IJCNN'93-NAGOYA

PROCEEDINGS OF 1993 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS

VOLUME 1
OF 3

NAGOYA CONGRESS CENTER
OCTOBER 25-29, 1993, JAPAN

Co-sponsored by


Japanese Neural Network Society (JNNS)


IEEE Neural Networks Council (NNC)


International Neural Network Society (INNS)

ENNS *European Neural Network Society (ENNS)*


The Society of Instrument and Control Engineers (SICE)


The Institute of Electronics, Information and Communication Engineers (IEICE)

Chubu Bureau of International Trade and Industry

Aichi Prefecture

City of Nagoya

Nagoya Chamber of Commerce and Industry

Chubu Economic Federation

Nagoya Industrial Science Research Institute

The Chubu Industrial Advancement Center

Proceedings of 1993 International Joint Conference on Neural Networks

Learning of Robot Arm Impedance in Operational Space Using Neural Networks

Toshio TSUJI[1], Koji ITO[2] and Pietro MORASSO[3]
[1]Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1-chome, Higashi-Hiroshima, 724 Japan
[2]Toyohashi University of Technology
1-1, Hibarigaoka, Tempaku-cho, Toyohashi, 440 Japan
[3]Department of Informatics, Systems and Telecommunications, University of Genova
Via Opera Pia 11A, 16145 Genova, Italy

Abstract : Impedance control is one of the most effective control methods for the manipulators in contact with their environments. The characteristic of force and motion control, however, is influenced by a desired impedance of a manipulator's end-effector, which must be designed according to a given task and an environment. The present paper proposes a new method to regulate the impedance of the end-effector through learning of neural networks. The method can regulate not only stiffness and viscosity but also inertia and virtual trajectory of the end-effector, and can realize a smooth transition from free to contact movements by regulating impedance parameters before a contact.

1 Introduction

Impedance control can regulate end-effector dynamics of the manipulator to the desired one, and give us a unified approach for force and motion control (Hogan, 1985). The characteristics of control system, however, are influenced by desired impedance that must be planned according to a task and an environment. Learning by neural networks is one of possible approaches to adjust the impedance skillfully.

Recently, a few investigations that apply the neural network learning into the impedance control have been reported. Asada(1990) showed that nonlinear viscosity could be realized by using the neural network model as a force feedback controller. Cohen and Flash(1991) also proposed a method to regulate the end-effector stiffness and viscosity. Although the stiffness and/or viscosity were learned in their models, an inertia property couldn't be regulated.

The present paper proposes a new method to regulate the desired impedance through learning of neural networks. Three kinds of the error back propagation typed networks are prepared corresponding to the position, velocity and force control. First, the networks for position and velocity control are trained using iterative learning during free movements. Then, the network for force control is trained for contact movements. During contact movements, the virtual trajectory is also modified to reduce control error.

2.Impedance Control

A motion equation of an n-joint manipulator in contact with an environment is given as

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

where θ and $\tau \in R^n$ represent a joint angle and a joint torque, respectively; $M(\theta)\in R^{n\times n}$ is an inertia tensor; $h(\theta,\dot{\theta})\in R^n$ is centrifugal and Coriolis forces; $g(\theta)\in R^n$ is a gravitational torque; $J(\theta)\in R^{m\times n}$ is a Jacobean matrix; $F_{int}\in R^m$ is an external force exerted from the environment to the end-effector. m is a number of degrees of freedom of an operational space. In this paper, a characteristic of the environment is modeled as

$$F_{int}=M_e\ddot{X}+B_e\dot{X}+K_e(X-X_e), \quad (2)$$

where $X\in R^m$ is an end-effector position; $X_e\in R^m$ is an equilibrium position of the environment; and M_e, B_e and $K_e\in R^{m\times m}$ represent inertia, viscosity and stiffness of the environment, respectively.

Using a nonlinear compensation such as

$$\tau=h(\theta,\dot{\theta})+g(\theta)+J^T(\theta)F_{int}+M(\theta)J^{-1}(\theta)\left[F_{act}-\dot{J}(\theta)\dot{\theta}\right], \quad (3)$$

the dynamics of the manipulator in the operational space reduce to a simplified equation(Luo and Ito,1990):

$$\ddot{X}=F_{act}, \quad (4)$$

where $F_{act}\in R^m$ is a control force represented in the operational space. Also, a second order linear model is used as a desired end-effector impedance:

$$M_d d\ddot{X}+B_d d\dot{X}+K_d X=F_d-F_{int}, \quad (5)$$

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

$$F_{act}=-M_d^{-1}(B_d d\dot{X}+K_d dX)+M_d^{-1}(F_d-F_{int})+\ddot{X}_d. \quad (6)$$

Fig.1 shows a block diagram of the impedance control (Luo and Ito,1990). During free movements, the force feedback loop doesn't exist because of $F_d=F_{int}=0$.

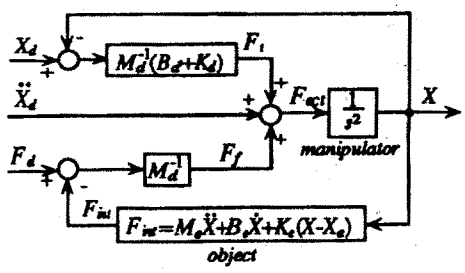

Fig.1 Impedance control in the operational space.

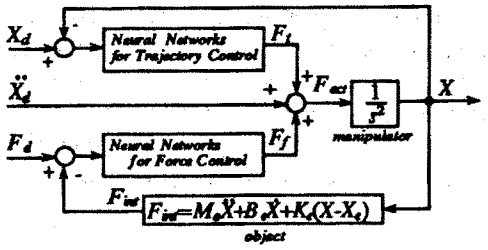

Fig.2 Impedance control using neural networks.

3.Iterative Learning of Impedance Parameters

3.1 Impedance control using neural networks

Fig.2 shows the impedance control system proposed in this paper, which includes two neural components. The one is the trajectory control network (TCN) that corresponds to $M_d^{-1}K_d$ and $M_d^{-1}B_d$, and the other is the force control network (FCN) corresponding to M_d^{-1} (see Fig.1).

3.2 Learning during free movements

Fig.3 represents a structure of the TCN. The position control network (PCN) and the velocity control network (VCN) are multi-layered networks with m input units and m^2 output units. The input units represent the end-effector position X, and the output units represent $M_d^{-1}K_d$ for the PCN and $M_d^{-1}B_d$ for the VCN. The linear activation functions are used for input units and the sigmoid ones are used for hidden and output units.

The outputs of the PCN and VCN are denoted as $O_p = (o_{p1}^T, o_{p2}^T, \cdots, o_{pm}^T)^T$ and $O_v = (o_{v1}^T, o_{v2}^T, \cdots, o_{vm}^T)^T$,

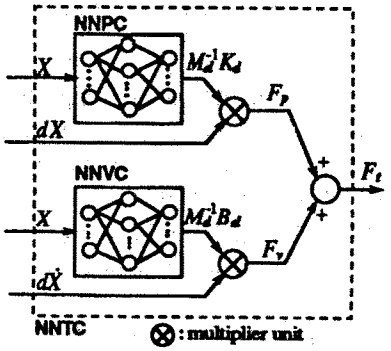

Fig.3 Neural networks for trajectory control, TCN.

where o_{pi} and $o_{vi} \in \mathrm{R}^m$ are the vectors that correspond to the i-th row of $M_d^{-1}K_d$ and $M_d^{-1}B_d$, respectively. Using these notations, the control force F_{act} during free movements is given as

$$F_{act} = F_p + F_v + \ddot{X}_d = -\begin{bmatrix} o_{p1}^T \\ o_{p2}^T \\ \vdots \\ o_{pm}^T \end{bmatrix} dX - \begin{bmatrix} o_{v1}^T \\ o_{v2}^T \\ \vdots \\ o_{vm}^T \end{bmatrix} d\dot{X} + \ddot{X}_d , \quad (7)$$

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

An energy function for the TCN is defined as

$$E_1 = \frac{1}{2}\sum_{t=0}^{N_f} \{E_p(t) + E_v(t)\} , \quad (8)$$

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 a number of data, where t_f is a final time 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:

$$\Delta w_{ij}^{(p)} = -\eta_p \frac{\partial E_1}{\partial w_{ij}^{(p)}} , \quad \Delta w_{ij}^{(v)} = -\eta_v \frac{\partial E_1}{\partial w_{ij}^{(v)}} , \quad (9)$$

$$\frac{\partial E_1}{\partial w_{ij}^{(p)}} = \frac{\partial E_1}{\partial F_p(t)} \frac{\partial F_p(t)}{\partial O_p(t)} \frac{\partial O_p(t)}{\partial w_{ij}^{(p)}} , \quad (10)$$

$$\frac{\partial E_1}{\partial w_{ij}^{(v)}} = \frac{\partial E_1}{\partial F_v(t)} \frac{\partial F_v(t)}{\partial O_v(t)} \frac{\partial O_v(t)}{\partial w_{ij}^{(v)}} , \quad (11)$$

where η_p and η_v are learning rates. Except for the terms $\partial E_1 / \partial F_p(t)$ and $\partial E_1 / \partial F_v(t)$, all other terms can be computed by the error back propagation. Since $\partial E_1 / \partial F_p(t)$ and $\partial E_1 / \partial F_v(t)$ cannot be obtained directly because of the manipulator dynamics, a betterment process (Kawamura, Miyazaki and Arimoto, 1986) is used to approximate them.

In the betterment process, a time series of input signal is iteratively modified using an error signal, so that an output signal at the next iteration approaches to the desired one. Using a DP typed betterment process, a control force at the $(k+1)$-th iteration is defined as

$$F_{act}^{k+1}(t) = F_p^{k+1}(t) + F_v^{k+1}(t) + \ddot{X}_d(t) , \quad (12)$$

$$F_p^{k+1}(t) = F_p^k(t) - \Gamma_p dX^k(t) , \quad (13)$$

$$F_v^{k+1}(t) = F_v^k(t) - \Gamma_v d\dot{X}^k(t) , \quad (14)$$

where Γ_p and $\Gamma_v \in \mathrm{R}^{m \times m}$ represent gain matrices. Note that convergency of the betterment process is assured under appropriate gain matrices(Kawamura, Miyazaki and Arimoto, 1986). From (13) and (14), 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, the second terms are used as the partial derivatives, $\partial E_1 / \partial F_p(t)$ and $\partial E_1 / \partial F_v(t)$:

$$\frac{\partial E_1}{\partial F_p(t)} \equiv \left[\Gamma_p dX^k(t)\right]^T \text{ and } \frac{\partial E_1}{\partial F_v(t)} \equiv \left[\Gamma_v d\dot{X}^k(t)\right]^T . \quad (15)$$

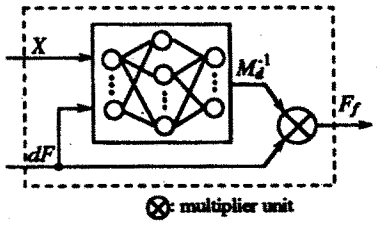

Fig.4 Neural network for force control, FCN.

3.3 Learning during contact movements

Next, the FCN is trained during contact movements. Fig.4 shows a structure of the FCN. The input units represent the end-effector position $X(t)$ and force control errors $dF(t) = F_d(t) - F_{ext}(t)$ and $dF(t\text{-}1)$. Also, the output units represent M_d^{-1}.

The output of the FCN is denoted as $O_f = (o_{f1}^T, o_{f2}^T, \cdots, o_{fm}^T)^T$, where $o_{fi} \in \mathrm{R}^m$ is the vector that corresponds to the i-th row of the matrix M_d^{-1}. From (6) and (7), the control force F_{act} is given as

$$F_{act} = -\begin{bmatrix} o_{p1}^T \\ o_{p2}^T \\ \vdots \\ o_{pm}^T \end{bmatrix} dX - \begin{bmatrix} o_{v1}^T \\ o_{v2}^T \\ \vdots \\ o_{vm}^T \end{bmatrix} d\dot{X} + \begin{bmatrix} o_{f1}^T \\ o_{f2}^T \\ \vdots \\ o_{fm}^T \end{bmatrix} dF + \ddot{X}_d . \quad (16)$$

Also, an energy function is defined as

$$E_2 = \frac{1}{2}\sum_{t=0}^{N_f} \tilde{E}_2(t) = \frac{1}{2}\sum_{t=0}^{N_f} \left\{ \frac{1}{2}\sum_{i=-N}^{N} h^2(i)\left[E_p(t+i)+E_f(t+i)\right] \right\} \quad (17)$$

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. The error function $\tilde{E}_2(t)$ is a weighting sum of position and force errors from t-N to t+N. Therefore it includes the control errors within N unit time in the future. Since the present paper uses the iterative learning scheme, the control errors after time t in k-th iteration can be used for the (k+1)-th iteration.

The direction of the gradient descent of the synaptic weights $w_{ij}^{(l)}$ is given as

$$\Delta w_{ij}^{(l)} = -\eta_f \sum_{i=-N}^{N} h^2(i) \frac{\partial E_3}{\partial F_{act}(t+i)} \frac{\partial F_{act}(t+i)}{\partial O_f(t+i)} \frac{\partial O_f(t+i)}{\partial w_{ij}^{(l)}}, \quad (18)$$

where $E_3 = \frac{1}{2}\sum_{t=0}^{N_f} \left[E_p(t+i)+E_f(t+i)\right]$ and η_f is learning rates. Except for $\partial E_3/\partial F_{act}(t+i)$, all terms in (18) can be computed using the error back propagation. 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 longer than the data window N. As a result, the error function E_3 can be approximated as

$$\frac{1}{2}\sum_{t=0}^{N_f} \left[E_p(t+i)+E_f(t+i)\right] \approx \frac{1}{2}\sum_{t+i=0}^{N_f} \left[E_p(t+i)+E_f(t+i)\right]. \quad (19)$$

The direction of the gradient descent of the right hand side of (19) is approximated by the betterment process, and it can be defined as

$$\frac{\partial E_3}{\partial F_{act}(t+i)} \approx \left[\Gamma_p dX^k(t+i) - \{\Gamma_f dF^k(t+i) + \Phi_f \dot{dF}^k(t+i)\}\right]^T \quad (20)$$

where Γ_p, Γ_f and $\Phi_f \in \mathrm{R}^{m\times m}$ represent gain matrices.

During learning of the contact movement, the virtual trajectory is also modified. The learning rule can be derived in the same way as the FCN except for the energy function. Because the modification of the virtual trajectory makes the position error $E_p(t+i)$ insignificant, the energy function is modified as

$$E_4 = \sum_{t=0}^{N_f} \left\{ \frac{1}{2}\sum_{i=-N}^{N} h^2(i) E_f(t+i) \right\}. \quad (21)$$

4 Simulation Experiments

Computer simulations of planar movements (m=2) are performed under an assumption that the end-effector dynamics have already simplified as (3).

First, a free movement is simulated. The PCN and VCN are of three layered networks with two input, ten hidden and four output units. The learning rates are η_p=2.0×10^{-4} and η_v=5.0×10^{-5}. Γ_p and Γ_v are determined as $\Gamma_p = \frac{1}{2}\mathrm{diag}\left[\min_t(o_{p11}(t)), \min_t(o_{p22}(t))\right]$ and $\Gamma_v = \frac{1}{2}\mathrm{diag}\left[\min_t(o_{v11}(t)), \min_t(o_{v22}(t))\right]$, respectively (Kawamura, Miyazaki and Arimoto, 1986).

Fig.5 shows changes of the end-effector trajectory by learning. The number in the figure denotes the iteration of the betterment process. Each iteration includes 100 times of neural network learning. The desired trajectory of the end-effector is determined using the fifth-order polynomial, where t_f=1.0[sec], Δt=0.001 [sec] and N_f=1000. The end-effector trajectory coincides with the desired one within several iterations.

Table 1 shows impedance parameters, $M_d^{-1}K_d$

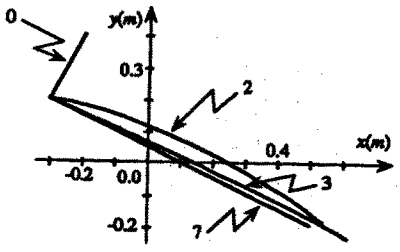

Fig.5 End-effector trajectories during learning of a free movement.

Table 1 Impedance parameters before and after learning for 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)	$\begin{bmatrix} 495.19279 & 0.44871 \\ 0.44652 & 475.05515 \end{bmatrix}$	$\begin{bmatrix} 497.59935 & 0.41674 \\ 0.56942 & 494.30498 \end{bmatrix}$

and $M_d^{-1}B_d$, before and after learning. $E[M_d^{-1}K_d]$ and $E[M_d^{-1}B_d]$ represent time averages of the corresponding impedance parameters. The diagonal elements of the impedance matrices increase after ten iterations.

Next, learning of a contact movement is performed for the same environment as Fig.5 except for an object placed along the x axis. The dynamics of the object are characterized as (2) where M_e=diag.[0, 0][kg], B_e=diag.[0,10][N·sec/m] and K_e=diag.[0,1.0×10^5][N/m].

The FCN is a three layered network with six input, ten hidden and four output units. The parameters are the same as TCN learning except for η_f=6.43×10^{-11} and η_d=6.0×10^{-8}. Also, Γ_f and Φ_f are determined as $\Gamma_f = \frac{1}{2}\mathrm{diag}\left[\min_t(o_{f11}(t)), \min_t(o_{f22}(t))\right]$ and $\Phi_f = \frac{1}{1000}\Gamma_f$.

Fig.6 shows changes of the end-effector forces by learning, where the Hanning window is used for $h(i)$ with N=100(see (18)) and the desired end-effector force is set to F_d=(0, -100)^T[N]. Before learning, large interaction force is exerted along the y axis (see the iteration number 0 in the figure). After thirty iterations, however, the end-effector force coincides with the desired one.

Fig.7 shows the effect of the window length N

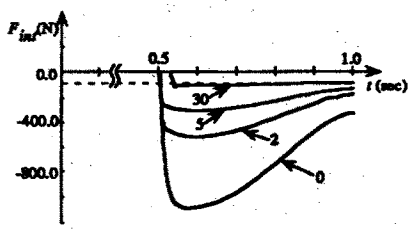

Fig.6 End-effector forces in the normal direction of the object during learning of a contact movement(N=100).

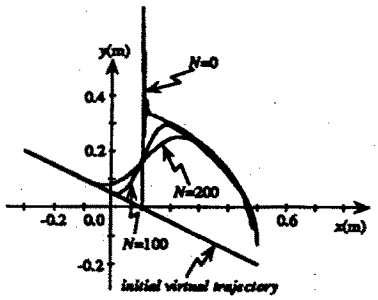

Fig.7 Learning results of virtual trajectories for contact movements(N=0, 100, 200).

to the learned virtual trajectories(after thirty iterations). In all cases, the end-effector forces coincide with the desired one. For N=0, the virtual trajectory suddenly changes just after the contact in order to absorb the impact force. As the window length increases, the virtual trajectories change before the contact so that a smooth transition from free to contact movements can be realized.

Table 2 shows impedance parameters before and after learning for the contact movement. After thirty iterations, the impedance parameters become considerably small in the normal direction of the contact surface. The impedance can be regulated according to the task by the iterative learning.

5 Concluding Remarks

The present paper proposed a new method to regulate the end-effector's impedance using neural networks. The method can regulate the second order impedance through iterative learning. Introducing the window function into the error functions, the virtual trajectory as well as the impedance parameters can be modified before contact.

The further research will be directed to improvements of learning speed and a generalizing ability of the proposed method for various classes of the constrained tasks. Finally, we are grateful to Mr. M.Nishida for the development of computer programs. The work was supported in part by Nissan Science Foundation to T.Tsuji.

References

Asada, H.: Teaching and learning of compliance using neural nets : representation of generation of nonlinear compliance. Proc. of IEEE Int. Conf. on Robotics and Automation, 1237-1244 (1990).

Cohen, M., Flash, T.: Learning impedance parameters for robot control using associative search network. IEEE Trans. RA-7-3, 382-390 (1991).

Hogan, N.: Impedance control : An approach to manipulation: Part I, II, III. ASME J.Dyn. Syst., Meas., and Cont., 107, 1-24 (1985).

Luo, Z. and Ito, M.: On control design for robot compliant manipulation, Trans. of the SICE of Japan, 26-4, 427-434 (1990)(in Japanese).

Table 2 Impedance parameters before and after learning for 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]