual trajectory can be derived in the same way as the FCN. Because the modification of the virtual trajectory makes the position error $E_p(t+i)$ insignificant, the error function for learning of the virtual trajectory is modified as

$$E_4 = \sum_{t=0}^{N_f}\left\{\frac{1}{2}\sum_{i=-N}^{N} h^2(i)E_f(t+i)\right\}. \quad (27)$$

In this case, the direction of the gradient descent for the error function is given by

$$\Delta X_d(t) = -\eta_d\left[\frac{\partial E_4}{\partial X_d(t)}\right]^T \quad (28)$$

$$\frac{\partial E_4}{\partial X_d(t)} = \sum_{i=-N}^{N} h^2(t)\frac{\partial E_5}{\partial F_{act}(t+i)} \cdot \frac{\partial F_{act}(t+i)}{\partial X_d(t+i)}\frac{\partial X_d(t+i)}{\partial x_d(t)} \quad (29)$$

where $E_5 = \frac{1}{2}\sum_{t=0}^{N_f} E_f(t+i)$. It could be assumed that the modification of $X_d(t)$ is almost constant in the period from $t-N$ to $t+N$, since the data window $h(i)$ behaves like a smoothing filter. In this case, $\partial X_d(t+i)/\partial X_d(t)$ reduces to the unit matrix and the other terms in (29) can be computed using the betterment process and error back propagation method. Under this learning strategy, the virtual trajectory X_v can be modified according to the characteristics of the given environment (2) and the desired force F_d, even if the virtual trajectory is originally planned for a free movement without any consideration on the contact movement. It should be noted that the learning rule for $\dot{X}_d(t)$ can be derived in the same way as $X_d(t)$.

A learning algorithm during contact movements proposed in the paper is summarized as follows:

Step 0. Initial Values: The initial network weights $w_{ij}^{(f)}$ for the FCN and a desired end-effector force are given.

Step 1. Impedance Control: Using the TCN's and the initial virtual trajectory, the impedance control is performed.

Step 2. Betterment Process: Using the control errors resulted from step 1, the new control force $F_{act}^{k+1}(t)$ is computed. Also, error signals $dX(t)$, $dF(t)$, and $d\dot{F}(t)$ corresponding to the control input $F_{act}^{k+1}(t)$ are computed.

Step 3. Learning of the FCN and the Virtual Trajectory: Using (24) and (29), the synaptic weights, $w_{ij}^{(f)}$, and the virtual trajectories, $X_d(t)$ and $\dot{X}_d(t)$, are modified until the learning error reaches the minimum (or a local minimum).

The procedures from Steps 1–3 are executed iteratively. Because this learning algorithm is based on the steepest descent method, the error function E_2 (23) reaches the minimum (usually a local minimum) if an appropriate learning rate is used. When the learning has terminated, the FCN may express the optimal impedance parameters M_d^{-1} as the output values of the networks $O_f(t)$. It should be noted that the output signal of the FCN always represents the corresponding impedance parameter since the control system shown in Fig. 2 has the same structure as the one in Fig. 1.

IV. Simulation Experiments

In order to confirm the effectiveness of the method proposed in this paper, a series of computer simulations on planar movements ($m = 2$) is performed under an assumption that the end-effector dynamics of the manipulator has already been simplified as given by (3).

A. Free Movements

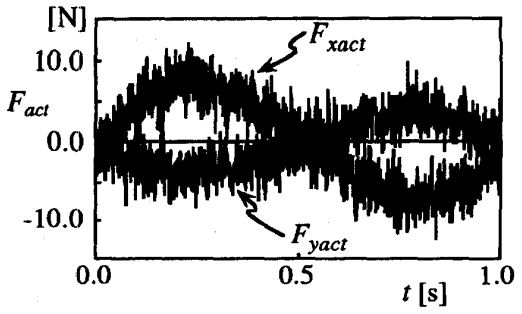
First, learning of a free movement is performed. The PCN and VCN used in the simulations are of three layered networks with two input units, ten hidden units and four output units. The initial values of the synaptic weights are randomly chosen from $|w_{ij}^{(p)}|$, $|w_{ij}^{(v)}| < 0.25$, and the white Gaussian signal (mean value, 0.0 [N]: standard deviation, 1.0 [N]) is added to the control input F_{act} in order to simulate noise from the environment. Also the following parameters are used in the simulations: $\theta = 6.0$ and $U = 500.0$ in (8), $\eta_p = 2.0 \times 10^{-4}$ and $\eta_v = 5.0 \times 10^{-5}$ in (13) and Γ_p and Γ_v in (19) are determined as follows [8]:

$$\Gamma_p = \tfrac{1}{2}\,\mathrm{diag}\left\{\min_t [o_{p11}(t)],\ \min_t [o_{p22}(t)]\right\} \tag{30}$$

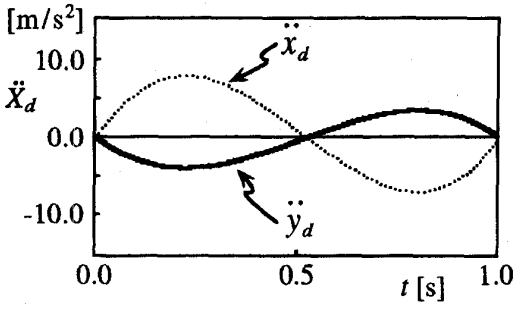
$$\Gamma_v = \tfrac{1}{2}\,\mathrm{diag}\left\{\min_t [o_{v11}(t)],\ \min_t [o_{v22}(t)]\right\} \tag{31}$$

where diag[] denotes a diagonal matrix. The topological structure of the networks and the learning parameters used in this simulation are determined after trial and error.

Fig. 5 shows changes of the end-effector trajectory by learning, where P_0 and P_T in the figure denote the initial and target positions, respectively, and the number in the figure denotes the iteration of the betterment process. Each iteration of the betterment process includes 100 times of the error back propagation learning of the TCN's. The desired trajectory of the end-effector is determined using the fifth-order polynomial under the boundary conditions,



(a)



(b)

Fig. 6. Comparison between control input F_{act} and the desired acceleration. (a) F_{act} after seven iterations of learning process during free movements. (b) $\ddot{X}_d$.

$$X_d(0) = [x_d(0),\ y_d(0)]^t = (-0.3,\ 0.2)^T\ [\mathrm{m}] \tag{32}$$

$$X_d(t_f) = [x_d(t_f),\ y_d(t_f)]^T = (0.5,\ -0.2)^T\ [\mathrm{m}] \tag{33}$$

$$\dot{X}_d(0) = (0,\ 0)^T\ [\mathrm{m/sec}]$$

$$\dot{X}_d(t_f) = (0,\ 0)^T\ [\mathrm{m/sec}] \tag{34}$$

$$\ddot{X}_d(0) = (0,\ 0)^T\ [\mathrm{m/sec^2}]$$

$$\ddot{X}_d(t_f) = (0,\ 0)^T\ [\mathrm{m/sec^2}] \tag{35}$$

where $t_f = 1.0$ [s], $\Delta t = 0.001$ [s], and $N_f = 1000$. From the figure, it can be seen that the end-effector trajectory coincides with the desired one after several iterations. It should be noted that local minima of the error function did not pose a serious problem during learning. This might be expected the effect of the white Gaussian signal added to the control input.

Also Fig. 6 shows the control input F_{act} and the desired acceleration $\ddot{X}_d$ after seven iterations of learning process during free movements. Note that F_{act} includes the white Gaussian signal. It can be seen that the time history of F_{act} without a noise component almost coincides with $\ddot{X}_d$.

Table I shows the impedance parameters, $M_d^{-1}K_d$ and $M_d^{-1}B_d$, before and after learning. $E[M_d^{-1}K_d]$ and $E[M_d^{-1}B_d]$ in the table represent time averages of the

TABLE I
IMPEDANCE PARAMETERS BEFORE AND AFTER LEARNING OF A FREE MOVEMENT

	$E[M_d^{-1}K_d]$	$E[M_d^{-1}B_d]$
before learning	$\begin{bmatrix} 1.58855 & 1.14292 \\ 1.00605 & 1.08828 \end{bmatrix}$	$\begin{bmatrix} 1.24700 & 1.62821 \\ 1.40561 & 1.25643 \end{bmatrix}$
after learning (10 trials)	$\begin{bmatrix} 495.19279 & 0.44871 \\ 0.44652 & 475.05515 \end{bmatrix}$	$\begin{bmatrix} 497.59935 & 0.41674 \\ 0.56942 & 494.30498 \end{bmatrix}$

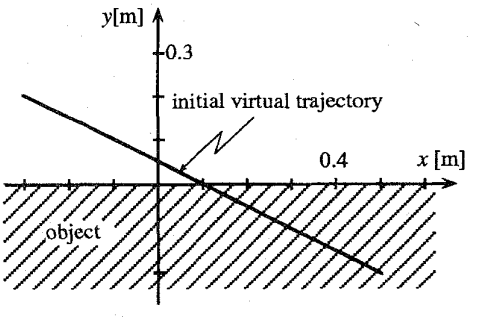


Fig. 7. Initial virtual trajectory and the object placed along the x axis of the task space.

corresponding impedance parameters:

$$E[M_d^{-1}K_d] = \frac{1}{1000}\sum_{t=0}^{1000} M_d^{-1}K_d(t) \tag{36}$$

$$E[M_d^{-1}B_d] = \frac{1}{1000}\sum_{t=0}^{1000} M_d^{-1}B_d(t). \tag{37}$$

It can be seen from the table that the diagonal elements of the impedance matrices increase significantly, and the appropriate impedance matrices for trajectory control during free movements are realized after ten iterations. It should be noted that the maximum value of each output of the networks is determined by $U = 500.0$ in (8).

B. Contact Movements

Next, learning of a contact movement is performed for the task environment shown in Fig. 7, where an object is placed along the x axis of Fig. 6. The dynamics of the object is characterized as (2) where $M_e = \text{diag}\,[0.0, 0.0]$ [kg], $B_e = \text{diag}\,[0.0, 10.0]$ [N·s/m], and $K_e = \text{diag}\,[0.0, 1.0 \times 10^5]$ [N/m].

The FCN used in the simulation is of three layered networks with six input units, ten hidden units and four output units. The experimental conditions are the same as learning of TCN's except that $\eta_f = 6.43 \times 10^{-11}$ in (24), $\eta_d = 6.0 \times 10^{-8}$ in (28) and Γ_f and Φ_f in (26) are determined as follows:

$$\Gamma_f = \tfrac{1}{2}\,\text{diag}\left\{\min_t [o_{f11}(t)],\ \min_t [o_{f22}(t)]\right\}, \tag{38}$$

$$\Phi_f = \tfrac{1}{1000}\,\Gamma_f. \tag{39}$$

Fig. 8 shows changes of the end-effector forces by learning, where the Hanning window is used as the window function $h(i)$ in (24) with $N = 100$ (i.e., 0.1 [s]); and the desired end-effector force $F_d = (0.0, -100.0)^T$ [N]. Because the manipulator tries to follow the initial virtual trajectory and the impedance parameters learned during free movements are considerably large before learning of the contact movements (see Table I), a large interaction force is exerted along the direction of the y axis in the task space (see iteration number 0 in Fig. 8). After thirty iterations, however, the end-effector force coincides with the desired one.

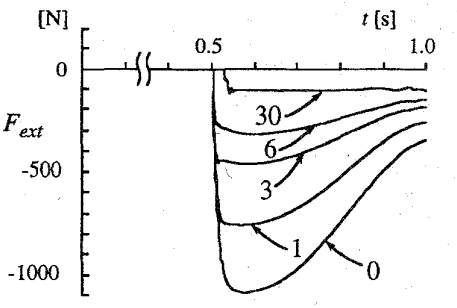


Fig. 8. End-effector forces of the manipulator in the normal direction of the object during learning of a contact movement ($N = 100$).

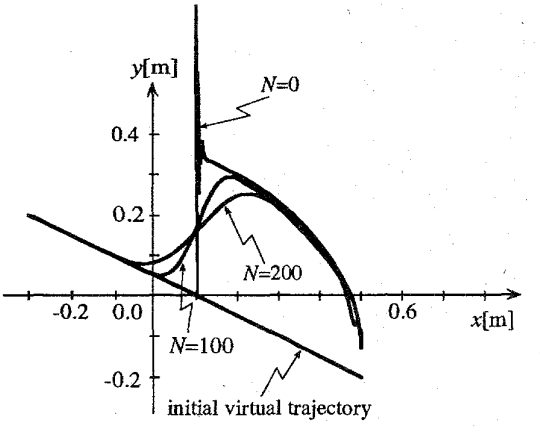


Fig. 9. Virtual trajectories learned during contact movements ($N = 0$, 100, 200).

Fig. 9 shows the effect of the window length N on the learned virtual trajectories, X_d, which are obtained after thirty iterations. In all cases, the end-effector force almost coincides with the desired one. However, the learned virtual trajectories are quite different. For the case of $N = 0$, the virtual trajectory changes suddenly just after contact with the object in order to absorb the impact force due to the contact. On the other hand, as the window length becomes long, the virtual trajectories change before the contact so that a smooth transition from free to contact movements can be realized.

Table II shows impedance parameters before and after learning of the contact movement. $E[K_d]$, $E[B_d]$, and $E[M_d]$ in the table are computed as follows:

$$E[M_d] = \frac{1}{N_f - N_c}\sum_{t=N_c}^{N_f} [M_d^{-1}(t)]^{-1}, \tag{40}$$

$$E[B_d] = \frac{1}{N_f - N_c}\sum_{t=N_c}^{N_f} [M_d^{-1}(t)]^{-1}\,[M_d^{-1}B_d(t)], \tag{41}$$

TABLE II
IMPEDANCE PARAMETERS BEFORE AND AFTER LEARNING OF A CONTACT MOVEMENT ($N = 100$)

	$E[M_d]$	$E[B_d]$	$E[K_d]$
before learning	3.64677 -1.81733 -4.53264 4.88745	1816.51098 -899.88635 -2256.31941 2422.31156	1802.87264 -858.30643 -2239.63566 2310.56670
after learning (30 trials)	2.40574 -0.10942 -0.30697 0.27765	1198.88872 -53.43823 -152.88977 137.68601	1190.01721 -50.52533 -151.66848 131.14677

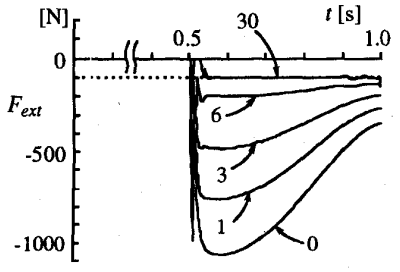


Fig. 10. End-effector forces of the manipulator in the normal direction of the object, where an impact force is included.

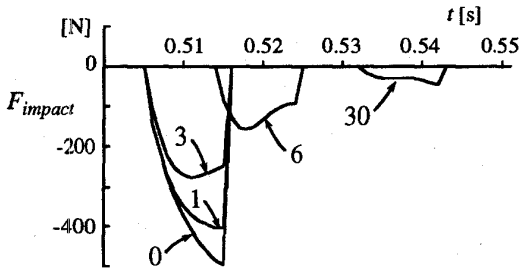


Fig. 11. Changes of the impact force F_{impact} during learning of a contact movement.

$$E[K_d] = \frac{1}{N_f - N_c} \sum_{t=N_c}^{N_f} [M_d^{-1}(t)]^{-1}[M_d^{-1} K_d(t)], \quad (42)$$

where N_c denotes the contact time with the object. After thirty iterations, the impedance parameters including the stiffness, viscosity and inertia become considerably small in the normal direction of the contact surface, i.e., the y axis of the task space, when compared with the ones in the tangential direction. The impedance parameters can be regulated according to the task environment by the iterative learning.

Finally, the dynamics of the object is changed including object inertia and friction not only in the normal direction but the tangential direction: M_e =diag [0.2, 0.2] [kg], B_e = diag [10.0, 10.0] [N·s/m], and K_e = diag [0.0, 1.0×10^5] [N/m]. Also, in order to represent a collision between the end-effector and the object, a simple impact model is used [15]:

$$F'_{ext} = F_{ext} + F_{impact} \quad (43)$$

where $F'_{ext} \in R^2$ represents the interaction force between the end-effector and the object; and F_{ext} is defined by (2). $F_{impact} \in R^2$ represents the time-varying impact force and is defined as

$$F_{impact} = \begin{cases} K_{impact}[X(t) - X_e] & t_c \le t \le t_c + \delta t \\ 0 & \text{all other } t \end{cases} \quad (44)$$

where $K_{impact} \in R^2$ is the impact stiffness matrix, t_c is the collision time and Δt is the time duration of the impact force. In this simulation, K_{impact} = diag [0.0, 1.0×10^5] [N/m] and $\delta t = 0.01$ [s] are used.

Fig. 10 shows changes of the end-effector forces by learning, where the topological structure of the networks and the parameters such as η_f, η_d, Γ_f, Φ_f, and N used in this computer simulation are the same as the ones used in Fig. 8. Although a sharp impact force just after a collision is observed, a steady-state force almost coincides with the desired one after thirty iterations.

Fig. 11 shows changes of the impact forces F_{impact} by learning. Although the impact force is quite large before learning of the contact movements, it is decreased considerably after thirty iterations. It should be noted that the collision time t_c is delayed as shown in the figure. This means that the end-effector velocity just before the contact is decreased in order to avoid large impact force between the end-effector and the object.

V. CONCLUSIONS

The present paper proposed the new method to regulate the impedance parameters of the end-effector using neural networks. The method can regulate the second order impedance including the stiffness, viscosity and inertia through iterative learning to minimize the position and force control errors. Introducing the window function into the error functions, the virtual trajectory as well as the impedance parameters can be modified before contact so that a smooth transition from free to contact movements is realized. It should be noted that the impedance parameter obtained through learning in this method is not always a unique solution. Since any direct teaching signal for the impedance parameter is not available, the networks used in the proposed method are trained to minimize a position control error and/or a force control error instead of an impedance error. In this sense, a unique solution of the impedance parameter is not needed, since the impedance parameters learned in the networks can be assured to be optimal or sub optimal in terms of the error functions.

Future research will be directed to improvements of learning rule for the neural networks used in this paper and a generalizing ability of the proposed method for various classes of the constrained tasks.

ACKNOWLEDGMENT

The authors wish to thank Prof. M. Kaneko of Hiroshima University, and Dr. A. Jazidie of Surabaya Institute of Technology for their kind help in this work. Thanks also to Mr. M Nishida for the development of the computer programs.

REFERENCES

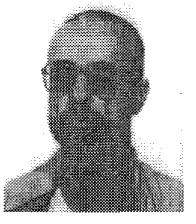
[1] N. Hogan, "Impedance control: An approach to manipulation: Parts I, II, and III," *ASME J. Dyn. Syst., Meas., and Cont.,* vol. 107, no. 1, pp. 1–24, 1985.
[2] H. Gomi and M. Kawato, "Neural network control for a closed loop system using feedback-error-learning," *Neural Networks,* vol. 6, no. 7, pp. 933–946, 1993.
[3] S. T. Venkataraman, S. Glati, J. Barhen, and N. Toomarian, "A neural network based identification of environments models for compliant control of space robots," *IEEE Trans. Robot. Automat.,* vol. 9, no. 5, pp. 685–697, Oct. 1993.
[4] Y. Maeda and H. Kano, "Learning control for impedance controlled manipulator," in *Proc. 31st IEEE Conf. Decision and Control,* 1992, pp. 3135–3140.
[5] C. C. Cheah and D. Wang, "Learning impedance control of robotic manipulators," in *Proc. Third Int. Conf. Automation, Robotics and Computer Vision,* 1994, pp. 227–231.
[6] H. Asada, "Teaching and learning of compliance using neural nets: representation of generation of nonlinear compliance," in *Proc. IEEE Int. Conf. Robotics and Automation,* 1990, pp. 1237–1244.
[7] M. Cohen and T. Flash, "Learning impedance parameters for robot control using associative search network," *IEEE Trans. Robot. Automat.,* vol. 7, no. 3, pp. 382–390, May 1991.
[8] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural network model for control and learning of voluntary movement," *Biolog. Cybern.,* vol. 57, pp. 169–185, 1987.
[9] J. K. Mills, "Stability of robotic manipulators during transition to and from compliant motion," *Automatica,* vol. 26, no. 5, pp. 861–874, 1990.
[10] Z. Luo and M. Ito, "Control design of robot for compliant manipulation on dynamic environment," in *Proc. IEEE Int. Conf. Robotics and Automation,* 1991, pp. 42–47.
[11] N. Hogan, "An organizing principle for a class of voluntary movements," *J. Neuroscience,* vol. 4, pp. 2745–2754, Nov. 1984.
[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing, Vol. 1: Foundations,* D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Eds. Cambridge,MA: MIT Press, 1986, pp. 318–362.
[13] S. Kawamura, F. Miyazaki, and S. Arimoto, "Realization of robot motion based on a learning method," *IEEE Trans. Syst., Man., Cybern.,* vol. 18, no. 1, pp. 126–134, Jan. 1988.
[14] G. M. Jenkins and D. G. Watts, *Spectral Analysis and Its Application.* New York: Holden-Day, 1968, p. 244.
[15] J. K. Mills, "Manipulator transition to and from contact tasks: a discontinuous control approach," in *Proc. IEEE Int. Conf. Robotics and Automation,* 1990, pp. 440–446.

Toshio Tsuji (A'88) was born in 1959. He received the B.E. degree in industrial engineering, and the M.E. and Doctor of Engineering degrees in system engineering in 1982, 1985, and 1989, respectively, all from Hiroshima University, Hiroshima, Japan.

He was a research associate on the Faculty of Engineering, Hiroshima University since 1985, and is now an Associate Professor. He has been interested in various aspects of motor control in robot and human movements. His current research interests have focused on distributed planning and learning of motor coordination. He was a visiting researcher of University of Genova, Italy, for the academic year 1992–1993.

Koji Ito (M'87) was born in 1944. He received the B.S. degree from Nagoya Institute of Technology in 1967, the M.S. degree in 1969, and the Dr.Eng. degree in 1976 from Nagoya University, Nagoya, Japan.

From 1970 to 1979, he was a Research Assistant at the Automatic Control Laboratory, Faculty of Engineering, Nagoya University. From 1979 to 1992, he was an Associate Professor, Department of Computer and System Engineering, Hiroshima University. Since 1992, he has been a professor in the Department on Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Japan. Since 1993, he has also held an additional post of head, Laboratory for the Bio-Mimetic Control Systems at RIKEN, Nagoya. His main research interests are in the design and control of robotics and prostheses, and computational neural sciences, in particular, biological motor control.



Pietro G. Morasso earned in the Master Degree in electronic engineering from the University of Genova, Italy, in 1968, with a thesis on biomedical signal processing. He was post-doctoral fellow in the Department of Psychology, Massachusetts Institute of Technology, Cambridge, from 1970 to 1972, where he worked on problems of motor control with humans and primates.

Since 1986, he has been a full Professor of Anthropomorphic Robotics, Department of Informatics, Systems, and Telecommunication, University of Genova. His interests include motor planning and control in biological and robotic systems with emphasis on neural network models.