# Control Scheme for SCARA by Recurrent Neural Network Using Simultaneous Perturbation

YUTAKA MAEDA and NAOYUKI ISHIBASHI
Kansai University
Department of Electrical and Electronic Eng.
Yamate-cho, 3-3-35, Suita
JAPAN
maedayut@kansai-u.ac.jp

*Abstract:* Robots are widely used in many fields. It is important to provide many different methodologies for robot control. This paper proposes a real time scheme for robots control and learning using recurrent neural network. We handle a problem to control a position and a trajectory of tip of a Selective Compliance Assembly Robot Arm(SCARA) robot. We adopt the simultaneous perturbation optimization method as a learning rule of the recurrent neural networks(RNNs). Then the RNNs have to learn an inverse dynamics of the SCARA robot. Position and trajectory control of a SCARA robot using RNN are considered. We could confirm that the RNNs can learn the inverse dynamics and work as a neuro-controller. We describe details of the control scheme. Some experimental results for these control using an actual SCARA robot are shown.

*Key–Words:* Robot control, Learning, Recurrent neural networks, Simultaneous perturbation, SCARA, Inverse dynamics, Real time control

## 1 Introduction

Nowadays robots have very important role in our society and are widely used in many fields including industry, medicine and daily life. Therefore, it is crucial to contrive novel control scheme for robots under diverse environments.

There are many different types of control schemes for robots. Especially control scheme which can cope with changing environment is attractive and interesting. On the other hand, neural networks can change their characteristic by learning. From these points of view, it seems that control scheme using artificial neural networks are promising[1, 2, 3, 4].

In this research, we present an on-line control and learning scheme using recurrent neural networks. Since neural networks can change their characteristic by learning, the control scheme presented here flexibly copes with changing characteristic of the objective robot and environments.

The control objective of this work is a SCARA(Selective Compliance Assembly Robot Arm) robot. We consider a position control and a trajectory control of the robot.

We use the simultaneous perturbation optimization method as a learning rule of the neuro-controllers. Adopting this method, we can obtain the proper recurrent neural network without a prior knowledge on the objective system.

We made experiments using actual SCARA robot. We confirmed that we could control the actual SCARA robot by using this control scheme.

This paper consists of six chapters. In chapter 2, we describe the presented control scheme using recurrent neural network. Chapter 3 describes the learning procedure using simultaneous perturbation method. In chapter 4, we explain about simulation results based on the proposed method. In chapter 5, we show some experimental results using actual SCARA robot for the position control and the trajectory control. Conclusion is in chapter 6.

## 2 Control scheme using recurrent neural network

Many different types of controllers are used for many control problems including positioning or trajectory control of robots. However, these controllers are not sometimes suitable for robots changing their characteristics, for example, changing of payload. With characteristics change, it is necessary to readjust their parameters to maintain their performance.

One approach for the problem is to utilize artificial neural networks[5]. One interesting characteristic of neural networks is their flexibility. Control scheme using neural networks, that is, neuro-controllers can

change their characteristics by learning. Therefore, the neuro-controllers can cope with control of the robots under changing environment.

We handle a problem to control a tip of a SCARA robot. We would like to control the tip of the SCARA robot using neuro-controller.

### 2.1 System configuration

Our control objective is a SCARA robot with two joints shown in Figure 1. Their arms are rigid and tip of the arm moves only in the XY-plane.A schematic diagram of our system is shown in Figure 2.

Target signals are for two joint angles $\theta_1$ and $\theta_2$ of the SCARA robot. Target signals and state quantities of the robot are fed to a recurrent neural network. The recurrent neural network receives them and produces two controlling torques $\tau_1$ and $\tau_2$ corresponding to the two arms of the SCARA robot. The SCARA robot receives them and the tip of the arm moves . Therefore, the recurrent neural network have to learn an inverse dynamics of the SCARA robot and works as a controller.

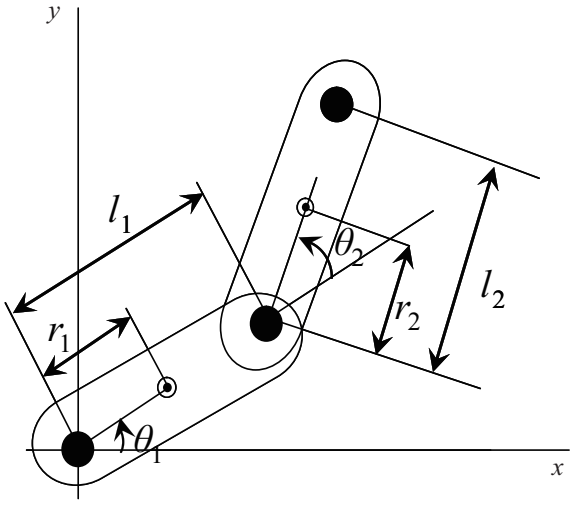


Figure 1: SCARA

### 2.2 Recurrent neural network

Figure 3 shows the recurrent neural network used as a controller of the SCARA robot.

This neural network has one middle layer. Input layer has 14 neurons, middle layer has 20 neurons and output layer has two neurons. Inputs of the network are desired two joint angles $\theta_{1d}$ and $\theta_{2d}$ and state quantities of the robot. Moreover, the network has feedback of past outputs of the network.Weight

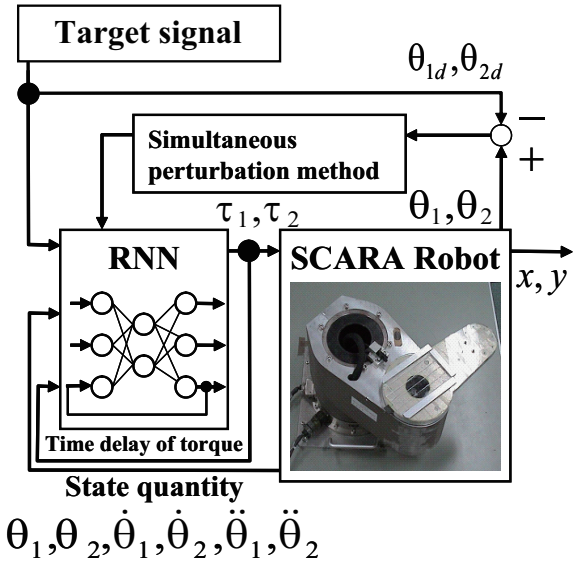


Figure 2: System configuration

values of the network are updated by the simultaneous perturbation learning rule.

## 3 Learning scheme

### 3.1 Simultaneous perturbation optimization method

Learning scheme is one of important issues, when we use neural networks. Also for the neuro-controller, it is crucial to realize an on-line learning scheme.

The back-propagation learning method is successful learning scheme for neural networks. In many applications of neural networks, the back-propagation method is ordinarily used.

However, when we use neural networks as controller of a certain objective, we have to know information; sensitivity of the objective. If the sensitivity of the plant is unknown, it is practically difficult to apply the back-propagation method.

In this research, we used the simultaneous perturbation optimization method as a learning rule of the recurrent neural network. Even the sensitivity is unknown, the simultaneous perturbation learning scheme is applicable, since the method uses only values of error function. The algorithm of the method is as follows;

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \Delta \boldsymbol{w}_t \tag{1}$$

$$\Delta w_{t,i} = \frac{J(\boldsymbol{w}_t + c\boldsymbol{s}_t) - J(\boldsymbol{w}_t)}{cs_{t,i}} \ (i = 1, 2, \ldots, n) \tag{2}$$

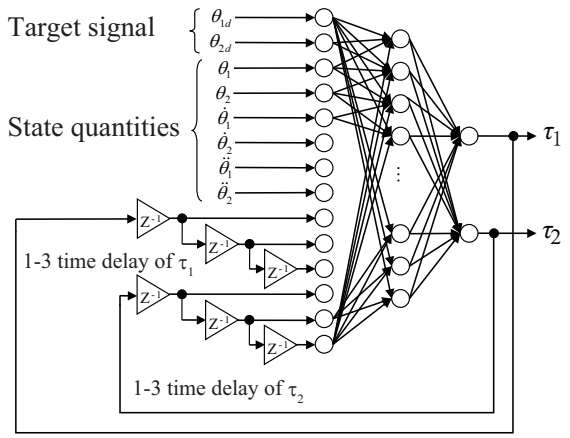


Figure 3: Recurrent neural network for inverse dynamics problem

Where $J$ denotes an evaluation or an error function. $w$ is weight values of the neural network including threshold. $\alpha$ means a learning coefficient. $c$ means a magnitude of the perturbation. $s_t$ is a sign vector whose elements $s_{t,i}$ are $-1$ or $1$.

The simultaneous perturbation method is widely used as a stochastic gradient method in many fields. The optimization method was introduced by J. C. Spall[6, 7]. Y. Maeda also independently proposed a learning rule for artificial neural networks using simultaneous perturbation and reported on the feasibility of the learning rule for some problems[8, 9, 10]. This method is well suited to hardware implementation of learning mechanism of artificial neural networks[9, 11, 12].

As mentioned before, the simultaneous perturbation learning method does not require the Jacobian of the objective SCARA robot. We use only two values of the error function; with the perturbation and without the perturbation. Based on these values, the method estimates the gradient vector of the function. Even if we do not know the characteristics of the objective process, we can construct the learning and control scheme using this method.

Moreover it is generally difficult to construct a learning scheme of recurrent types of neural networks, since signals in the network go back and forth. Therefore error propagation through time is necessary for learning. This mechanism is actually difficult to realize in real time.

On the other hand, the simultaneous perturbation method is very easy to realize. Error propagation through time is not necessary even for recurrent neural networks. Only values of the evaluation function are required to update the weights of the network. The method is easily applicable to on-line problems as well.

## 3.2 Evaluation function

We define the evaluation function $J$ and the square error $r_n$ as follows;

$$J = \sum_n \{1 - exp(-g_n)\} \tag{3}$$

$$g_n = (\theta_{1d} - \theta_{1n})^2 + (\theta_{2d} - \theta_{2n})^2 \tag{4}$$

Where $\theta_{1d}$ and $\theta_{2d}$ are desired two joint angles. $\theta_{1n}$ and $\theta_{2n}$ are actual two joint angles at every sampling time $n$. $r_n$ denotes the squared errors for the first and the second joints.

Generally speaking, using Euclidean squared error $g_n$ is reasonable. Therefore, it seems appropriate to use accumulated squared error for a certain period. However, when we consider overall learning process of the neuro-controller, the accumulated squared error in early stage is so large that the change of the error is very rude. Using the original accumulated squared error, the error is so large that modifying quantities for weights are also large. As a result, the learning precess is getting unstable in the early stage. On the other hand, the error is extremely small in final stage of learning process. This results in very slow learning. In order to obtain stable and efficient learning, scaling or smoothing is necessary.

In order to avoid such situation, we utilize the evaluation function $J$ shown in Eq.(3) which is an accumulated squared error of two angles at every sampling time for a certain period with scaling[5]. That is, the evaluation function $J$ is scaled reasonably with exponential function.

## 3.3 Learning procedure

A flowchart of overall learning scheme is shown in Figure 4. Sampling period is 0.05[sec.].

1. We set some parameters such as the perturbation, the learning coefficient, teaching signal and so on.
2. Initial weights of the neural network are set.
3. The robot operates without the perturbation. Then we observe a value of the evaluation function without the perturbation.
4. The robot operates with the perturbation. Then we observe a value of the evaluation function with the perturbation.
5. If the evaluation function decreases enough, we finish the learning process.

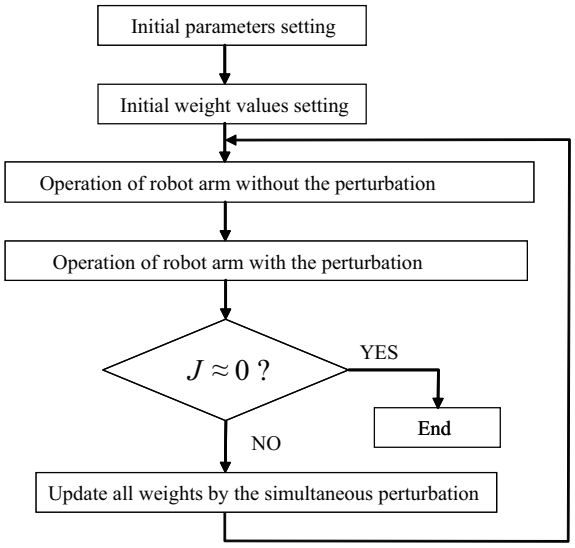


Figure 4: Flowchart

6. Based on the two values, we modify the all weights of the neural network using the simultaneous perturbation learning rule.

This learning method also has the advantage in calculation of learning process. In order to calculate two values of the error function with the perturbation and without the perturbation, the overall control system works twice. However, even the number of parameters, that is, weights is large, only twice operations realize modification of all weights in the neuro-controller. Moreover, we can control the objective and update the neuro-controller at the same time. That is, online learning is possible.

Using this control scheme, the neuro-controller copes with environmental change. Online learning adjusts the weight values of the network, so that control performance is maintained.

# 4 Simulation results

In this Chapter, we describe some simulations and their results. MatLab is used for the simulation.

In simulation, we have to model a control objective using mathematical equation. For target SCARA, we have to assume dynamics of the SCARA. In our simulation, we use the following motion equation for the SCARA.

$$M(\theta)\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\end{bmatrix}+\begin{bmatrix}-m_2 l_1 r_2 \sin\theta_2\left(\dot{\theta}_2^2+2\dot{\theta}_1\dot{\theta}_2\right)\\ m_2 l_1 r_2 \sin\theta_2\dot{\theta}_1\end{bmatrix}=\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} \quad (5)$$

Where, $l$ and $r$ are dimension of the SCARA and $m$ are mass of the two arms. $\theta$ are angles of them(see Figure 1). Elements of matrix $M(\theta)$ are defined as follows;

$$M_{11}=I_{zzg1}+I_{zzg2}+m_1{r_1}^2+m_2{r_2}^2+m_2{l_1}^2+2m_2 l_1 r_2\cos\theta_2 \quad (6)$$

$$M_{12}=I_{zzg2}+m_2{r_2}^2+m_2 l_1 r_2\cos\theta_2 \quad (7)$$

$$M_{21}=I_{zzg2}+m_2{r_2}^2+m_2 l_1 r_2\cos\theta_2 \quad (8)$$

$$M_{22}=I_{zzg2}+m_2{r_2}^2 \quad (9)$$

$I$ denote motions of inertia around an axis of rotation and chosen axis for the two arms.

## 4.1 Position control

Firstly, we handle position control of the robot using RNN. Five target positions and initial position of the robot arm are shown in Figure 5 and Table1.

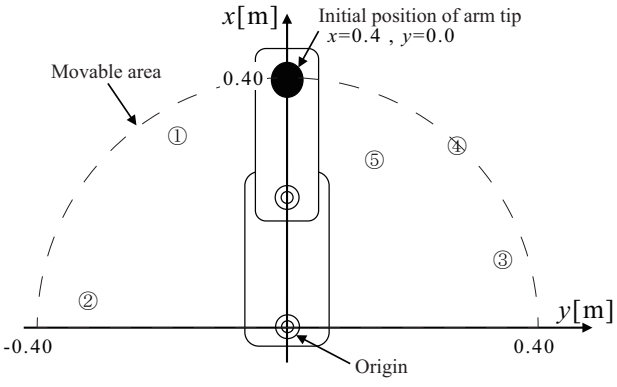


Figure 5: Target positions

The simulation was carried out using the procedure shown in Chapter 3. Initial weight values for the RNN were randomly generated using uniform distribution in the interval [-1 +1]. Learning coefficient

Table 1: Target positions

| targets | $\theta_1[^\circ]$ | $\theta_2[^\circ]$ | $x[m]$ | $y[m]$ |
|---|---|---|---|---|
| ① | -10 | 75 | 0.28159 | 0.1465 |
| ② | 30 | 90 | 0.073425 | 0.273263 |
| ③ | -75 | 1 | 0.107146 | -0.38537 |
| ④ | -60 | 25 | 0.26396 | -0.28782 |
| ⑤ | -75 | 95 | 0.239843 | -0.12478 |

$\alpha = 2.0 \times 10^{-5}$ and perturbation $c = 1.0 \times 10^{-5}$. Maximum learning iteration is 5000 times.

Results are depicted in Figure 6, Figure 7, Figure 8, Figure 9 and Figure 10. Target positions and loci of the tip of SCARA before learning process and after learning are depicted in these figures. We can compare the loci before learning process with those after learning. For initial setting, that is, before learning, the tip does not approaches to the corresponding target positions. After learning, all results show that the tip of the arm approaches the corresponding target positions for all cases.

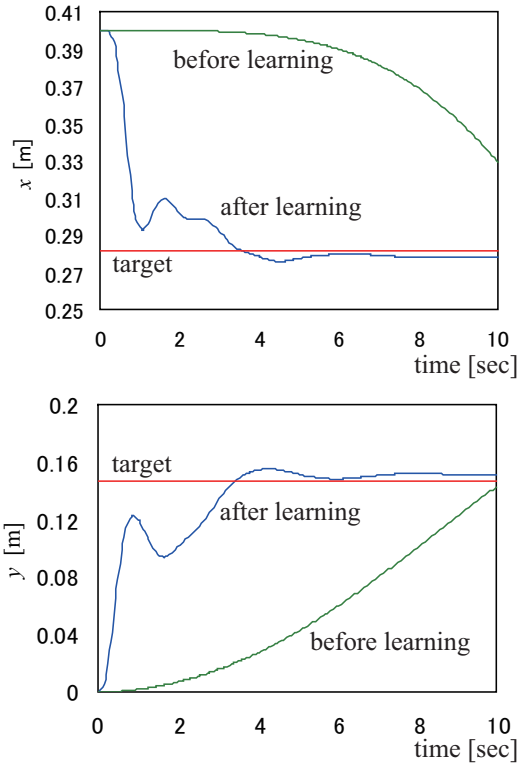


Figure 6: Simulation result for target 1

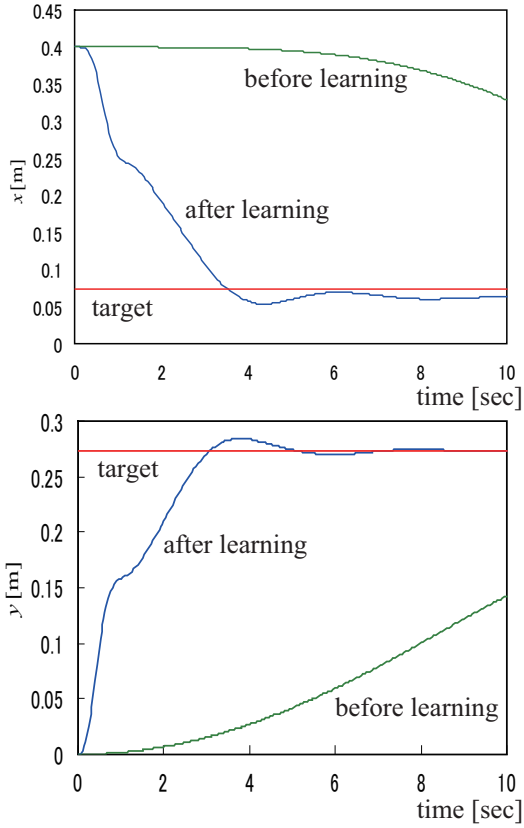


Figure 7: Simulation result for target 2

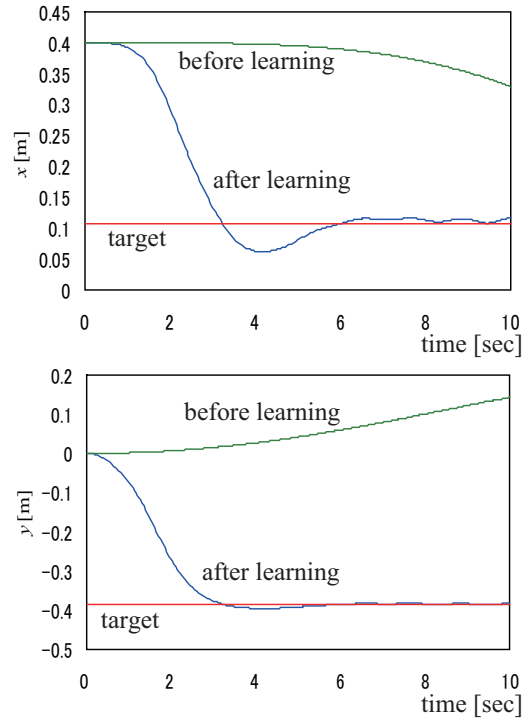


Figure 8: Simulation result for target 3

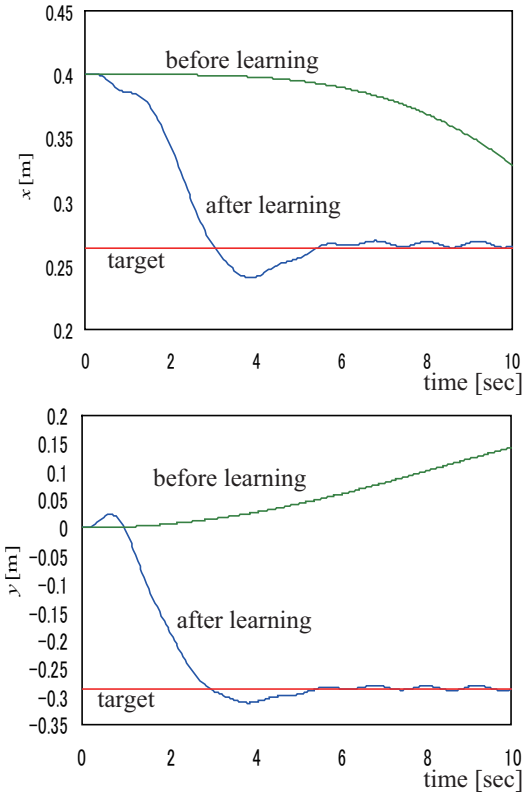


Figure 9: Simulation result for target 4

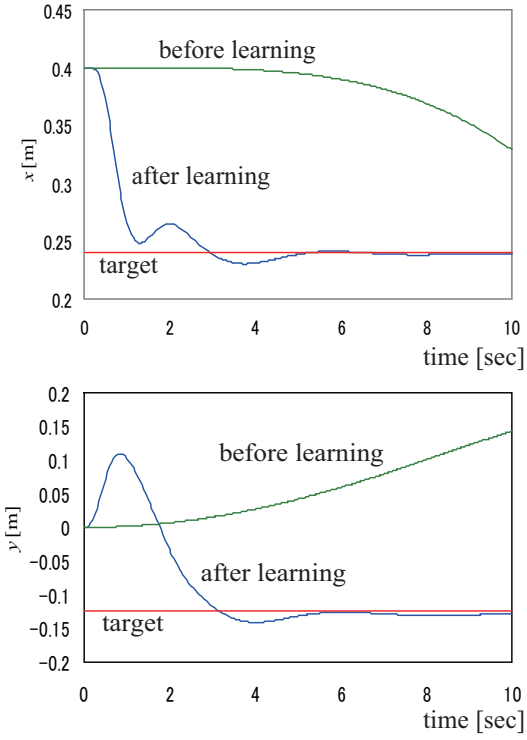


Figure 10: Simulation result for target 5

Then final position error are shown in Table 2. Maximum error is about 1[cm] for target 2. We can see that the learning and control scheme proposed here work well in the simulation.

Table 2: Error for target positions

| targets | Final position | | Error | |
|---|---|---|---|---|
| | $x[m]$ | $y[m]$ | $x[m]$ | $y[m]$ |
| ① | 0.278471 | 0.150982 | 0.003121 | 0.00447 |
| ② | 0.0063202 | 0.272685 | 0.010222 | 0.000577 |
| ③ | 0.0116182 | -0.38261 | 0.00904 | 0.00275 |
| ④ | 0.265595 | -0.28722 | 0.00165 | 0.00061 |
| ⑤ | 0.239057 | -0.129 | 0.000786 | 0.004224 |

Change of the evaluation function through learning process is shown in Figure 11. After 500 times learning, value of the error function decreses steadily. We can see that evaluation decreses as learning proceeds.

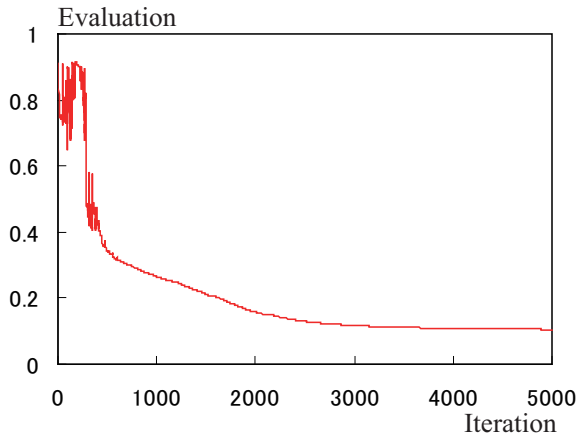


Figure 11: Change of evaluation function for position control

## 4.2 Trajectory control

Next, we consider a trajectory control problem of the SCARA robot. We set a trajectory shown in Figure 12. As in the figure, the trajectory consists of eight transit points. The tip of the arm should trace these points and consequently move on the trajectory.

Initial weight values for the RNN were randomly generated using uniform distribution in the interval [-1 +1] as well. Learning coefficient $\alpha = 1.0 \times 10^{-5}$ and perturbation $c = 5.0 \times 10^{-5}$. Maximum learning iteration is 15000 times.

Trajectories by the simulation after 3000 times learning, 6000 times learning and 9000 times learn-

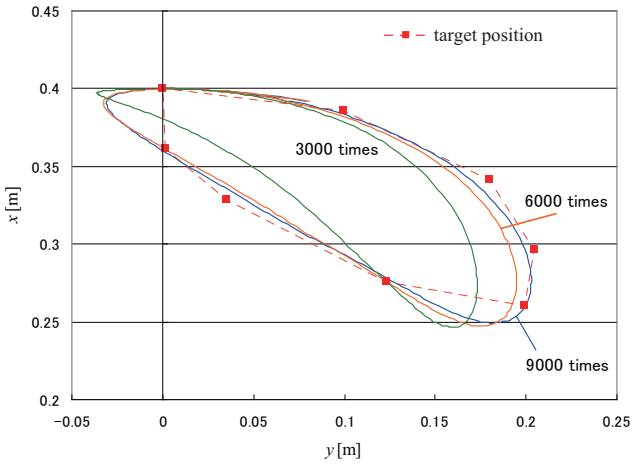


Figure 12: Result for trajectory control

ing are shown in Figure 12. As learning proceeds, trajectories are approaching to the target positions.

Change of evaluation function is shown in Figure 13. After 2000 times learning, the error decreses monotonously.

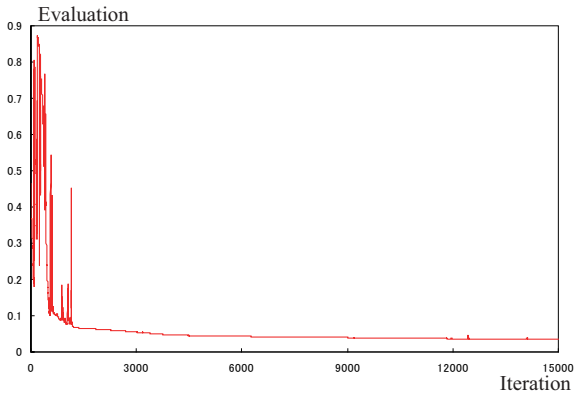


Figure 13: Change of evaluation function for trajectory control

Moreover, we consider two different target trajectories(see Figure 14). The same RNN should learn these two trajectories simultaneously.

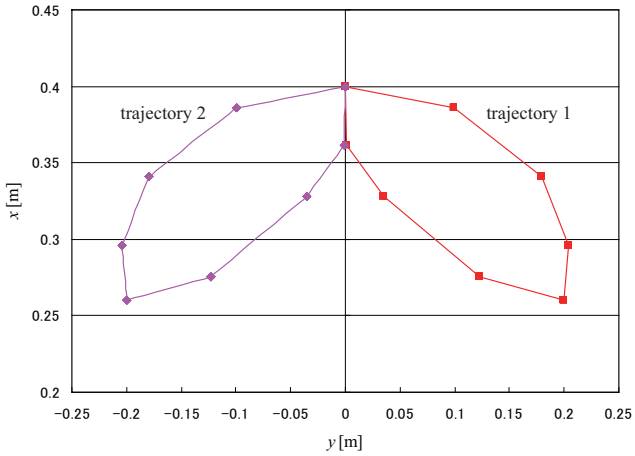


Figure 14: Two target trajectories

Trajectories by the simulation after 3000 times learning, 9000 times learning, 12000 times learning and 18000 times learning are shown in Figure 15 and Figure 16. As learning proceeds, trajectories are approaching to the target positions for both two trajectories.

Change of evaluation function is shown in Figure 17. After 10000 times learning, the error decreses monotonously.

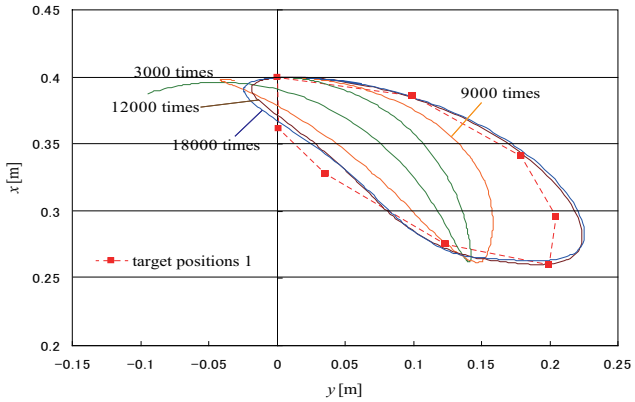


Figure 15: Result for target trajectory 1

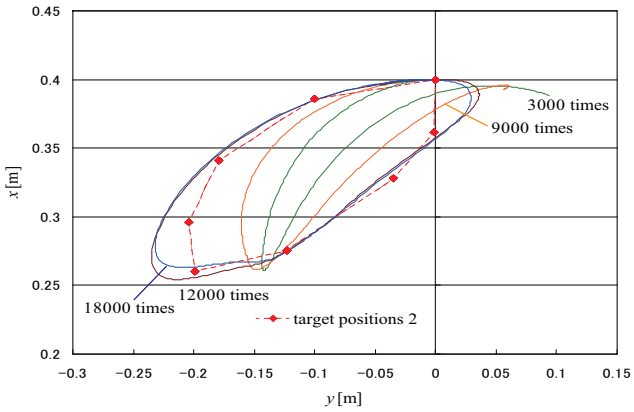


Figure 16: Result for target trajectory 2

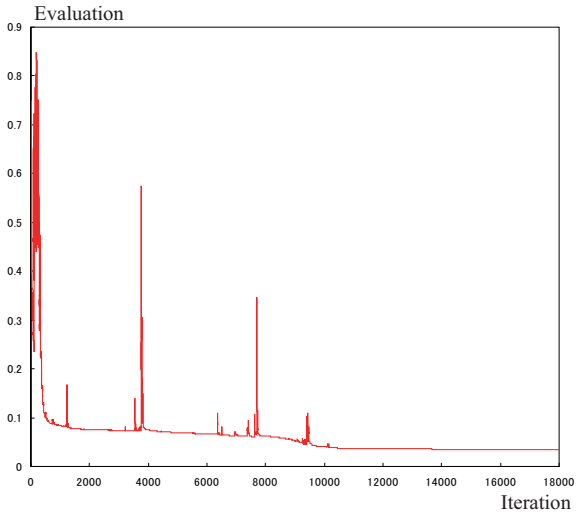


Figure 17: Change of evaluation function for two trajectories

# 5 Experimental results

## 5.1 Experimental system

Using practical robot system(see Figure 18), we carried out similar experiments. The control unit is a personal computer. In this computer, the recurrent neural network and its learning scheme are realized. The objective is SCARA (SR-402DD, Toshiba). This robot has an encoder. Two joint angles are measured by the encoder. The direct drive arm controller is also used to generate actual manipulating signals.

The control unit calculates two controlling torque $\tau_1$ and $\tau_2$ corresponding to the two arms of the SCARA and gives the drive unit them. The drive unit converts them into suitable manipulating signal. The SCARA receives them and moves the tip of the arm.

The control unit updates weights of the recurrent neural network based on the simultaneous perturbation method using observed positions of SCARA. And the control unit calculates two controlling torque again.

In the proposed control scheme, it is possible to carry out the learning under operation of the system. That is, operation and learning are simultaneously carried out.

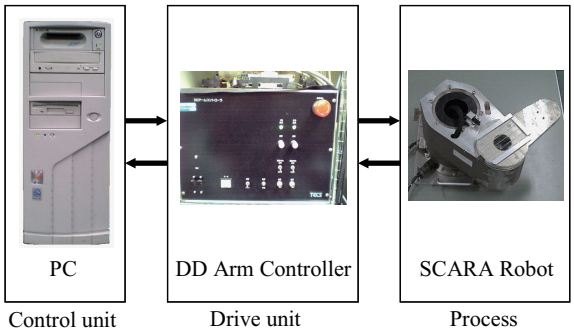


Figure 18: Practical experimental system

## 5.2 Position control

Firstly, in order to confirm a feasibility of the control and learning schemes, we made experiments for positioning control. We consider five target positions.

Initial X-Y position of the tip of the arm is (0.4 0.0). Target five positions are the same used in simulation and shown in Table 1 and Figure 5. In the previous Chapter, we carried out simulation for the same task. After 15000 times learning, the RNN worked well for the task. We use the same RNN as a controller and the weight values obtained by the simulation as initial values for practical SCARA system. Learning process is shown in Figure 4 in Chapter 3. We repeated the learning 1000 times.

Table 3 shows final positions and the corresponding errors for five target positions. Maximum error for these targets is about 4[cm].

Learning curve for the practical system is shown in Figure 19. We use result learnt by simulation as initial weight values. However, there exists difference. In the simulation, we assumed dynamics for the SCARA. In other words, we modeled the SCARA by certain motion equations. The model and the practical system had some differences. This results in the error shown in early stage of learning curve of Figure 19.

After 200 times learning, the error decreses monotonously but slowly, compared with change by

Table 3: Error for target positions by practical system

| targets | Final position | | Error | |
|---|---|---|---|---|
| | $x[m]$ | $y[m]$ | $x[m]$ | $y[m]$ |
| ① | 0.292636 | 0.138297 | 0.011046 | 0.008203 |
| ② | 0.027381 | 0.290961 | 0.046044 | 0.017698 |
| ③ | 0.136427 | -0.37595 | 0.029281 | 0.00276 |
| ④ | 0.281512 | -0.26996 | 0.001635 | 0.01786 |
| ⑤ | 0.2332 | -0.13974 | 0.006643 | 0.01496 |

the simulation result.

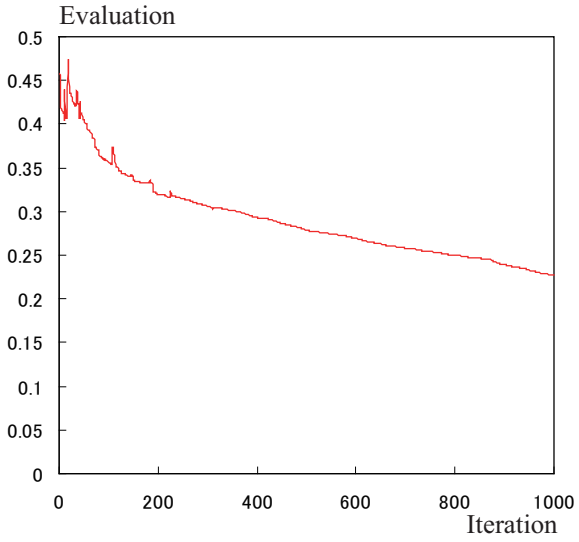


Figure 19: Change of evaluation function for position control using practical system

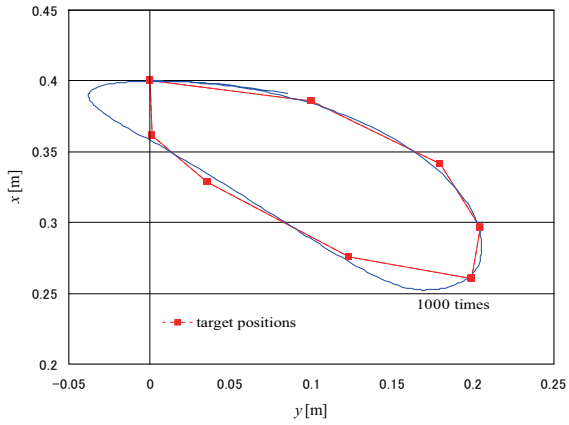


Figure 20: Result for trajectory control using practical system

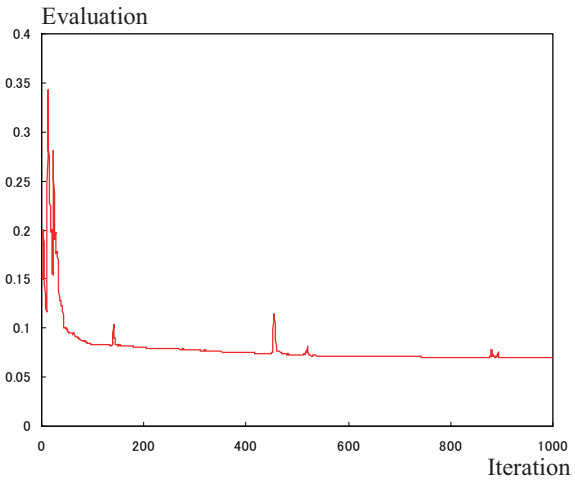


Figure 21: Change of evaluation function for trajectory control using practical system

### 5.3 Trajectory control

Next, we consider a trajectory control problem for practical SCARA. Similar to the simulation, we set a trajectory shown in Figure 12. The trajectory consists of eight transit points as well.

Initial weight values for the RNN were set to those obtained by the simulation. Maximum learning iteration is 1000 times. Figure 20 shows result after 1000 times learning. Change of the evaluation function is depicted in Figure 21.

Consecutively we handle the same problem with plural trajectories. Target trajectories are shown in Figure 14 in Chapter 4. The same RNN controls the SCARA robot for two trajectories. Results are shown in Figure 22 and Figure 23. Change of the evaluation function for these two trajectories is depicted in Figure 24.

## 6 Conclusion

We presented a control scheme using recurrent neural network for robot arms. The simultaneous perturbation learning method is applied. The method does not require sensitivity of the controlled objective. Moreover, learning procedure is simple and easy to implement even for recurrent neural networks. As a result, on-line learning of SCARA robot was realized.

We considered positioning control and trajectory control of the SCARA robot. The recurrent neural network learns inverse dynamics of the robot using the simultaneous perturbation learning scheme.

For the position control, we could confirm that the control system worked well for the target positions after about 400 times learning. Also for the trajectory

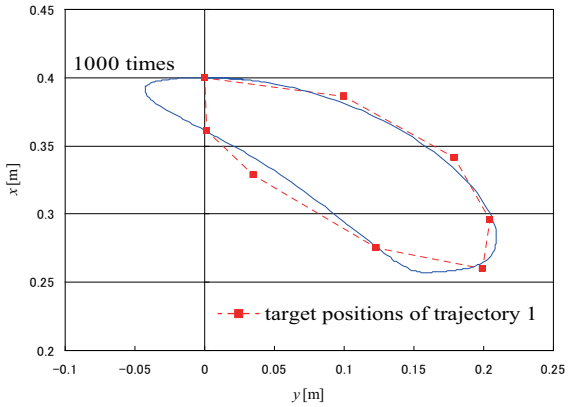


Figure 22: Result for target trajectory using practical system

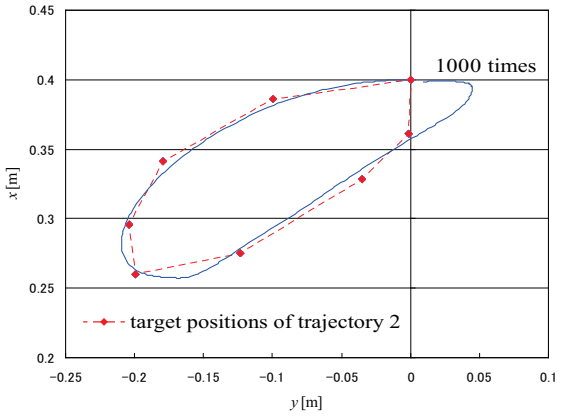


Figure 23: Result for target trajectory using practical system

control, the neural network could work well for desired trajectory.

*References:*

[1] S. Yildirim, Robot trajectory control using neural networks, *Electronics Letters,* 38, 2002, pp. 1111-1113.

[2] J. Reyes-Reyes, C. M. Astorga-Zaragoza, M. Adam-Medina, and G. V. Guerrero-Ramirez, Bounded neuro-control position regulation for a geared DC motor, *Engineering Applications of Artificial Intelligence,* 23, 2010, pp. 1398–1407.

[3] A. M. Shahri, B. J. Evans, and F. Naghdy, Neuro-fuzzy adaptive torque control of a SCARA robot, *1996 Australian New Zealand Conference on Intelligent Information Systems,* 1996, pp. 241-244.

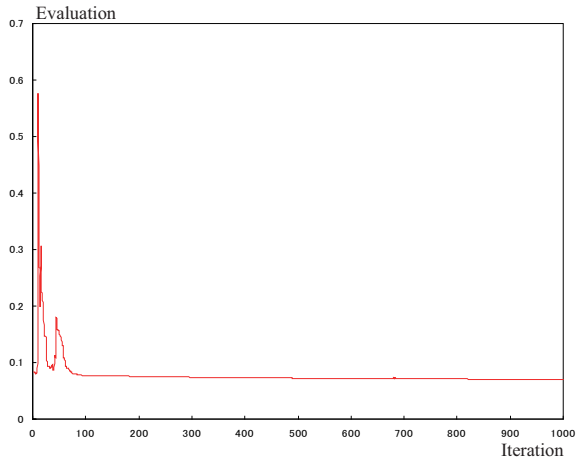


Figure 24: Change of evaluation function for two trajectories using practical system

[4] S. Omatu, M. Khalid and R. Yusof, *Neuro-Control and Its Applications,* Springer 1996.

[5] N. Ishibashi and Y. Maeda, Learning of Inverse-Dynamics for SCARA Robot, *SICE Annual Conference 2011,* 2011, pp. 1300–1303.

[6] J. C. Spall, Multivariable stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Trans. Autom. Control,* 37, 1992, pp. 332–341.

[7] J. C. Spall, *Introduction to Stochastic Search and Optimization,* John Wiley & Sons, Inc., 2003.

[8] Y. Maeda, Y. Kanata, A learning rule of neural networks for neuro-controller, *Proceedings of the 1995 World Congress of Neural Networks,* 2, 1995, pp.402–405.

[9] Y. Maeda, H. Hirano and Y. Kanata, A learning rule of neural networks via simultaneous perturbation and its hardware implementation, *Neural Networks,* 8, 1995, pp. 251–259.

[10] Y. Maeda, R.J.P.de Figueiredo, Learning rules for neuro-controller via simultaneous perturbation, *IEEE Transaction on Neural Networks,* 8, 1997, pp.1119–1130.

[11] Y. Maeda, T. Tada, FPGA implementation of a pulse density neural network with learning ability using simultaneous perturbation, *IEEE Trans. on Neural Networks,* 14, 2003, pp.688-695.

[12] Y. Maeda and M. Wakamura, Simultaneous Perturbation Learning Rule for Recurrent Neural Networks and Its FPGA Implementation, *IEEE Trans. on Neural Networks,* 16, 2005, pp.1664-1672.