SELF-ORGANIZING ADAPTIVE CONTROLLERS: APPLICATION TO THE INVERTED PENDULUM

T. Minatohara **T. Furukawa**
Department of Brain Science and Engineering, Kyushu Institute of Technology
Hibikino, Wakamatsu-ku, Kitakyushu 808-0196, Japan
minatohara-tetsuya@edu.brain.kyutech.ac.jp
furukawa@brain.kyutech.ac.jp

Abstract - *In this paper, a novel framework of adaptive controller is proposed. The architecture, called a 'self-organizing adaptive controller, (SOAC)' is based on a modular network SOM (mnSOM), in which each nodal unit of a conventional SOM is replaced by a pair of a controller and a predictor. Thus, an SOAC is regarded as an assembly of neural controllers, each of which is specialized to a context or a target object. An SOAC learns the way to control multiple contexts or objects, in parallel with generating a feature map of them. Simulations performed on an inverted pendulum suggested a high adaptive control ability.*

Key words - mnSOM, adaptive control, feedback-error-learning, inverted pendulum

1 Introduction

How can we give robots human control skills, which are accurate and flexible not too strict or sloppy, and such that they allow us to take it easy? This presents us with challenges. We especially focus on the generalization of performance from a small number of training patterns and quick responses to unexpected changes. The study aims to tackle the problem using a stepping stone of self-organizing architectures. Thus, our immediate target is to establish a '*self-organizing adaptive controller (SOAC)*'. In this paper, two ideas are adopted to our architecture to achieve the target. One is that the architecture is based on a modular network SOM (mnSOM), and the other is that each module consists of two blocks, namely, controller and predictor blocks.

The mnSOM proposed by Tokunaga et al. consists of an assembly of functional modules arrayed on a lattice; these are the replacements for the vector units of a conventional SOM [1–3]. An mnSOM has two aspects; one is its topology preserving mapping, and the other is its functional differentiation. After training has been completed, the mnSOM selects a functional module that best matches the context, and then processes input data with the module. Since many types of neural architectures are available for functional modules, it is also possible to employ neural controllers as the modules. In this case, the mnSOM is regarded as an assembly of controllers, each of which is specialized to different control contexts or different target objects. Therefore, the mnSOM is expected to switch those neural controllers depending on the context or the object; thus, the *best matching controller (BMC)* is selected adaptively. Further, all of these functional modules can be trained from a small number of training data sequences; thus, even if there are hundreds of modules and only a few training datasets, the mnSOM trains all modules by making appropriate interpolations. This generalization of

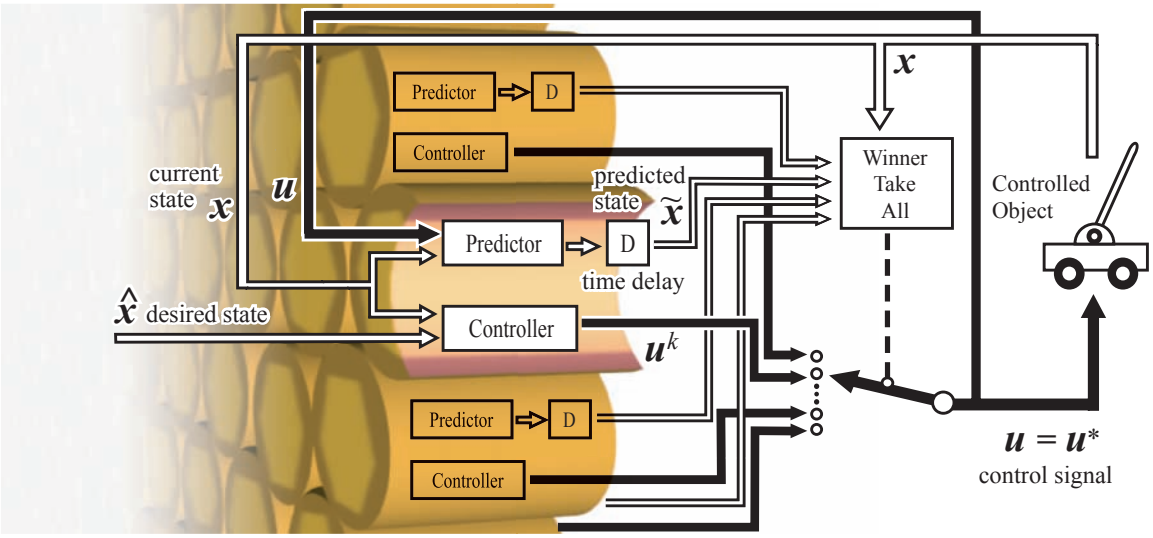

Figure 1: Scheme of self-organizing adaptive controllers.

performance is an advantage of using an mnSOM.

The second idea involves determining the BMC in a real-time control task. It is easy to determine the BMC *after* applying all the controllers to the target object, but this is unrealistic in a real-time task. Therefore, it is important to determine the BMC *before* controlling the object. To resolve this problem, functional modules are extended to consist of two blocks, i.e., controller and predictor blocks. The functional module that minimizes the prediction error is determined as the BMC, and the paired controller block is used for actual control. In other words, the mnSOM always predicts the object state at the next moment, and once an unexpected change occurs, it quickly changes the BMC.

A similar strategy with a modular architecture has been proposed by Wolpert and Kawato; their architecture consists of multiple paired forward and inverse models [4]. In this architecture, the forward and inverse dynamics models that are coupled together correspond to our predictor and controller, respectively. The main difference of our model from Wolpert's is that ours forms a self-organizing map of the target objects as well as controlling them. This property of our model should have some advantages for making an appropriate interpolation, analyzing objects, and finding hidden parameter spaces. Therefore, our model can be regarded as an SOM based extension of Wolpert's model. One may also find a resemblance to the Local Linear Map proposed by Martinetz et al., which was applied to the visual guided control task of a robot arm [5]. However, our architecture is essentially different. The purpose of a Local Linear Map is to represent a single nonlinear controller as the combination of the local linear operators; whereas, our model aims to generate an assembly of nonlinear controllers. It is worth noting that an mnSOM can employ Local Linear Maps as controller modules. Thus, 'the mnSOM with Local Linear Map modules' is one of the possible architectures of an SOAC.

In this paper, first the general idea of an SOAC is introduced, and then the results of its application to an inverted pendulum are presented.

2 Theoretical Framework

2.1 General Framework

The scheme of an SOAC based on an mnSOM is presented in Fig. 1. Basically, an SOAC has an arrayed structure in which each nodal unit of a conventional SOM is replaced by a functional module, such as a multilayer perceptron (MLP). As shown in the figure, each module consists of controller and predictor blocks. The controller block has 2 input vectors $\boldsymbol{x}(t)$ and $\hat{\boldsymbol{x}}(t)$, which are the current and the desired states of the target object, respectively. The corresponding output is the control signal, denoted by $\boldsymbol{u}^k(t)$ for the kth controller[1]. Thus,

$$\boldsymbol{u}^k(t) = {}^c\!f^k(\boldsymbol{x}(t), \hat{\boldsymbol{x}}(t)). \tag{1}$$

Here, ${}^c\!f^k(\cdot)$ represents the function describing the kth controller. On the other hand, each predictor has 2 input vectors $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, i.e., the current state and the current (real) control signal of the target object. The corresponding output is the prediction of the object state at Δt sec later;

$$\tilde{\boldsymbol{x}}^k(t + \Delta t) = {}^p\!f^k(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2}$$

Here, ${}^p\!f^k(\cdot)$ represents the function describing the kth predictor. The outputs of the predictors are delayed for Δt.

Before explaining the learning algorithm, we consider the execution phase. Suppose that an unknown target object is given after finishing the training phase. First, the prediction error is calculated as follows. (It is assumed that the initial control signal $\boldsymbol{u}(0)$ is set to 0 or a small random value).

$${}^p\!e^k(t) = (1 - \varepsilon)\,{}^p\!e^k(t - \Delta t) + \varepsilon \left\| \boldsymbol{x}(t) - \tilde{\boldsymbol{x}}^k(t) \right\|^2 \tag{3}$$

Then the least prediction error module is determined as the BMC at the time t. Here, let $*$ denote the index of the BMC given by

$$*(t) = \arg\min_k {}^p\!e^k(t). \tag{4}$$

If $\varepsilon = 1$ then the BMC is determined by the prediction error only at time t, whereas if $0 < \varepsilon < 1$ then the BMC is determined by the time averaged prediction error. The BMC control signal is selected as the actual control signal.

$$\boldsymbol{u}(t) = \boldsymbol{u}^*(t) = {}^c\!f^*(\boldsymbol{x}(t), \hat{\boldsymbol{x}}(t)) \tag{5}$$

Therefore, the controller coupled with the best predictor is always selected as the best controller.

2.2 Learning Algorithm of SOAC

Now suppose that there is a family of target objects that are described by the same equations, but are defined by different parameter sets. Further, suppose that the parameters continuously vary the dynamics of the object. I out of them are assumed to be known in advance, and can be used for training. Therefore, I desired controllers are also assumed to be prepared in advance, and are specialized to those objects. By using these controllers, I training sequences $\{(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t))\}$ $(i = 1, \ldots, I)$ are obtained for the objects.

[1]In this paper, superscripts and subscripts represent the indexes of modules and the training sequences, respectively.

The learning algorithm of an SOAC is identical to the algorithm of an mnSOM [3]. Thus, the training of an SOAC is performed iteratively, with each learning step consists of four processes, *evaluative process*, *competitive process*, *cooperative process* and *adaptive process*; the same as in the mnSOM. Here, the controller and the predictor blocks are assumed to be MLPs, the weight vectors of which are ${}^c\boldsymbol{w}^k$ and ${}^p\boldsymbol{w}^k$.

Evaluative Process First, the prediction error of each module is evaluated for all training sequences, defined as follows.

$$ {}^pE_i^k = \frac{1}{T}\int_0^T \left\| \boldsymbol{x}_i(t) - \tilde{\boldsymbol{x}}_i^k(t) \right\|^2 dt \tag{6} $$

Here, $\tilde{\boldsymbol{x}}_i^k(t)$ and ${}^pE_i^k$ are the output and the averaged prediction error of the kth predictor for the ith training sequence. T is the length of the sequence; thus the time average is taken from the whole sequence.

Competitive Process After the prediction errors are evaluated, the BMC is determined for each sequence. Here, the module that minimizes the prediction error is determined as the BMC of the sequence. Thus, the index of the BMC for the ith sequence is given by,

$$ *_i = \arg\min_k {}^pE_i^k. \tag{7} $$

Cooperative Process The learning rates $\{\psi_i^k\}$ are calculated by using the neighborhood function.

$$ \psi_i^k = \frac{\exp\left[-\left\| \boldsymbol{\xi}^k - \boldsymbol{\xi}_i^* \right\|^2 / 2\sigma \right]}{\sum_{i'=1}^I \exp\left[-\left\| \boldsymbol{\xi}^k - \boldsymbol{\xi}_{i'}^* \right\|^2 / 2\sigma \right]} \tag{8} $$

Here $\boldsymbol{\xi}^k$ and $\boldsymbol{\xi}_i^*$ represent the coordinates in the map space of the kth module and the BMC.

Adaptive Process The weight vectors of the predictors and the controllers are all innovated with the learning rates $\{\psi_i^k\}$, as follows.

$$ \Delta {}^p\boldsymbol{w}^k = -\eta \sum_{i=1}^I \psi_i^k \frac{\partial {}^pE_i^k}{\partial {}^p\boldsymbol{w}^k} \tag{9} $$

$$ \Delta {}^c\boldsymbol{w}^k = -\eta \sum_{i=1}^I \psi_i^k \frac{\partial {}^cE_i^k}{\partial {}^c\boldsymbol{w}^k} \tag{10} $$

Here, ${}^cE_i^k$ is the error between the kth controller and the controller specialized to the ith object. Thus,

$$ {}^cE_i^k = \frac{1}{T}\int_0^T \left\| \boldsymbol{u}_i(t) - \boldsymbol{u}_i^k(t) \right\|^2 dt. \tag{11} $$

These four processes are iterated until the network achieves a steady state. As a result, controllers having similar properties are expected to be located to near points in the map space.

2.3 Feedback Error Learning for Neural Controllers

Though the above algorithm is a natural extension of an mnSOM, further extensions are also possible by replacing with other types of neural controllers. For example, the feedback-error-learning

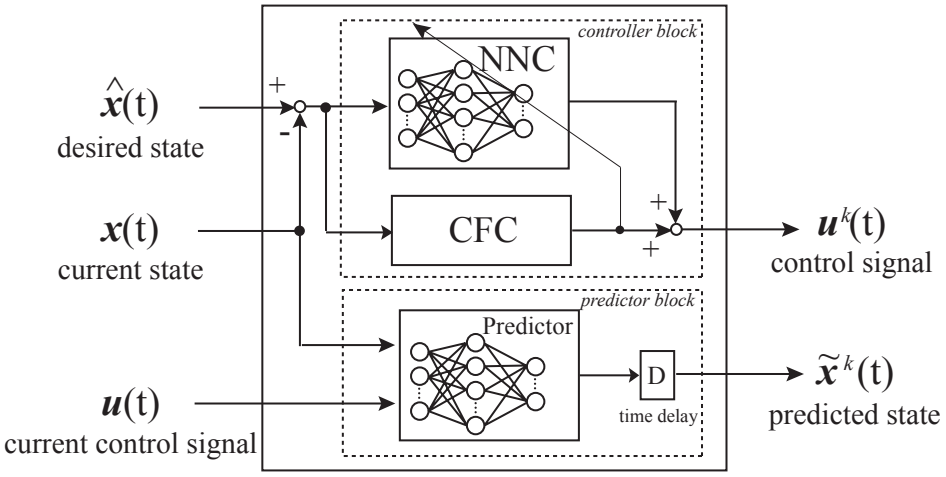

Figure 2: Block diagram of the functional module of an SOAC with feedback-error-learning

algorithm proposed by Kawato et al. can be adopted into our model [6]. The advantages of employing feedback-error-learning are that (i) the controllers can be trained using conventional feedback controllers, and there is no necessity to determine the desired controllers in advance, and (ii) the feedback-error-learning allows incremental learning for the SOAC.

The block diagram of the module is presented in Fig. 2; this is the neural controller for a closed-loop system proposed by Gomi and Kawato [7]. In this model, the controller block consists of a linear conventional feedback controller (CFC) and a nonlinear neural network controller (NNC). Here ${}^{\mathrm{cfc}}W^k$ is the feedback matrix of the CFC, whereas ${}^{\mathrm{nnc}}f(\cdot)$ means the function describing the NNC of the kth controller. The feedback matrix of the given I targets are calculated in advance, denoted as ${}^{\mathrm{cfc}}W_i$. Then the algorithm for the SOAC is rewritten as follows.

$$ {}^{\mathrm{nnc}}\boldsymbol{u}^k(t) = {}^{\mathrm{nnc}}f^k(\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)) \tag{12} $$

$$ {}^{\mathrm{cfc}}\boldsymbol{u}^k(t) = {}^{\mathrm{cfc}}W^k\,(\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)) \tag{13} $$

$$ \boldsymbol{u}^k(t) = {}^{\mathrm{nnc}}\boldsymbol{u}^k(t) + {}^{\mathrm{cfc}}\boldsymbol{u}^k(t) \tag{14} $$

These then are the definitions of the control signals, whereas the control error is redefined as follows.

$$ {}^{\mathrm{nnc}}E^k = \frac{1}{T}\int_0^T \left\| {}^{\mathrm{cfc}}\boldsymbol{u}^k(t) \right\|^2 dt \tag{15} $$

At last, the feedback matrix ${}^{\mathrm{cfc}}W^k$ and the weight vector ${}^{\mathrm{nnc}}\boldsymbol{w}^k$ are innovated in the *adaptive process* as follows.

$$ {}^{\mathrm{cfc}}W^k := \sum_{i=1}^{I} \psi_i^k\ {}^{\mathrm{cfc}}W_i \tag{16} $$

$$ \Delta^{\mathrm{nnc}}\boldsymbol{w}^k = -\eta\frac{\partial^{\mathrm{nnc}}E^k}{\partial^{\mathrm{nnc}}\boldsymbol{w}^k} \tag{17} $$

Therefore, all controllers are accompanied with corresponding CFCs, which provide the corresponding error signals to the NNCs.

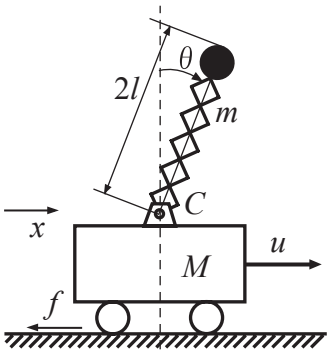

Table 1: Parameters of the inverted pendulum

l	length to the pendulum center of the mass	0.20 [m] (Short) 0.50 [m] (Half) 0.80 [m] (Long)
m	mass of the pendulum	0.10 [kg] (Light) 0.35 [kg] (Middle) 0.60 [kg] (Heavy)
M	mass of cart	5.0 [kg]
C	friction coefficient of the pendulum	4.0×10^{-4} [kgm^2/s]
f	friction coefficient of the cart	10.0 [kg/s]
g	gravity acceleration	9.8 [m/s^2]

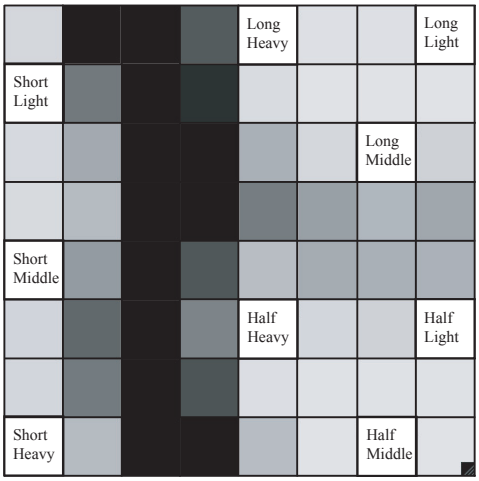

Figure 3: Map of the inverted pendulum control modules. Gray level represents the distance of predictors between neighbors

3 SOAC Applied to an Inverted Pendulum

To validate the ability of an SOAC, a simulation with an inverted pendulum was performed. The architecture presented in Fig. 2 was employed for the task. However, it was unrealistic situation, as the length and the mass of the pendulum were assumed to be variable. For the training, 9 parameter sets were chosen, i.e., combinations of 3 kinds of length of pendulum; 'Short', 'Half' and 'Long', and 3 kinds of mass of the pendulum; 'Light', 'Middle' and 'Heavy' as listed in Table 1. The feedback matrix $\{^{\mathrm{cfc}}W_i\}$ was determined by the state feedback control law. NNCs were the 3 layer MLPs, whereas the predictors were described by linear operators. In addition, external disturbance was added to the cart.

After finishing the training phase, the network was fixed and the adaptation ability was then examined. Fig. 3 shows the map generated by the SOAC after training; this was divided to 3 areas that approximately corresponded to the 3 clusters of 'Short', 'Middle' and 'Long'. To examine the adaptability, the length and mass of the pendulum was suddenly changed every 30 sec during the control task. Those values were ones not used in the training phase except for the first 30 sec. Simulations

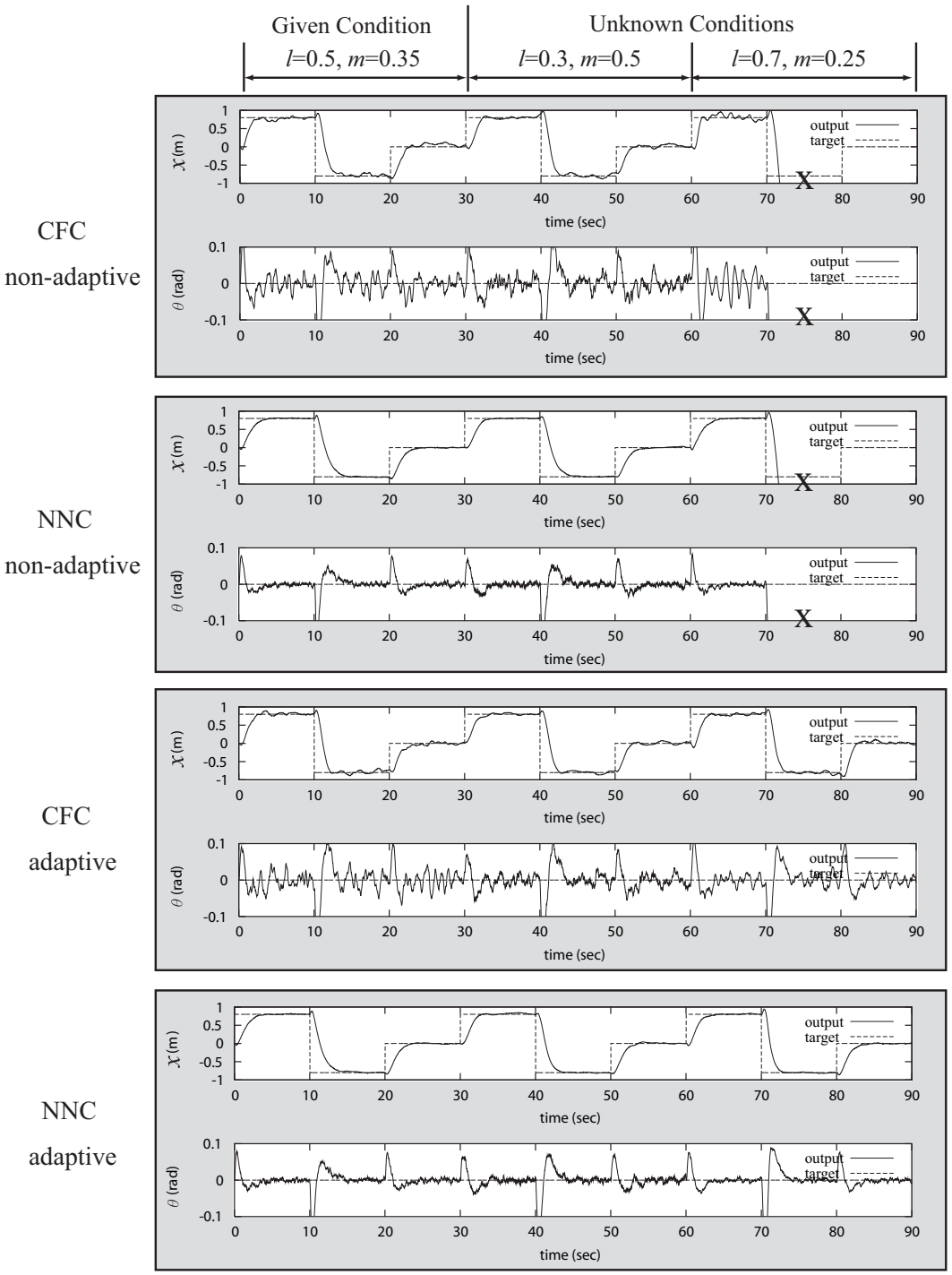

Figure 4: Simulation results of the inverted pendulum. The length and mass of the pendulum were changed every 30 sec. The parameter set of the pendulum in the first period was the same as the one used in the training phase, whereas those for the second and the third periods were unknown conditions for the SOAC. Experiments were performed with and without switching the modules, i.e., adaptive and non-adaptive conditions. In addition, either CFC or NNC was used for the task. $x(t)$ and $\theta(t)$ denote the position of the cart and the angle of the pendulum, respectively. The 'X's represent the time when the pendulum fell down

were performed with or without switching the modules, correspondingd to adaptive or non-adaptive controls, respectively. In addition, either CFC or NNC were used for the control task.

The results are shown in Fig. 4. When the pendulum was controlled by a module without switching, the pendulum fell down in both CFC and NNC cases. On the other hand, the pendulum remained up and successfully moved to the desire position when the modules were adaptively switched. Furthermore, NNCs showed better control results than CFCs. These results suggest the effectiveness of an SOAC for adaptive control tasks.

4 Conclusion

In this paper, we proposed a novel framework for an mnSOM-based adaptive controller called an SOAC; and the simulation results for an inverted pendulum suggested a high adaptation ability. An SOAC not only works as an adaptive controller, it also generates a feature map of the controlled objects. If the dynamics of the target object are continuously varied by the hidden parameters, then the generated feature map is expected to be a topological map of the hidden parameter space. Therefore, it is important how the feature map is utilized in the control task. Furthermore, incremental learning allows the controllers to be refined in parallel with controlling the unknown objects. These are areas we are currently working on.

Acknowledgements

This work was supported by a COE program (Center #J19) grant to the Kyushu Institute of Technology by the Japanese MEXT.

References

[1] Tokunaga, K., Furukawa, T., Yasui, S.: Modular network SOM: Extension of SOM to the realm of function space. Proc. of WSOM2003, 173–178, 2003

[2] Furukawa, T., Tokunaga, K., Kaneko, S., Kimotsuki, K., Yasui, S.: Generalized self-organizing maps (mnSOM) for dealing with dynamical systems. Proc. of NOLTA2004, 231–234, 2004

[3] Furukawa, T., Tokunaga, K., Morishita, K., Yasui, S.: Modular network SOM (mnSOM): From vector space to function space. Proc. of IJCNN2005, 2005 (*accepted*)

[4] Wolpert, D.M., Kawato, M.: Multiple paired forward and inverse models for motor control. Neural Networks **11**, 1317–1329, 1998

[5] Martinetz, T.M., Ritter, H.J., Schulten, K.J.:Three-dimensional neural net for learning visuomotor coordination of a robot arm. IEEE Transactions on Neural Networks **1** (1), 131–136, 1990

[6] Kawato, M., Furukawa, K., Suruzuki, R.: A hierachical neural network model for control and learning of voluntary movement. Biological Cyternetics **57**, 169-185, 1987

[7] Gomi, H., Kawato, M.: Neural network control for a closed-loop system using feedback-error-learning. Neural Networks **6**, 933–946, 1993