Paper:

A Hybrid Learning Strategy for Real Hardware of Swing-Up Pendulum

Shingo Nakamura, Ryo Saegusa, and Shuji Hashimoto

Dept. of Applied Physics, School of Science and Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
E-mail: shingo@shalab.phys.waseda.ac.jp, ryos@ieee.org, shuji@waseda.jp
[Received March 16, 2007; accepted May 28, 2007]

Generally, the bottom-up learning approaches, such as neural-network, to obtain the optimal controller of target task for mechanical system face a problem including huge number of trials, which require much time and give stress against the hardware. To avoid such problems, a simulator is often built and performed with a learning method. However, there are also problems that how simulator is constructed and how accurate it performs. In this paper, we are considering a construction of simulator directly from the real hardware. Afterward a constructed simulator is used for learning target task and the obtained optimal controller is applied to the real hardware. As an example, we picked up the pendulum swing-up task which was a typical nonlinear control problem. The construction of a simulator is performed by back-propagation method with neural-network and the optimal controller is obtained by reinforcement learning method. Both processes are implemented without using the real hardware after the data sampling, therefore, load against the hardware gets sufficiently smaller, and the objective controller can be obtained faster than using only the hardware. And we consider that our proposed method can be a basic learning strategy to obtain the optimal controller of mechanical systems.

Keywords: simulator building, reinforcement learning, neural network, swing-up pendulum

1. Introduction

Recently, many kinds of machine control methods for the nonlinear systems have been proposed. The most of machine learning methods for the complicated mechanical systems utilize the bottom-up approaches to search the optimum controller, such as genetic algorithm and neural networks, and they work very effectively [7, 9, 10]. However such approaches require a lot of trials for optimization, therefore, they are evaluated with a simulator. Since when they are implemented as the real system, the machinery must be put a harsh strain and continuously stressed, and may break down particularly on parts of precision mechanics, while a simulator never includes such issues. Furthermore, we have another advantage to use the simulator instead of real hardware such that the simulator can be run and optimized faster than using the real hardware which allows us to make it perform many trails. Thus, a simulator is considered as a powerful and useful tool for the methods which are not directly applicable for the real hardware. On the other hand, we face a problem in constructing a simulator which emulates the real hardware in high fidelity. The simulator construction is very difficult particularly for the system of which behaviors cannot be clearly and preliminarily known or may be unpredictably changed by some reasons, such as replacement of components or environment transition. Therefore a simulator should be always built and updated according to the actual system behavior simultaneously with optimization of the controller.

Many researches with a sort of pendulum have been studied and reported in the field of the nonlinear machine control. Doya [7] and Iguchi et al. [9] dealt with a swing-up pendulum problem. They used a kinematics model to perform and evaluate their proposed methods. However, they never applied them to the real system. Therefore, the effectiveness of the method is not guaranteed in the real system because their model may have the gap with the reality. Astrom et al. [6] and Yoshida [5] controlled the pendulums by the techniques based on the mechanical energy of the system, and applied them to the stabilizing problem of a pendulum on a cart. The simulators they employed were ideally built according to the physical law, and calculated the kinematics energy. However it cannot be always assumed that such inner physical parameters are given in any systems. Xi et al. [4] dealt with the swing-up problem called Furuta pendulum and applied their method to the real hardware system. They measured the physical properties with high accuracy, such as the length, mass and friction coefficients, of which values are critical for application to the real system, hence, their method works well with the real system. However, when the properties change because of aging deterioration or replacement of the components, they must be measured again or newly. Thus, such methods are very useful and effective just only on a temporary basis. Therefore, in order to make the system always work well in any cases, a simulator should be directly built and updated according

to the real hardware behavior and the objective controller should be obtained by suitable method with the updated simulator.

In this paper, we describe a novel method of machine learning using both a simulator and a real hardware, and apply it to the swing-up pendulum problem. A simulator of the hardware is directly built by learning the relationship between acquired input and output real data of the hardware and it performs with the neural-networks and the back-propagation learning method without any information of kinematics model. Herewith, we do not have to consider an arduous simulator modeling with physical law. Afterward the objective controller of the hardware is trained only with the built simulator by the reinforcement learning method. Finally, the optimum controller is applied to the real hardware and evaluated. Thus the process to obtain the optimal controller is executed through hybrid types of platform, hardware and its simulator. Therefore, even if the hardware constitution is changed, it can be possible that the optimal controller for hardware is updated according to the update of simulator.

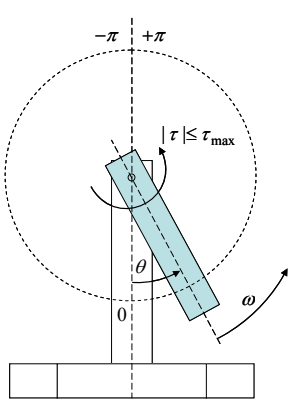


Fig. 1. Swing-up pendulum.

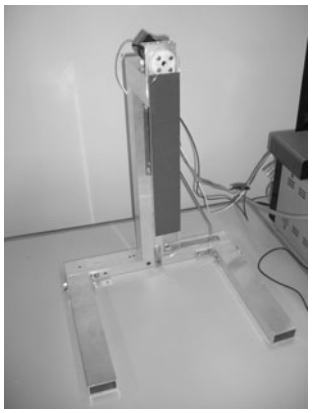


Fig. 2. Real hardware.

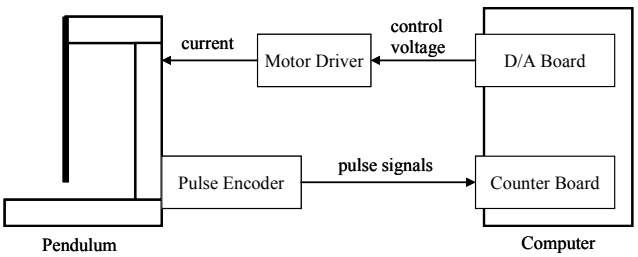


Fig. 3. Setup of pendulum system.

2. Learning Strategy

2.1. Swing-Up Pendulum Problem

To swing up pendulum with a limited torque $|\tau| \leq \tau_{max}$, is a typical nonlinear control problem. **Fig. 1** shows the swing-up pendulum model which we use in this paper. The pendulum is controlled only by torque generated at the rotational axis, and the controller can observe the angle and speed values of the pendulum. Because of its limited torque, the pendulum cannot be swung up from the most inferior position (initial position) with one push, and therefore, the controller has to swing the pendulum several times to gain enough mechanical energy to get up to the top position. Our final goal is to build the controller which can learn and control the sequential behavior of the pendulum from the initial position to the inverted standing state.

2.2. Simulator Building

In our previous work [1], we built a simulator of the pendulum from the real hardware using a genetic algorithm (GA). In this approach, we formulated a physical model of the pendulum according to the Euler-Lagrange differential equation, and the coefficients of the equation are optimized by GA. As a result, the simulator performed very efficiently when the pendulum was acting in fast speed state. However, as the speed of the pendulum motion became slower, the precision of the simulator was gradually getting worse. The reason was that the formulation did not describe the static and dynamic friction effect at once. That is to say that the formulation of the real pendulum behavior could not exactly match to the physical model. And it is too complex to express the analytical formulation. In this paper, to solve this issue, we discard an application of the differential equation based on the physical model. Alternatively, we employ the neural-network (NN), which is superior to approximate even nonlinear functions, and the NN directly estimates and outputs the next state from the current state. The details of the learning using the NN are described in section 4.

2.3. Learning of Swinging-Up Behavior

As mentioned above, the swinging-up behavior process is not simple, because of its limited torque. It means that the controller of the pendulum must learn the sequential process in the correct order to achieve the objective state. Therefore, the controller takes a lot of trials and explorations to learn it. When using a real hardware, it may need a huge number of trials, and may break down. Thus using a real hardware for learning is clearly exceptional, while using a simulator is very effective because it makes the controller learn faster time than real time without actual degradation. In this paper, the process of swinging-up behavior learning is performed only with the built simulator introducing the reinforcement learning. These details are described in section 5.

3. Real Hardware Construction

Figure 2 shows the overview of the swing-up pendulum and **Fig. 3** illustrates the setup of the real pendulum system used in this paper. The pendulum is directly attached to the axis of the geared DC-motor and its behavior is controlled only by the generated torque at the motor axis. The Motor-Driver, which can supply the current to

the motor by the Pulse Width Modulation (PWM) control, generates the torque τ according to the analog signal command from the D/A board inserted into the PC. The torque τ at the rotational axis is generated as be proportional to an analog signal voltage by the Motor-Driver. Therefore, the maximal torque $\tau_{\max}$ can be configured with the output limit voltage $V_{\max}$ of the D/A board which is not enough to swing-up the pendulum to the inverted state by just one force application.

At the back of the DC-motor, the pulse encoder is assembled. This is a device which generates some number of pulses corresponding to the moved angle value, and then the pulses are sent to the counter-board inserted into the PC. By taking count of the pulses, the system can obtain the position angle θ and calculate the rotational speed ω of the pendulum.

In this setup environment, the state of the real pendulum is measured and the behavior is controlled. The pendulum swing-up behavior is optimized for this hardware.

4. Simulator Building

The precision of a simulator is very important, since it affects the accuracy of the real hardware swing-up behavior which is obtained by the built simulator. Though it performs successfully in the simulation, the real hardware does not always perform successfully. In this section, we describe how to build a simulator using the neural network, and how the simulator performs.

4.1. Simulator with Neural Network

The neural-network (NN) is known as an information processing paradigm that is inspired by the biological nervous systems such as the animal's brain. Especially the Multi-Layered Perceptron (MLP) trained with back-propagation algorithm is effective for the approximation of a nonlinear function.

In our case, the MLP consists of three layers, i.e. input, hidden and output layer. The MLP is input three parameters, angle θ and speed ω that express the state of the pendulum, and command voltage V for the Motor-Driver from the D/A board, instead of torque τ. Then it outputs the difference values of the pendulum state parameters, $\Delta\theta$ and $\Delta\omega$ during time Δt. It means that, the MLP translate from a data set (θ, ω, V) at time t, to the data set of $(\Delta\theta, \Delta\omega)$ in Δt later.

4.2. Simulator Performance

Each neuron in the output layer is a nonlinear unit expressed with the arctangent sigmoid function whose output is limited in $[-1, 1]$. Therefore, the output value $\Delta\theta$ and $\Delta\omega$ should be normalized also to $[-1, 1]$. The angle parameter θ is already limited in $[-\pi, +\pi]$, then its normalization means dividing by π. However the speed parameter ω is not limited. Therefore we suppose the maximum value $\omega_{\max}$ and the speed ω is normalized from $[-\omega_{\max}, +\omega_{\max}]$ to $[-1, 1]$.

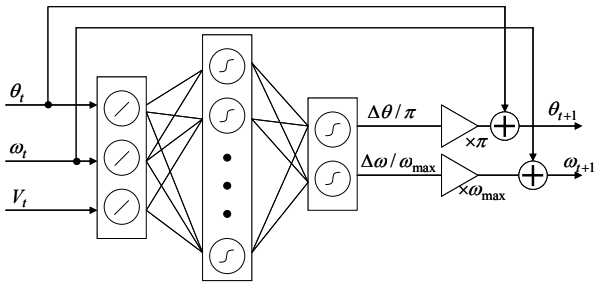


Fig. 4. Pendulum simulator with MLP.

A simulator has the MLP network inside. In the running process of the simulator, it follows the MLP. The simulator inputs the state (θ, ω, V) at time t, and obtains the normalized difference values in $[-1, 1] \times [-1, 1]$ as an output signal from the MLP. The de-normalized values $(\Delta\theta, \Delta\omega)$ is added to (θ, ω), then the state at time $t + \Delta t$ is simply represented as the state $s(\theta + \Delta\theta, \omega + \Delta\omega)$. **Fig. 4** illustrates the overview of running process of the simulator with the MLP.

In this way, the simulator simulates the total system including the electric circuit to manage the hardware. Therefore we do not have to take care of the mechanism of the real hardware, and just have to concentrate its control only with the interface of the simulator.

5. Swing Up Learning

After building a simulator, the controller for the swing-up behavior is learned and obtained only with the simulator. The controller has to swing the pendulum a few times in a correct procedure forward the final goal, i.e. the inverted standing state. In this section, we explain details about a learning method of swing-up behavior using the reinforcement learning techniques.

5.1. Reinforcement Learning

We focus on the reinforcement learning [2] to obtain the ability to swing up behavior of the pendulum. In the reinforcement learning, the action is evaluated by the given rewards, and the system empirically learns the optimum action by trials and errors. Therefore, the reinforcement learning is very robust and effective in uncertain environment, comparing with the conventional supervised learning method, such as the NN and the GA methods. In this paper, we employ the actor-critic architecture method [3] for the reinforcement learning, which is useful to output the continuous action. This architecture is composed of two modules called actor and critic. The actor module generates the action according to each state of environment, and the critic evaluates value of each state of environment and holds them. In our case, the action and state of the reinforcement learning correspond to the command voltage V and the pendulum state $s(\theta, \omega)$, respectively. The rewards are given only at the inverted standing state $s(|\theta| \approx \pi, \omega \approx 0)$.

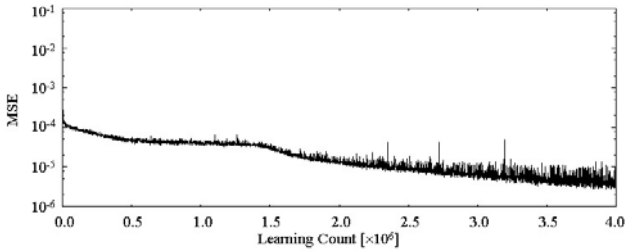


Fig. 5. Result of simulator building learning.

5.2. Update Rule

The actor module possesses the Gaussian functions for each state $s(\theta,\omega)$. This Gaussian function expresses a probability density function for mapping the state $s(\theta,\omega)$ onto the action $a(V)$, which is described as $P(a|s)$. This function has two parameters to be defined, average $\bar{V}$ and standard deviation σ. The update of the parameters is performed only when the actor module receives the reinforcement signal from the critic module. In this time, update follows as the below rule:

$$\bar{V} \leftarrow \bar{V} + \alpha(V - \bar{V}) \quad . . . . . (1)$$

$$\sigma \leftarrow \begin{cases} \beta \cdot \sigma & |V - \bar{V}| \leq \sigma \\ \beta^{-1} \cdot \sigma & \text{otherwise} \end{cases} \quad . . . . . (2)$$

where α and β are learning ratio and update ratio of standard deviation. This update rule means that the actor tends to output the act $a(V)$ which can get more reward hereafter.

The critic module possesses the evaluation value $E(s)$ for each state $s(\theta,\omega)$ by given rewards. The update is performed by the following the TD-error method. Its update rule is described as

$$E(s_t) \leftarrow E(s_t) + \alpha[r_{t+1} + \gamma E(s_{t+1}) - E(s_t)] \quad . \quad (3)$$

where α and γ are learning ratio and discount ratio, respectively. s_t and r_t indicates the state and the given reward at time t. When the evaluation value $E(s)$ is increasing on the update process, the reinforcement signal is sent to the actor module. In this update rule, the evaluation value $E(s)$ for each state s approaches the true evaluation value $E^{\pi}(s)$. In this way, the controller of the pendulum explores the optimum action for each state s to perform a swing-up behavior.

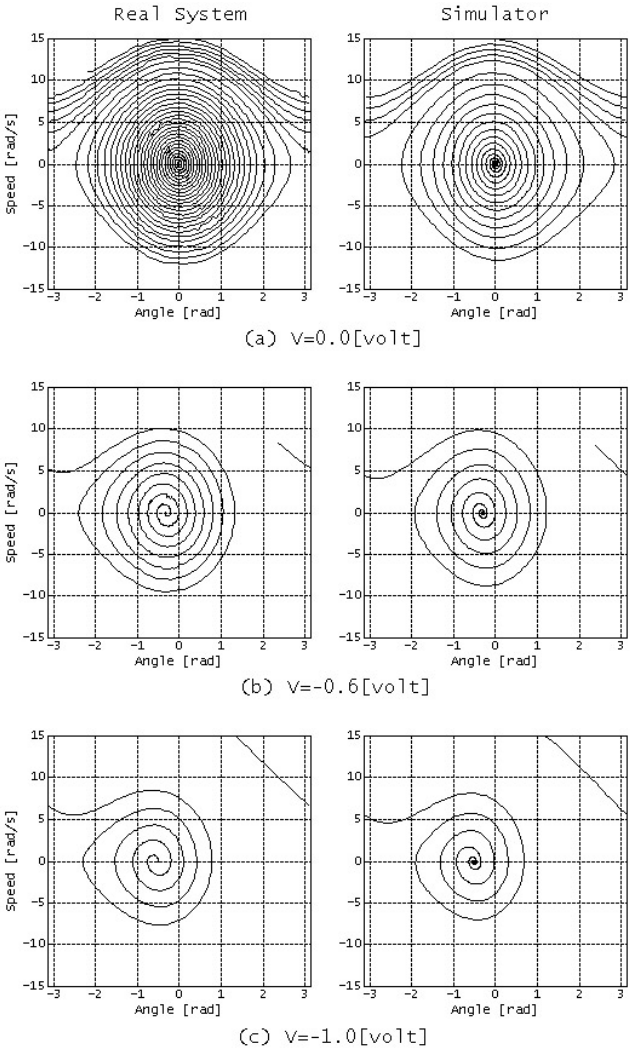


Fig. 6. Trajectories in phase space.

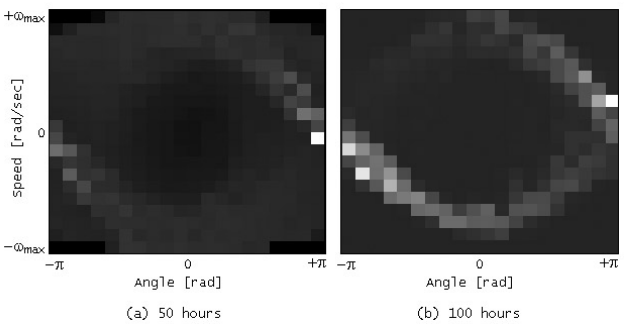


Fig. 7. Result of evaluation value in parameter space.

6. Experiments and Results

6.1. Simulator Building

The target data to train a simulator was acquired from the real pendulum. At first, we configured the maximal torque command voltage $V_{\max}$ as 1.0 volts, with which the pendulum could be lifted up to the angle around 36°. We changed the torque command voltage V which was sent to the Motor-Driver from -1.0 to $+1.0$ volts in 0.2 volt interval, and gave the acceleration by the external force with a human's hand for each voltage. Then we acquired the sequential sample data of state (θ,ω,V) with sampling interval $\Delta t = 0.01$ sec. However, the acquired data were not sampled exactly in the same interval Δt, because the acquisition operation was not performed in the correct time because of the multi-task operation system of the PC. Therefore we re-sampled the acquired data in the correct interval by the spline interpolation method of 3rd-ordered polynomials and obtained new sequential data which was sampled exactly in interval Δt. They were used as the target data to train the simulator. In addition, the speed value was obtained from the differential value of the cor-

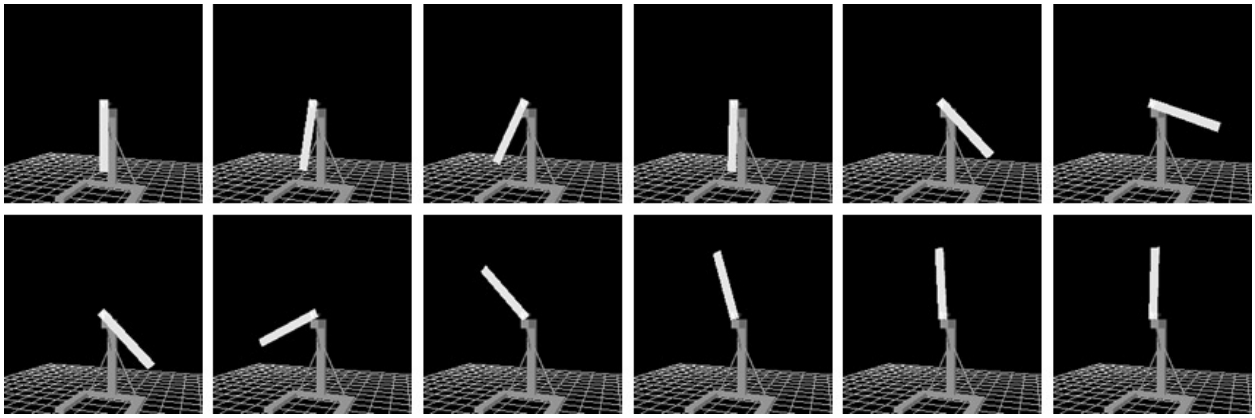


Fig. 8. Result of swing-up behavior learning with simulator.

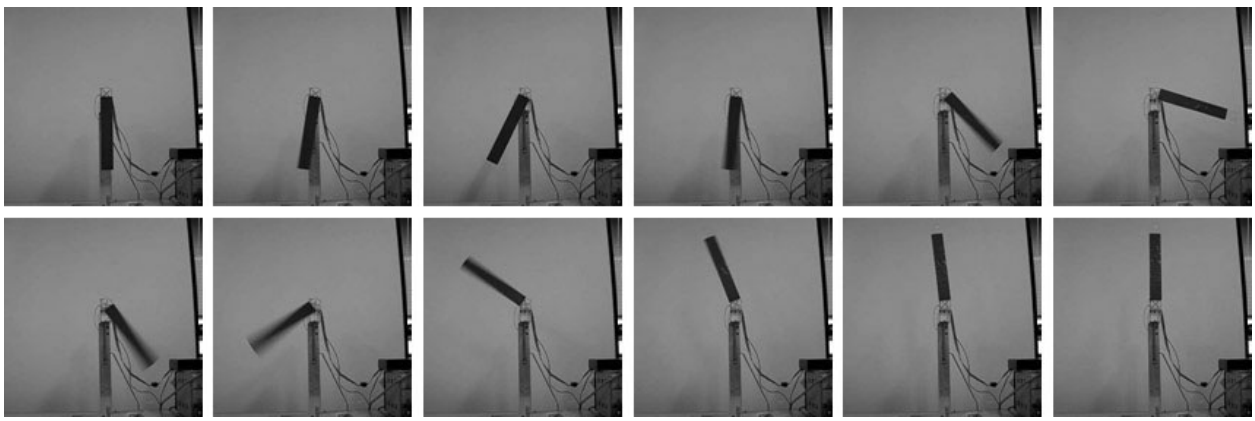


Fig. 9. Result of swing-up behavior with real system.

responding spline curve.

The learning of the NN was performed by the back-propagation method. The numbers of neurons in the input, hidden and output layer we used were 3, 10 and 2, and a target sample data set: (θ, ω, V) and $(\Delta\theta, \Delta\omega)$, were randomly selected from the re-sampled sequential data. A parameter $\omega_{\max}$ used for normalization of speed was set to 15.0 rad/sec which was given at the bottom position by the freely dropped pendulum from the top position. Learning ratio was set to 0.02, and nonlinearity ratio of nonlinear neurons in the MLP was set to 1.0. The number of iterations of the learning was 4 millions.

Figure 5 shows the mean squared error (MSE) of MLP every one thousand learning operations. The MSE value decreased until around 1.0×10^{-6}. This means that the accuracy of learning was about 0.1% in normalized space. **Fig. 6** shows the trajectories in the phase space with a variable torque voltage. The overview shape of the trajectories and the convergent angles are similar to the real pendulum in any cases, while the density of the trajectory in the case of lower torque is less in fidelity. The reason is that the NN is only evaluated with the differential values $(\Delta\theta, \Delta\omega)$. Therefore, the error is gradually accumulated and the trajectories went away from the true trajectory of the real pendulum. However, the trajectories represent the movements of the pendulum well in any voltage.

6.2. Learning Swing-Up Behavior

The swing-up behavior learning was performed with the obtained simulator in the previous section. The state parameter space $[-\pi, +\pi] \times [-\omega_{\max}, +\omega_{\max}]$ is divided in 20×20 at even intervals and treated as a discrete space. When the torque command voltage V the actor outputs is more than $V_{\max}$, it is set that $V = V_{\max}$ as saturation. If the state of the pendulum goes out of the parameter space, i.e. $|\omega| > \omega_{\max}$, we reset both the angle and speed to 0. Such operation can be executed only in the simulation environment, though it is impossible with the real hardware. The reinforcement learning performed according to the actor-critic update rule with interval 0.1 seconds for 100 hours in the simulator world. The parameters α, β and γ, used for the update by the formula (1) and (2), are set to 0.1, 0.9, and 0.9, respectively, they were determined by some preliminary experiments. The rewards are given only when the pendulum is in the state $s(|\theta| > 3.0, |\omega| < 0.2)$.

The learning process finished in 4 minutes at most. **Fig. 7** shows the halfway and final resultant evaluation value in the critic module for each divided parameter space, where the light color means higher value and dark one means lower value. As we expected, the highest value was located around the inverted standing state $s(|\theta| \approx$

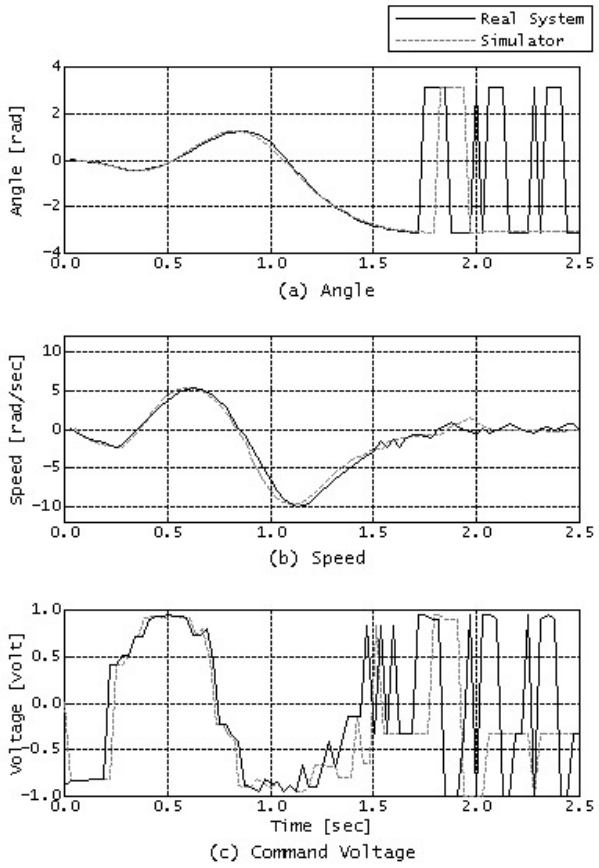


Fig. 10. Result of parameter time-shifts.

$\pi, \omega \approx 0$), and the values were high in the state where the mechanical energy was same as the inverted standing state. Meanwhile, as the state gets farther from the standing state, the values are gradually lower. This means that the controlled pendulum stocks the enough mechanical energy in advance, and then approaches the goal state. In the final resultant **Fig. 7(b)**, the under half area has higher values more than upper one. Therefore the controller finally recognizes that the crock-wise movement is better to be close to the goal state, and the inverted state is reached always from crock-wise direction. **Fig. 8** shows the conclusive sequential images of the swing-up behavior with the simulator from the left-top to the right-bottom. They prove that the learning was performed correctly.

6.3. Application to Real Hardware

Finally, we applied the learning result obtained in the previous section to the real hardware to control the swing-up behavior. **Fig. 9** shows that the sequential images of the swing up behavior by the real hardware. It can be seen that the real pendulum acts almost in the same way as the simulated result. **Fig. 10** shows the time-shift data of the angle θ, speed ω and torque voltage V of the swing-up behavior with the real hardware and simulator. The solid lines indicate the results of the real pendulum system, and the dashed lines indicates the simulator's ones. It can be confirmed that the pendulum was swung a few times, and after that it kept the inverted standing state. Comparing with the simulator, the real system performs as to trace it. This means that the simulator is built with a satisfactory precision and it has an enough capability to realize the real pendulum.

7. Conclusion

We proposed a novel method of swing up behavior learning associating with a real hardware and its simulator. The simulator was constructed by the neural network trained with the actually acquired data from the real hardware without information of the physical law. The simulator accuracy was estimated with MSE value and it was effectively getting small. Therefore the simulator could cross the gap with the real hardware. Afterward, the swing-up control for the real hardware is learned only through the built simulator by the reinforcement learning method. By using the simulator, the learning could be finished much faster than using the real pendulum without stress. Moreover, we confirmed that, by implementation of the optimum controller obtained from simulator for the real pendulum, the real pendulum swung a few times and finally reached to the inverted standing state.

For future works, we are considering the hybrid strategy using both the real hardware and the simulator for the robot learning. In this paper, we regard that the physical parameters of the real pendulum, such as a friction coefficient, a mass of pendulum and the gravitational constant, are never changed in the environment. Therefore, if they change by the aged deterioration or some physical accidents, the results of learning should be discarded and the system learns again from the first. However, if the simulator always observes the real pendulum movements and learns the relationship between acts and its results even during the system is executing the objective tasks, the system can search the optimal controller for objective behaviors with the simulator again. This cycle can make the system flexible against the changing and unpredictable environment.

Acknowledgements

This work was supported in part by the following funds: (i) "Establishment of Consolidated Research Institute for Advanced Science and Medical Care," Encouraging Development Strategic Research Centers Program, the Special Coordination Funds for Promoting Science and Technology, Ministry of Education, Culture, Sports, Science and Technology, Japan, (ii) "The innovative research on symbiosis technologies for human and robots in the elderly dominated society," 21st Century Center of Excellence (COE) Program, Japan Society for the Promotion of Science, and (iii) the Grant-in-Aid for the WABOT-HOUSE Project by Gifu Prefecture.

References:

[1] M. F. Speider, S. Nakamura, and S. Hashimoto, "Crossing the reality gap for a swing-up pendulum," Proc. of the 2006 IEICE General Conf., CD-Proc, D-2-12, 2006.

[2] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," A Bradford Book, The MIT Press, 1988.
[3] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," IEEE Trans. Syst. Man. & Cybern, Vol.SMC-13, pp. 835-846, 1983.
[4] Y. Xu, M. Iwase, and K. Furuta, "Time Optimal Swing-up Control of Single Pendulum," Trans. of ASME: Journal of Dynamics Systems, Measurement and Control, Vol.123, No.5, pp. 518-527, 2001.
[5] K. Yoshida, "Swing-up control of an inverted pendulum by energy-based methods," Proc. of the American Control Conf. 1999, pp. 4045-4047, 1999.
[6] K. J. Astrom and K. Furuta, "Swing-up a pendulum by a energy control," Automatica, Vol.36, pp. 287-295, 2000.
[7] K. Doya, "Efficient Nonlinear Control with Actor-Tutor Architecture," Advances in Neural Information Processing System, 9, pp. 1012-1018, 1996.
[8] M. Bugeja, "Non-linear swing-up and stabilizing control of an inverted pendulum system," Proc. IEEE Region 8 EUROCON 2003, 2003.
[9] K. Iguchi, H. Kimura, and S. Kobayashi, "GA-based Control for Swinging up and Stabilizing Parallel Double Inverted Pendulums," Proceedings of the 13th SICE Symposium on Decentralized Autonomous Systems, pp. 277-282, 2001 (in Japanese).
[10] K. Doya, K. Samejima, K. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," Neural Comput., Vol.14, No.6, pp. 1347-1369, 2002.

Name:
Ryo Saegusa

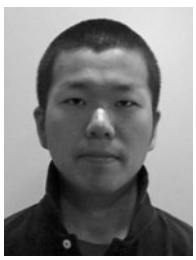
Affiliation:
Post Doctor Researcher, The Italian Institute of Technology
Visiting Researcher, Humanoid Robotics Institute, Waseda University

Address:
Via Morego 30, 16163 Genova, Italy

Brief Biographical History:
2004- Waseda University, Japan
2007- The Italian Institute of Technology, Italy

Main Works:
• R. Seagusa, H. Sakano, and S. Hashimoto, "Nonlinear Principal Component Analysis to Preserve the Order of Principal Components," Neurocomputing, No.61, pp. 57-70, 2004.

Membership in Academic Societies:
• The Institute of Electrical and Electronic Engineers (IEEE)
• The Institute of Electronics, Information and Communication Engineers (IEICE)



Name:
Shingo Nakamura

Affiliation:
Visiting Research Associate, Waseda University

Address:
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

Brief Biographical History:
2005- Visiting Research Associate, Waseda University

Main Works:
• "A learning strategy using simulator for real hardware of swing-up pendulum," Proc. of SCIS&ISIS 2006, pp. 971-976, 2006.

Membership in Academic Societies:
• The Institute of Electronics, Information and Communication Engineers (IEICE)

Name:
Shuji Hashimoto

Affiliation:
Professor, Faculty of Science and Engineering, Waseda University

Address:
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

Brief Biographical History:
1991- Associate Professor, Waseda University
1993- Professor, Waseda University
2000- Director, Humanoid Robotics Institute, Waseda University

Main Works:
• S. Hashimoto, S. Narita, H. Kasahara, et al., "Humanoid robots in Waseda University – Hadaly-2 and WABIAN," Autonomous Robot, Vol.12, No.1, pp. 25-38, 2002.

Membership in Academic Societies:
• The Institute of Electronics, Information and Communication Engineers (IEICE)
• Information Processing Society of Japan (IPSJ)
• The Society of Instrument and Control Engineers (SICE)