Proceedings of the American Control Conference
Anchorage, AK May 8-10, 2002

Real-time control and learning using neuro-controller via simultaneous perturbation for flexible arm system.

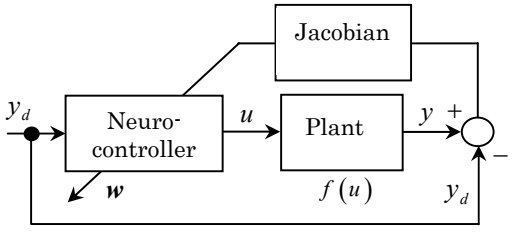
Yutaka Maeda
Department of Electrical Engineering, Kansai University
3-3-35 Yamate-cho, Suita Osaka, 564-8680 JAPAN.
PHONE : +81-6-6368-0932, FAX : +81-6-6388-8843
E-MAIL: maedayut@kansai-u.ac.jp

Abstract
This paper describes details of real-time control and real-time learning of neuro-controller for a flexible arm system using the simultaneous perturbation optimization method.
The simultaneous perturbation optimization method is useful, especially when dimension of the parameters to be adjusted is large. Therefore, it is beneficial to utilize the simultaneous perturbation method for neural networks.
On the other hand, when we use the ordinary gradient method as a learning rule of the neuro-controller, Jacobian of the plant is essential. However, the learning rule via the simultaneous perturbation does not require Jacobian of an objective plant so that the neural network uses only outputs of an objective system. Actual real-time control and real-time learning results of a real flexible arm system are described to confirm a feasibility of the proposed method.

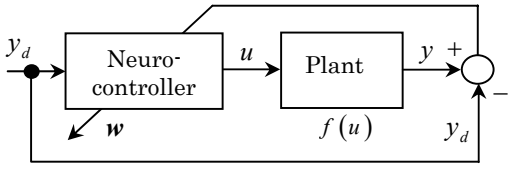
Keywords : Simultaneous perturbation, Neural networks, Neuro-controller, Real-time, Flexible arm

1. Introduction

Neural networks (NNs) are recently used in many fields. Especially, its non-linear information processing capability is intriguing.
In the field of control, NNs are one of hopeful tools. Neuro-controller (NC) by a direct inverse control scheme(see Fig.1) is one of promising approaches in non-linear control problems.
In order to use a NC as a direct controller, the NC must be an inverse system of an objective plant, that is, the NC must learn an inverse system in so-called indirect inverse modeling. Then, the learning rule plays a practical role, because it will be related to an arrangement of overall system.
In the case of indirect inverse modeling, generally, we need a plant model or a sensitivity function of the plant to acquire the derivatives needed for learning such as the backpropagation(BP) method, because the error function is usually defined not by an output of the NN but by that of the plant(see Fig.1(a)).



(a)Learning by the back-propagation.



(b)Learning by the SP method.

Fig.1 A basic scheme for a direct neuro-controller.

In this paper, we propose a NC using the simultaneous perturbation learning rule for a flexible arm system. This learning rule does not require a derivative of an error function but only values of the error function itself. Therefore, without knowing Jacobian of the objective arm system, we can design a direct neuro-controller. Then the neuro-controller can learn a changing environment about the system as well.
Ordinarily, an error function $J(\boldsymbol{w})$ is defined by a squared error of the plant. When we use a usual gradient method as a learning rule of the NC, we must know the quantity $\partial J(\boldsymbol{w})/\partial \boldsymbol{w}$.
Then, we have

$$\frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{\partial J(\boldsymbol{w})}{\partial y}\frac{\partial y}{\partial \boldsymbol{w}} = (y - y_d)\frac{\partial f(u)}{\partial u}\frac{\partial u}{\partial \boldsymbol{w}} \tag{1}$$

where y_d denotes the desired output of the plant. Therefore, $(y\text{-}y_d)$ is known. Moerover, we can calculate $\partial u/\partial \boldsymbol{w}$ like the back-propagation learning rule. However, we don't know the sensitivity function. As a resutlt, we can not obtain a proper modifying quantities for weights in the NC.
On the other hand, we can introduce an idea of the simultaneous perturbation learning rule. In this

case, there is no need to know the sensitivity function of the unknown plant. Only using the values of an error function, the learning rule can estimate a gradient of the error function with respect to adjustable parameters, weights of the NC in this case.

Moreover, the algorithm of the simultaneous perturbation is very simple. This implies easy implementation and real-time learning of NCs. Actually, real-time learning of NC was achieved in this research. The NC learns an inverse system of an objective plant and controls the plant at the same time.

Even if environment changes, e.g. change of mass of equipment, the NC can adapt new situation by the learning. In usual control scheme such as the state feedback control, if the environment changes, we have to recalculate parameters used in controller such as the feedback gain.

From these points of view, NCs using the simultaneous perturbation is significant in the control problems.

2. Simultaneous perturbation learning rule

The idea of the simultaneous perturbation was proposed by J.C.Spall as an extension of Kiefer-Wolfowitz stochastic approximation[1][2]. J.Alespector et al. and G.Cauwenberghs also proposed the same idea[3][4]. Independently, Y.Maeda introduced the same algorithm as a learning rule of neural networks[5][6]. J.C.Spall et al. and Y.Maeda reported some applications of the simultaneous perturbation method in control problems[7][8][9].

Now, we describe the simultaneous perturbation leaning rule used in this paper.

Define a weight vector and a sign vector as follows;

$$\boldsymbol{w}(t)=\left(w_1(t),w_2(t),\cdots,w_N(t)\right)^T$$
$$\boldsymbol{s}(t)=\left(s_1(t),s_2(t),\cdots,s_N(t)\right)^T \quad (2)$$

Where t denotes iteration, superscript T is transpose of a vector. $\boldsymbol{s}(t)$ is a sign vector whose components are +1 or −1.

The i-th component of the modifying vector of the weights $\Delta w_i(t)$ is defined as follows;

$$\Delta w_i(t)=\frac{J\left(\boldsymbol{w}(t)+c\boldsymbol{s}(t)\right)-J\left(\boldsymbol{w}(t)\right)}{c}s_i(t) \quad (3)$$

Where c is a magnitude of the perturbation. The weights are updated as follows;

$$\boldsymbol{w}(t+1)=\boldsymbol{w}(t)-\alpha\Delta\boldsymbol{w}(t) \quad (4)$$

Where, α is positive learning coefficient.

Note that only two values of the error function; $J(\boldsymbol{w}(t))$ and $J(\boldsymbol{w}(t)+c\boldsymbol{s}(t))$ are used to update the weights in the network. Any information about the objective plant does not included in the learning rule.

In this paper, we adopt the simultaneous perturbation with the sign vector that is equivalent to the random direction type of optimization method. This method is easy to implement. However, we have to pay attention to difference of the simultaneous perturbation and the random direction.

3. Simultaneous perturbation for flexible arm system

We consider a one-freedom flexible beam shown in Fig.2 as an objective plant.

In usual control scheme, we must have an exact model of an objective plant, since controllers are basically designed based on the identified model. Therefore, when some characteristics of the plant change, we must detect the change to compensate the controller.

On the other hand, our scheme used here works without this information, because only values of the error function are required. Without a model of the plant or information about the plant, the NC via the simultaneous perturbation can generates proper input for the plant.

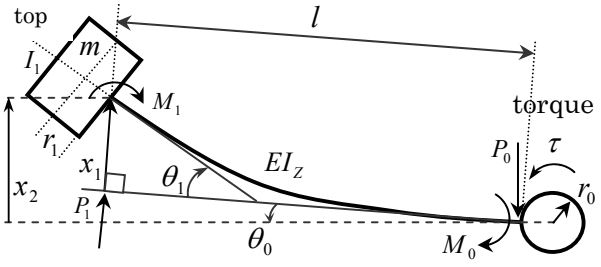


Fig.2 A flexible arm

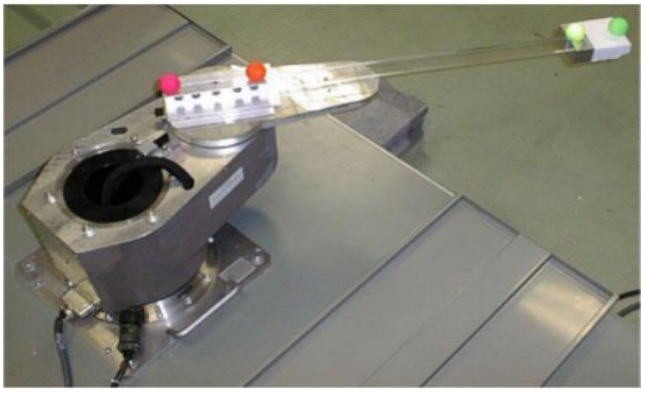


Fig.3 Picture of the flexible arm.

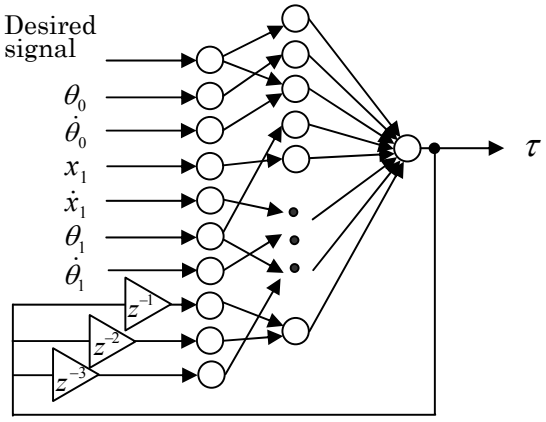


Fig.4 Neural network used here

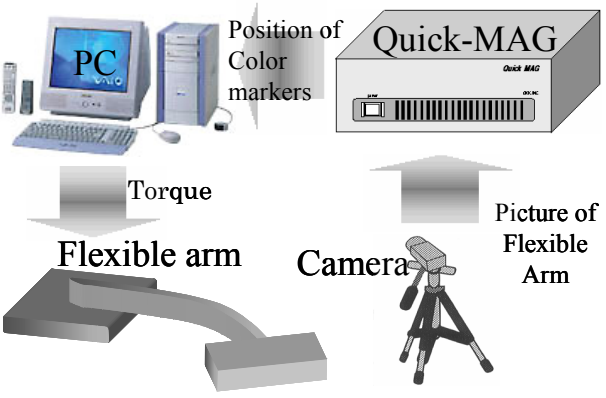


Fig.5 Flexible arm system.

Moreover, proper modification of the controller is carried out under operation. That is, on-line learning is possible in this learning scheme.

3.1 The flexible beam

A picture of the actual flexible arm is shown in Fig.3. The arm is made of acrylic and the rigid body is attached to top of the arm. There are some color markers on the beam to measure some states of the arm.

Mass of the top of the arm is 0.145[kg], and length of the arm is 0.472[m].

Then, we selected six states to control the plant. $\boldsymbol{x}$ is the six dimensional state variable as follows;

$$x = \left(\theta_0 \quad \dot{\theta}_0 \quad x_1 \quad \dot{x}_1 \quad \theta_1 \quad \dot{\theta}_1\right)^T \tag{5}$$

τ denotes a torque input as a command against the plant. Inputs of the NC are four states and their derivatives of Eq.(5) as shown in Fig.4. We know that these six states are necessary to produce a proper command input for the system.

Using these states, the neural network outputs a torque τ.

3.2 Neural network

The objective plant has a dynamics. Therefore, simple multi-layered neural network is not appropriate to control the plant, since the multi-layered network cannot have any memories. Such a network cannot handle a dynamic information processing. Thus we used a multi-layered neural network with feedback.

The network used here is shown in Fig.4. Basic construction is a simple multi-layered network. However, the network has time-delayed feedback inputs from output of itself. This feedback gives dynamics to the network.

3.3 Control system

Fig.5 is schematic flow of signals based on actual equipments. The system consists of the objective flexible arm, CCD camera, image processing device and PC which controls the system and realizes the recurrent neural network.

Color markers are attached to the both arm ends to get the arm positions.

States of the flexible arm is monitored by the camera. An image processing device Quick-MAG converts positions of color markers equipped on the arm into numerical values. A PC calculates states from these data and outputs torque by the NC realized in the PC. This cycle is repeated.

4. On-line learning by simultaneous perturbation.

Fig.6 shows a configuration of the system. The error function is defined as follows;

$$J\left(u\left(\boldsymbol{w}\right)\right) = \sum_i \left(x_{2,i} - d_i\right)^2 \tag{6}$$

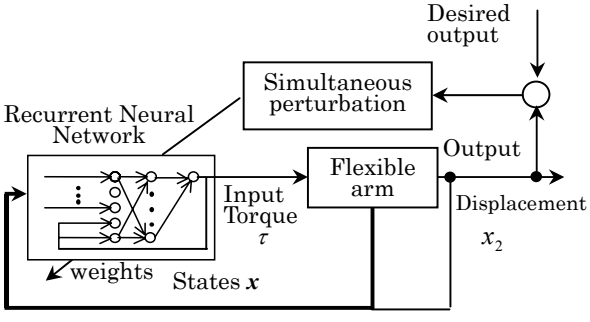


Fig.6 Overall configuration of the system

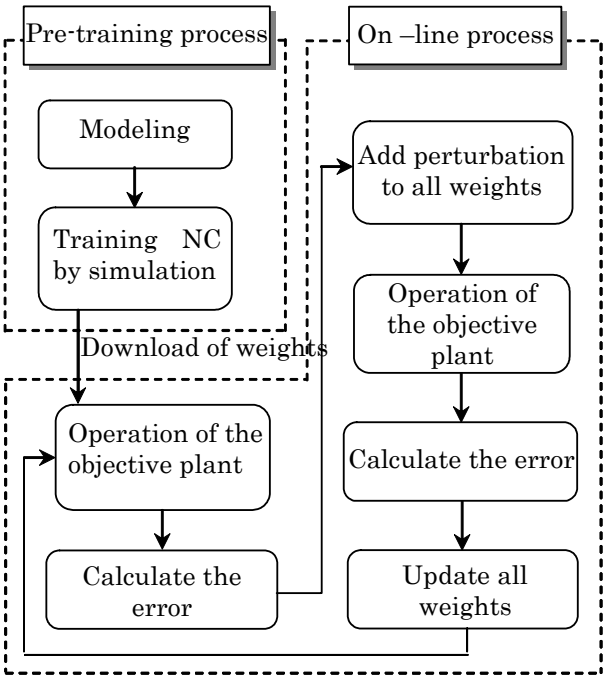


Fig.7 Flowchart

Where, d_i denotes a desired position of the top of the arm at the i-th sampling time. That is, the error is a sum of the squared error of the position of the end for ten seconds. Every trial gives a value of the error function.

Without perturbation, we make a trial and obtain a value of the error in Eq.(6). Next, we add perturbations to all weights simultaneously and make a trial. Then, we have a value of the error, i.e. $J(u(\boldsymbol{w}+c\boldsymbol{s}))$. By using Eq.(3) and (4), we can update all weights in the NC. We repeat this procedure. This is a learning cycle of the NC. This cycle is carried out with control of the flexible arm in every trial.

Total flowchart of this learning is shown in Fig.7. In a case, we need pre-training, since using untrained NC is reckless. After the pre-training of the neural network, the network is utilized as a controller of the plant. However, in many preliminary experiments, initial value of zeros for NC yields stable results without pre-training.

In control cycle, based on the measurements, a position of the top of the arm, angles θ_1 and θ_2 and their derivative are calculated. These measured data are fed into the NC realized by PC. Torque calculated by PC is sent to the flexible arm through a driver. This process is carried out for every sampling time. The sampling time is mainly restricted by capability of the image processing device.

4.1 Vibration reduction control

From a certain initial state, we would like to reduce vibration of the top of the flexible arm.

Initial weights of the NC are all zero. The learning coefficient α and the perturbation c are both 0.0002. Fig.8 is a result after 30 times on-line learning by our learning rule. Vibration is reduced, compared with free vibration. The NC controls the actual flexible arm system.

4.2 Tracking control

The learning coefficient α and the perturbation c are 0.00002 and 0.0001, respectively. Initial weights of the NC are all zero.

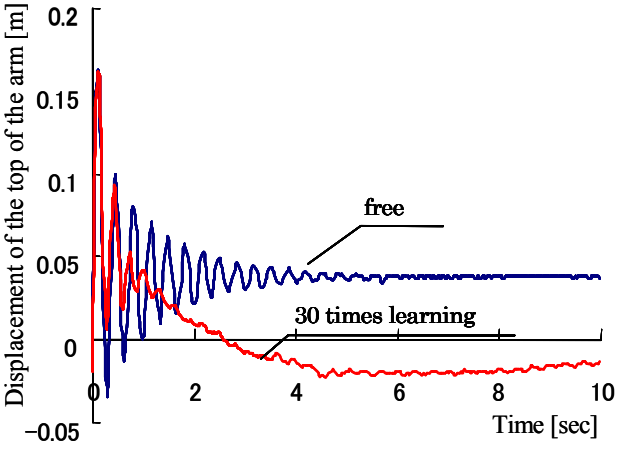


Fig.8 A vibration control of the flexible arm

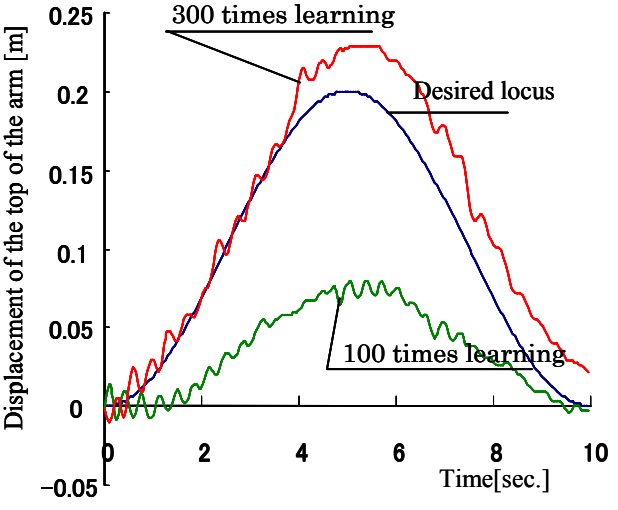


Fig.9 A result of a tracking control.

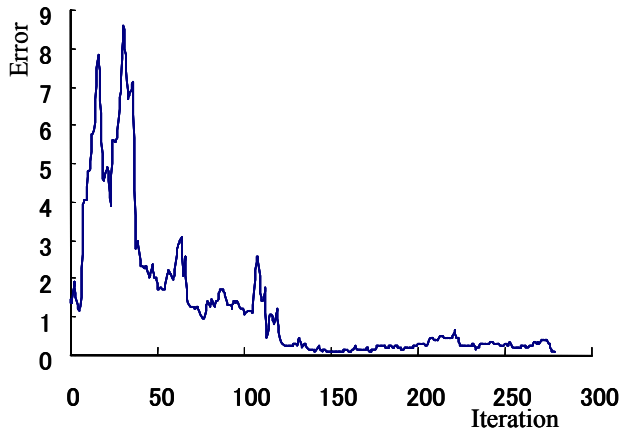


Fig.10 Change of error.

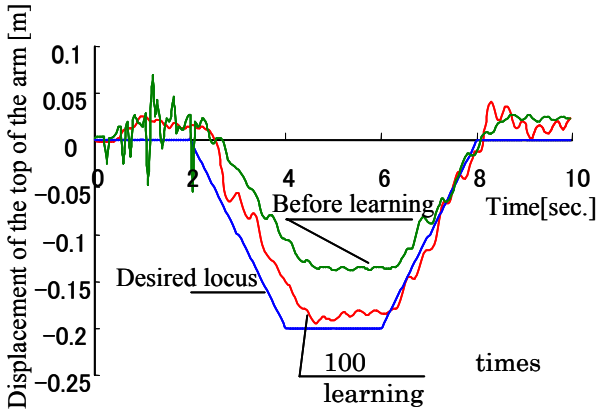


Fig.11 A result of a tracking control for random locus.

Fig.9 shows results after 100 times and 300 times on-line learning by our learning rule. Desired locus is sinusoidal wave which amplitude is 0.2[m], period is 10[sec]. As on-line learning proceeds, locus is closing the desired one.

Fig.10 shows change of error for sinusoidal locus. We can see that the NC works well as iteration goes. Next, we assigned other locus for the same NC. Weights values learned in the previous example are used as initial weights values of the NC in this example. Target locus changes randomly for every trial.

Fig.11 and Fig.12 show results for different locus. Even for different locus, the NC tried to adapt new target locus.

Change of error for different target locus is depicted in Fig.13. Since desired locus changes every trial, error changes violently. However, on average, the error decreases as learning goes.

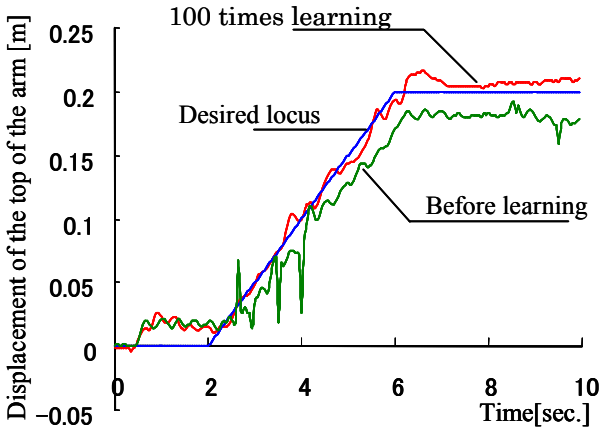


Fig.12 A result of a tracking control for random locus 2.

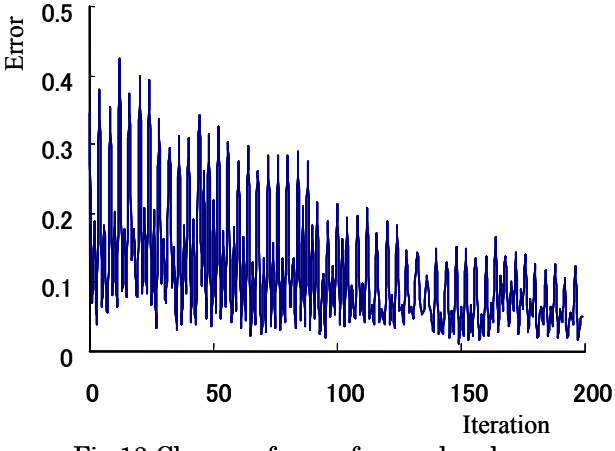


Fig.13 Change of error for random locus.

5. Conclusions

In this paper, a flexible arm system controlled by a NC using the simultaneous perturbation learning rule is described. Moreover, we could apply this scheme to a real time system. The back-propagation learning rule is not applicable to this without any information on the objective plant. However, by using the simultaneous perturbation learning rule, the NC can learn an inverse of the object plant.

Acknowledgement

This research is financially supported by Kansai University Frontier Sciences Center and High Technology Research Center.

Author would also like to thank to Mr.Kubo for his assistance.

References

[1]J.C.Spall(1987), A Stochastic approximation technique for generating maximum likelihood

parameter estimates, Proceedings of the 1987 American Control Conference, pp.1161-1167
[2]J.C.Spall(1992), Multivariable stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Automatic Control, vol.37, pp.332-341
[3]J.Alespector, R.Meir, B.Yuhas, A.Jayakumar and D.Lippe(1993), A parallel gradient descent method for learning in analog VLSI neural networks, in S.J.Hanson, J.D.Cowan and C.Lee(eds.), Advances in neural information processing systems 5(pp.836-844), San Mateo, CA : Morgan Kaufmann Publisher
[4]G.Cauwenberghs(1993), A fast stochastic error-descent algorithm for supervised learning and optimization, in S.J.Hanson, J.D.Cowan and C.Lee(eds.), Advances in neural information processing systems 5(pp.244-251), San Mateo, CA : Morgan Kaufmann Publisher
[5]Y.Maeda and Y.Kanata(1993), Learning rules for recurrent neural networks using perturbation and their application to neuro-control, Transactions of the Institute of Electrical Engineers of Japan, vol.113-C, pp.402-408 (in Japanese)
[6]Y.Maeda, H.Hirano and Y.Kanata(1995), A learning rule of neural networks via simultaneous perturbation and its hardware implementation, Neural Networks, vol.8, pp.251-259
[7]J.C.Spall and J.A.Cristion(1994), Nonlinear adaptive control using neural networks : Estimation with a smoothed form of simultaneous perturbation gradient approximation, Statistica Sinica, vol.4, pp.1-27
[8]J.C.Spall and D.C.Chin(1994), A model-free approach to optimal signal light timing for system-wide traffic control, Proceedings of the 1994 IEEE Conference on Decision and Control, pp.1868-1875
[9]Y.Maeda and R.J.P.deFigueiredo(1997), Learning Rules for Neuro-Controller Via Simultaneous Perturbation, IEEE Trans. on Neural Networks, vol.8, no.5, pp.1119-1130