Remarks on an Adaptive-type Self-tuning Controller using Quantum Neural Network with Qubit Neurons

Kazuhiko Takahashi, Yuka Shiotani and Masafumi Hashimoto
Faculty of Science and Engineering, Doshisha University, Kyoto, Japan

Keywords: Quantum Neural Network, Qubit neuron, Self-tuning Controller, PID Controller, Fuzzy Logic Controller.

Abstract: This paper presents a self-tuning controller based on a quantum neural network and investigates the controller's characteristics for control systems. A multi-layer quantum neural network which uses qubit neurons as an information processing unit is utilized to design an adaptive-type self-tuning controller which conducts the training of the quantum neural network as an online process. As an example of designing the self-tuning controller, either a proportional integral derivative controller or a fuzzy logic controller is utilized as a conventional controller for which parameters are tuned by the quantum neural network. To evaluate the learning performance and capability of the adaptive-type quantum neural self-tuning controller, we conduct computational experiments to control the single-input single-output non-linear discrete time plant. The results of the computational experiments confirm both feasibility and effectiveness of the proposed self-tuning controller.

1 INTRODUCTION

Over the past quarter of the century, many studies conducted worldwide have applied both flexibility and learning ability of artificial neural networks to control systems and have proposed many types of neural-network-based control systems (Hagan et al., 2002)(Meireles et al., 2003). Neural networks, which were utilized in a large number of previous studies in the field of control systems, conduct signal processing involving real numbers by using a sigmoid, binary or radial basis function as an information processing unit. On the other hand, there are several advantages to solving classically hard-to-treat, intractable problems by using real-valued (conventional) neural networks and to providing a new understanding of certain brain functions. Therefore, many studies of hyper-complex numbers neural networks based on Clifford algebra (Sommer, 2001), such as complex neural networks whose weights and activation functions are complex and quaternion neural networks developed in hypercomplex quaternion algebra, have been undertaken, and there have been many successful examples involving the use of such neural networks in applications requiring spatial processing, e.g. colour image processing and multiple-dimension time-series signal processing. Quantum neural networks (Ezhov and Ventura, 2000)(Manju and Nigam, 2012), which involve the introduction of quantum theoretical concepts and quantum computing techniques to neural networks, can also be classified as complex neural networks because the state of an arbitrary neuron in the quantum neural network is a coherent superposition of multiple quantum states which can be expressed by complex numbers. A quantum neural network which utilizes qubit-inspired neurons as information processing units has been proposed, and its high learning capability has been confirmed in several benchmark tests and applications (Kouda et al., 2005)(Zhou et al., 2006). As a servo-level controller application which uses the quantum neural network with qubit neurons, a direct controller in which the output of the quantum neural network is the control input of the object plant has been proposed and its feasibility demonstrated (Takahashi et al., 2011).

This paper proposes an adaptive-type self-tuning controller by using a quantum neural network, and investigates its characteristics for control systems. In the self-tuning controller, the control input of the plant is the output from the conventional controller whose parameters are tuned by the quantum neural network. Although the self-tuning controller is more complex than the direct controller, it offers the possibility of realizing increased robustness. The training of the quantum neural network can be classified into two types: online training and offline training. From the perspective of control applications, online training, which corresponds to adaptive con-

Takahashi K., Shiotani Y. and Hashimoto M..
Remarks on an Adaptive-type Self-tuning Controller using Quantum Neural Network with Qubit Neurons.
DOI: 10.5220/0004425701070112
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 107-112
ISBN: 978-989-8565-70-9

trol, is more practical than offline training, and this study, therefore, considers the online training of the quantum neural network. In Section 2, we present a design of the quantum-neural-network-based self-tuning controller (hereafter called quantum neural self-tuning controller), which conducts online training of the multi-layer quantum neural network with qubit neurons. In Section 3, computational experiments for controlling a discrete-time non-linear plant are conducted to evaluate the feasibility of the proposed controller.

2 CONTROLLER DESIGN

Figure 1 shows a schematic of a self-tuning controller for which the control input $\boldsymbol{u}(k)$ is calculated by a conventional controller for which parameters are tuned by the quantum neural network. The multi-layer quantum neural network is designed by combining qubit neurons in layers. The state of the j-th qubit neuron in the r-th layer at a sampling number of k can be given by:

$$\begin{cases} z^r_j(k) = f(\frac{\pi}{2}\delta^r_j(k) - \arg v^r_j(k)) \\ v^r_j(k) = \sum_i f(\theta^r_{i,j}(k)) f(z^{r-1}_i(k)) - f(\lambda^r_j(k)) \end{cases} . \quad (1)$$

Here, $f(\cdot)$ is a complex function of the phase which expresses the qubit state $|\psi\rangle$: $f(\phi) = e^{i\phi}$ (i is an imaginary unit), $\delta^r_j(k)$ is the reversal parameter corresponding to a 2-bit controlled NOT gate, $\theta^r_{i,j}(k)$ is the phase parameter corresponding to the phase of a 1-bit rotation gate and $\lambda^r_j(k)$ is the threshold parameter. In the input layer ($r = I$), the network input $x_j(k)$ is first converted into the quantum state with a phase in the range $[0, \pi/2]$, and then the output, given by $z^I_j(k) = f((\pi/2)\sigma(x_j(k)))$, is fed into qubit neurons in the hidden layer ($r = H$). Here $\sigma(\cdot)$ is a sigmoid function: $\sigma(x) = 1/(1+e^{-x})$. The outputs from the qubit neurons of the hidden and output layers are given by Eq. (1). By considering the probability of the state in which $|1\rangle$ is observed from the j-th qubit neuron in the output layer ($r = O$), the output from the quantum neural network $u_{qnn_j}(k)$ is defined as follows:

$$u_{qnn_j}(\omega(k), x(k)) = |\mathrm{Im}(z^O_j(k))|^2. \quad (2)$$

Here, the vector $\omega(k)$ is composed of parameters $\theta^r_{i,j}(k)$, $\delta^r_j(k)$ and $\lambda^r_j(k)$, and the vector $\boldsymbol{x}(k)$ is composed of input $x_j(k)$. The output from the quantum neural network $u_{n_j}(k)$ is converted from the range $[0, 1]$ into the range $[u_{min}, u_{max}]$ with a gain factor g_{0_j} and shift factor $u_{qnn_{0_j}}$:

$$u_{n_j}(k) = g_{0_j}\{u_{qnn_j}(\omega(k), x(k)) - u_{qnn_{0_j}}\}. \quad (3)$$

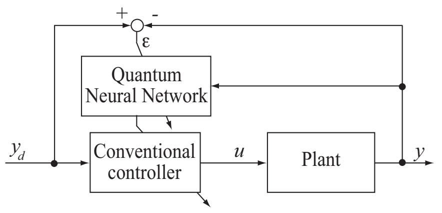

Figure 1: Schematic diagram of the quantum neural self-tuning controller.

To simplify the quantum neural self-tuning controller's design, the following single-input single-output (SISO) discrete-time plant is considered as a controlling target plant:

$$\begin{aligned} y(k+d) = F(&y(k), y(k-1), \cdots, y(k-n+1), \\ &u(k), u(k-1), \cdots, u(k-m-d+1)), \end{aligned} \quad (4)$$

where $y(k)$ is the plant output, $u(k)$ is the plant input, n and m are the plant orders, d is the dead time of the plant and $F(\cdot)$ is the function which expresses plant dynamics. This design makes the following assumptions: the upper limit orders and dead time of the plant are known. By considering the desired plant output $y_d(k)$, the output error $\varepsilon(k)$ can be defined by $\varepsilon(k) = y_d(k) - y(k)$. In this study, the following digital proportional integral derivative (PID) control law is utilized as the conventional controller:

$$u(k) = u(k-1) + g^{\mathrm{T}}(k)e(k), \quad (5)$$

where $\boldsymbol{g}(k) = [\ K_P(k) \quad K_I(k) \quad K_D(k)\ ]^{\mathrm{T}}$, in which $K_P(k)$, $K_I(k)$ and $K_D(k)$ are the proportional gain, integral gain and differential gain, respectively, $\boldsymbol{e}(k) = [\ \Delta\varepsilon(k) \quad \varepsilon(k) \quad \Delta\varepsilon(k) - \Delta\varepsilon(k-1)\ ]^{\mathrm{T}}$, in which $\Delta\varepsilon(k)$ is the difference of the output error given by $\Delta\varepsilon(k) = \varepsilon(k) - \varepsilon(k-1)$. The gain parameters are tuned by the output from the quantum neural network in the quantum neural self-tuning controller.

To define the input vector of the quantum neural network, the following direct controller which controls the plant represented by Eq. (4) is utilized:

$$u_d(k) = F_d(w(k), x_d(k)), \quad (6)$$

where $u_d(k)$ is the output from the direct controller, $F_d(\cdot)$ is the function of the direct controller, the vector $\boldsymbol{w}(k)$ is composed of the direct controller's parameters and $\boldsymbol{x_d}(k)$ is the input vector of the direct controller, which is defined as follows:

$$\begin{aligned} x_d(k) = [\ &y_d(k+d) \quad y(k) \quad \cdots \quad y(k-n+1) \\ &u(k-1) \quad \cdots \quad u(k-m-d+1)\ ]^{\mathrm{T}}. \end{aligned} \quad (7)$$

Assuming $u(k) - u(k-1) = u_d(k)$ yields

$$g(k) = \{e(k)e^{\mathrm{T}}(k) + \Gamma\}^{-1} e(k) F_d(w(k), x_d(k)), \quad (8)$$

where $\Gamma = diag(\gamma_1, \gamma_2, \gamma_3)$, in which γ_i $(i = 1,2,3)$ is an arbitrary constant $(0 < \gamma_i \ll 1)$. By considering the right hand side of Eq. (8), the input vector of the quantum neural network $\boldsymbol{x}(k)$ is defined as follows:

$$\begin{aligned} x(k) = [\ & y_d(k+d) \quad y(k) \quad \cdots \quad y(k-n+1) \\ & u(k-1) \quad \cdots \quad u(k-m-d+1) \quad \gamma_2\gamma_3\Delta\varepsilon(k) \\ & \gamma_1\gamma_3\varepsilon(k) \quad \gamma_1\gamma_2\{\Delta\varepsilon(k) - \Delta\varepsilon(k-1)\} \\ & \det(e(k)e^{\mathrm{T}}(k) + \Gamma)\ ]^{\mathrm{T}}. \end{aligned} \tag{9}$$

The training of the quantum neural network is carried out by using a back-propagation algorithm such that it minimizes the cost function $J(k)$:

$$\omega(k+1) = \omega(k-d) - \eta\frac{\partial J(k)}{\partial\omega(k-d)} + \alpha\Delta\omega(k-d), \tag{10}$$

$$J(k) = \frac{1}{2}\sum_q \varepsilon_q^2(k), \tag{11}$$

where $\varepsilon_q(k) = y_{d_q}(k) - y_q(k)$, $\Delta\omega(k)$ is the weight increment and η and α are the learning factors. The gradients of the cost function with respect to the parameters in the output layer are as follows:

$$\begin{aligned} \frac{\partial J(k)}{\partial\delta_j^O(k-d)} &= -\frac{\pi}{2}g_{0_j}\sum_q \varepsilon_q(k)\sum_s \frac{\partial\varepsilon_q(k)}{\partial y_s(k)} \\ &\quad \times\sum_t \frac{\partial y_s(k)}{\partial u_t(k-d)}\frac{\partial u_t(k-d)}{\partial u_{n_j}(k-d)}\xi_j(k-d), \\ \frac{\partial J(k)}{\partial\theta_{i,j}^O(k-d)} &= g_{0_j}\sum_q \varepsilon_q(k)\sum_s \frac{\partial\varepsilon_q(k)}{\partial y_s(k)} \\ &\quad \times\sum_t \frac{\partial y_s(k)}{\partial u_t(k-d)}\frac{\partial u_t(k-d)}{\partial u_{n_j}(k-d)}\frac{\xi_j(k-d)}{d_j^{OH}(k-d)} \\ &\quad \times\{a_j^{OH}(k-d)\cos(\theta_{i,j}^O(k-d) + v_i^H(k-d)) \\ &\quad + b_j^{OH}(k-d)\sin(\theta_{i,j}^O(k-d) + v_i^H(k-d))\}, \\ \frac{\partial J(k)}{\partial\lambda_j^O(k-d)} &= -g_{0_j}\sum_q \varepsilon_q(k)\sum_s \frac{\partial\varepsilon_q(k)}{\partial y_s(k)} \\ &\quad \times\sum_t \frac{\partial y_s(k)}{\partial u_t(k-d)}\frac{\partial u_t(k-d)}{\partial u_{n_j}(k-d)}\frac{\xi_j(k-d)}{d_j^{OH}(k-d)} \\ &\quad \times\{a_j^{OH}(k-d)\cos\lambda_j^O(k-d) \\ &\quad + b_j^{OH}(k-d)\sin\lambda_j^O(k-d)\}, \end{aligned}$$

where $\partial y_s(k)/\partial u_t(k-d)$ is the Jacobian of the plant, $\partial u_t(k)/\partial u_{n_j}(k)$ is the Jacobian of the controller,

$$\begin{aligned} \xi_j(k) &= 2|\sin v_j^O(k)|\mathrm{sgn}(\sin v_j^O(k))\cos v_j^O(k), \\ a_j^{rr'}(k) &= \sum_i \cos(\theta_{i,j}^r(k) + v_i^{r'}(k)) - \cos\lambda_j^r(k), \\ b_j^{rr'}(k) &= \sum_i \sin(\theta_{i,j}^r(k) + v_i^{r'}(k)) - \sin\lambda_j^r(k), \\ d_j^{rr'}(k) &= (a_j^{rr'}(k))^2 + (b_j^{rr'}(k))^2, \end{aligned}$$

and sgn$(\cdot)$ is the sign function. In the same manner, the gradients of the cost function with respect to the parameters in the hidden layer are as follows:

$$\begin{aligned} \frac{\partial J(k)}{\partial\delta_j^H(k-d)} &= \frac{\pi}{2}\sum_q \frac{\partial J(k)}{\partial\theta_{j,q}^O(k-d)}, \\ \frac{\partial J(k)}{\partial\theta_{i,j}^H(k-d)} &= -\sum_q \frac{\partial J(k)}{\partial\theta_{j,q}^O(k-d)}\frac{1}{d_j^{HH-1}(k-d)} \\ &\quad \times\{a_j^{HH-1}(k-d)\cos(\theta_{i,j}^H(k-d) + v_i^{H-1}(k-d)) \\ &\quad + b_j^{HH-1}(k-d)\sin(\theta_{i,j}^H(k-d) + v_i^{H-1}(k-d))\}, \\ \frac{\partial J(k)}{\partial\lambda_j^H(k-d)} &= \sum_q \frac{\partial J(k)}{\partial\theta_{j,q}^O(k-d)}\frac{1}{d_j^H(k-d)} \\ &\quad \times\{a_j^{HH-1}(k-d)\cos\lambda_j^H(k-d) \\ &\quad + b_j^{HH-1}(k-d)\sin\lambda_j^H(k-d)\}. \end{aligned}$$

When the unit in the hidden layer links to that in the input layer, $v_i^{H-1}(k)$ is replaced with $z_i^I(k)$.

3 NUMERICAL EXPERIMENTS

The quantum neural self-tuning controller was numerically investigated using the following SISO discrete-time non-linear plant:

$$\begin{aligned} y(k+1) = F_s[&-\sum_{i=1}^{2} a_i y(k-i+1) + a_3 y(k-2) \\ &+ \sum_{i=1}^{2} b_i u(k-i+1) + c_{non}y^2(k)], \end{aligned} \tag{12}$$

where a_3 is the coefficient of the parasitic term, c_{non} is the coefficient of the non-linear term and the function $F_s(\cdot)$ has the non-linear characteristic of saturation: $F_s(x) = 1(x \geq 1); x(-1 < x < 1); -1(x \leq -1)$. In the computational experiments, the plant parameters were set to $a_1 = -1.3$, $a_2 = 0.3$, $b_1 = 1$, $b_2 = 0.7$, $a_3 = 0.03$ and $c_{non} = 0.2$ (Yamada, 2011). The desired plant output $y_d(k)$ was set as a rectangular wave in order to take account of frequency richness. The number of samples within one period of the rectangular wave was 100, and the amplitude of the wave was ± 0.5. To design the quantum neural self-tuning controller, the plant was assumed to be a linear second-order plant: $d = 1$, $n = 2$, $m = 1$, $q = 1$, $s = 1$ and $t = 1$. The quantum neural network was an 8–24–3 network topology. The constant values in the input vector were $\gamma_1 = 0.01$, $\gamma_2 = 0.1$ and $\gamma_3 = 0.01$. The gain and shift factors were $g_{0_j} = 1$ and $u_{qnn_{0_j}} = 0$, respectively. The Jacobian of the plant was assumed to be 1, and its magnitude and sign were adjusted by the learning factor η. The Jacobian of the controller was derived from Eq. (5): $\partial u(k)/\partial u_n(k) =$

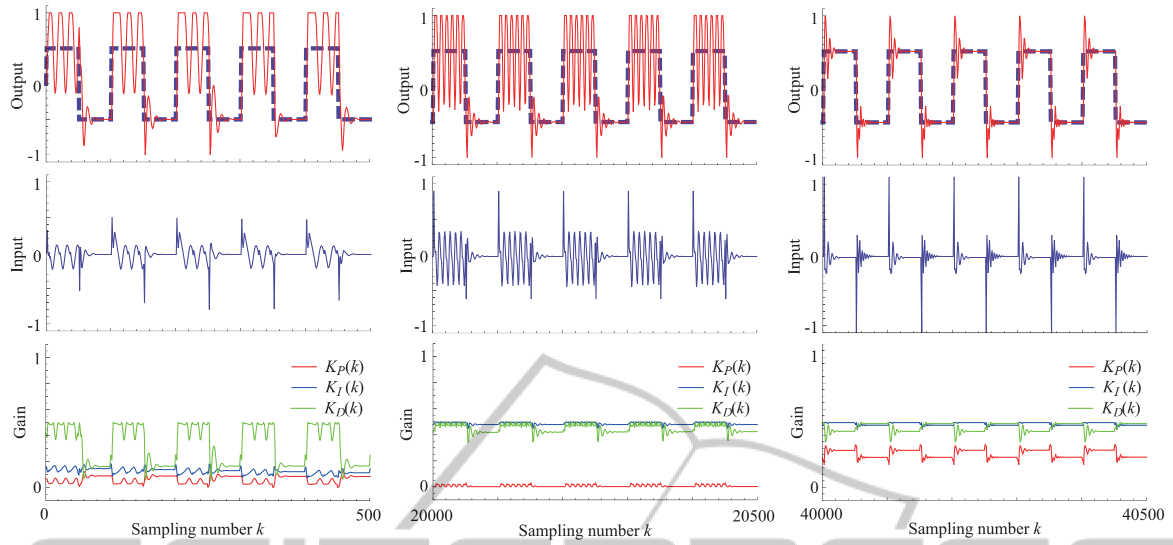

Figure 2: Example of the adaptation process controlled by the quantum neural self-tuning controller (top: plant output where the dotted line indicates the desired plant output $y_d(k)$, and the rigid line indicates the plant output $y(k)$; middle: control input from the PID controller; bottom: output from the quantum neural network where the red line indicates $K_P(k)$, the blue line indicates $K_I(k)$ and the green line indicates $K_D(k)$).

$[\ \Delta\varepsilon(k) \quad \varepsilon(k) \quad \Delta\varepsilon(k)-\Delta\varepsilon(k-1)\ ]$. The initial values of $\theta^r_{i,j}(0)$ and $\lambda^r_j(0)$ were randomly selected from the interval $[-\pi,\pi]$, and the initial value of $\delta^r_j(0)$ was set to 0.5. The learning factors were $\eta = 10^{-4}$ and $\alpha = 0.9$.

Figure 2 shows examples of the plant responses, and Fig. 3 indicates the normalized cost function during the adaptation process of the quantum neural self-tuning controller. In Fig. 3, the horizontal axis represents the number of periods of the desired plant output, and the vertical axis represents the normalized cost function averaged within one period of the desired plant output. The normalized cost function decreased as the training progressed. Although the overshoot and residual vibration are observed in the plant output when the desired plant output changes rapidly, the quantum neural self-tuning controller can achieve the control task of making the non-linear plant follow the desired plant output by tuning the gain parameters with the output from the quantum neural network, as shown in Fig. 2. These results indicate the feasibility of the quantum neural self-tuning controller; however, the control performance depends on the PID control law. To investigate the robustness of the quantum neural self-tuning controller, other desired plant outputs (e.g. a rectangular wave with various periods and amplitudes and a sinusoidal wave) are tested. Here the initial values of the parameters $\theta^r_{i,j}(0)$, $\lambda^r_j(0)$ and $\delta^r_j(0)$ are the trained parameters obtained after the 1000 periods shown in Fig. 3. Figure 4 shows the response of the plant to the untrained desired plant

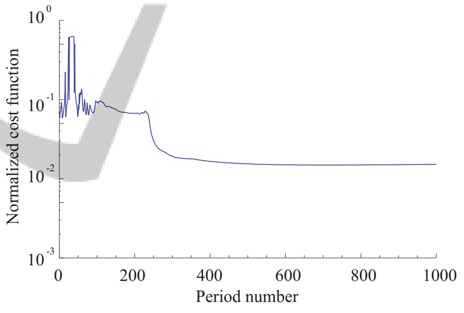

Figure 3: Relationship between the period number of the desired plant output and the normalized cost function (conventional controller: PID controller).

output. Using the learning capability of the quantum neural self-tuning controller, the plant output tracks the untrained desired plant output.

To demonstrate the extension of designing the quantum neural self-tuning controller, a fuzzy logic controller is considered as the conventional controller. Here the fuzzy logic controller is a velocity-type controller, the output of which is the increment of the control input $\Delta u(k)$ synthesized using the output error $\varepsilon(k)$ and its difference $\Delta\varepsilon(k)$. Rutherford's control rule is utilized as the fuzzy logic control rule in which bell-shaped-type (Gaussian) membership functions are used in the antecedent part, and singleton type membership functions are used in the consequent part. The i-th fuzzy logic control rule can be expressed by 'If $\varepsilon(k)$ is A_i and $\Delta\varepsilon(k)$ is B_i then $\Delta u(k)$ is c_i', where A_i and B_i are fuzzy variables in the antecedent part, and c_i is the position

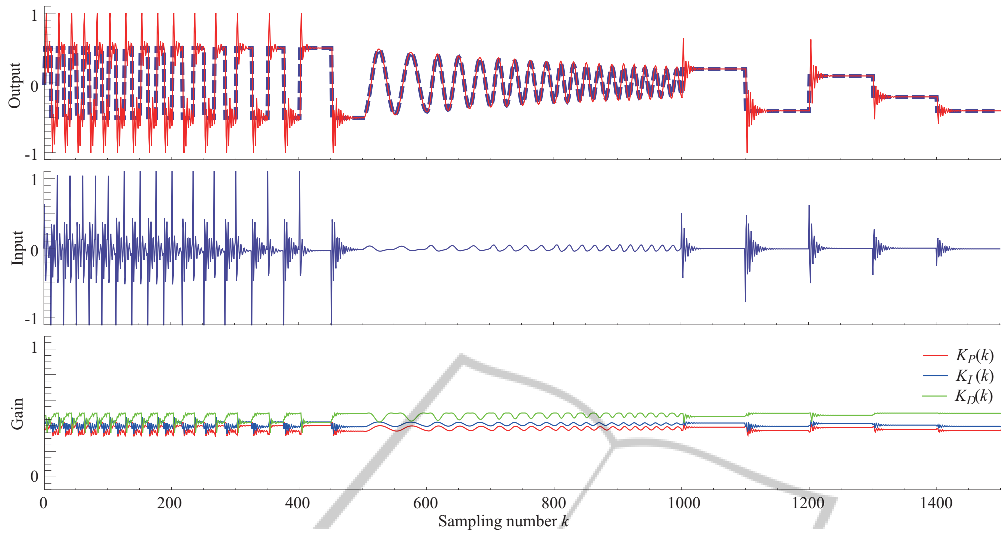

Figure 4: Response of the plant controlled by the quantum neural self-tuning controller to untrained desired plant output (top: plant output where the dotted line indicates the desired plant output $y_d(k)$, and the rigid line indicates the plant output $y(k)$; middle: control input; bottom: output from the quantum neural network where the red line indicates $K_P(k)$, the blue line indicates $K_I(k)$ and the green line indicates $K_D(k)$).

of the singleton type in the consequent part. The fuzzy logic controller can be represented by $\Delta u(k) = C_{flc}(\varepsilon(k), \Delta\varepsilon(k), K_\varepsilon, K_{\Delta\varepsilon}, K_{\Delta u})$, where $C_{flc}(\cdot)$ denotes the fuzzy relation defined by the fuzzy logic control rule, and K_i ($i = \varepsilon, \Delta\varepsilon, \Delta u$) represents the appropriate scale factors. In the antecedent and consequent parts, the support of a fuzzy set has seven partitions which are normalized in the range $[-6, 6]$ by the scale factors. Using a simple fuzzy inference with the settled values $\varepsilon^o(k)$ and $\Delta\varepsilon^o(k)$, the output from the fuzzy logic controller is calculated by $\Delta u^o(k) = \sum_i \zeta_i c_i / \sum_i \zeta_i$, where ζ_i is a fitness value in the antecedent part given by $\zeta_i = A_i(\varepsilon^o(k))B_i(\Delta\varepsilon^o(k))$. In the quantum neural self-tuning controller, the scale factors in the antecedent part, K_ε and $K_{\Delta\varepsilon}$, are tuned by the output from the quantum neural network. Because the fuzzy logic controller can be considered as a non-linear PI controller by considering an analogy to the PID controller, the input vector of the quantum neural network $\boldsymbol{x}(k)$ can be defined as follows:

$$x(k) = [\ y_d(k+d) \quad y(k) \quad \cdots \quad y(k-n+1) \quad u(k-1) \quad \cdots \quad u(k-m-d+1) \quad \delta_2\delta_3\Delta\varepsilon(k) \quad \delta_2\Delta\varepsilon(k) \quad \delta_1\varepsilon(k) \quad \det(e(k)e^{\mathrm{T}}(k)+D)\ ].$$

Figure 5 illustrates examples of the plant responses, and Fig. 6 shows the normalized cost function during the adaptation process of the quantum neural self-tuning controller. Here the plant was Eq. (12), and the quantum neural network was a 7–14–2 network topology in which the parameters were the same as the case in which the conventional controller was the PID controller, with the exception of $\eta = 10^{-5}$. Because the Jacobian of the fuzzy logic controller is hard to calculate analytically, it was approximated by $\partial u(k)/\partial u_n(k) \approx \left[\ \frac{\Delta u(k)-\Delta u(k-1)}{K_\varepsilon(k)-K_\varepsilon(k-1)} \quad \frac{\Delta u(k)-\Delta u(k-1)}{K_{\Delta\varepsilon}(k)-K_{\Delta\varepsilon}(k-1)}\ \right]$. The scale factor in the consequent part $K_{\Delta u}$ was 0.1. As shown in Fig. 6, the normalized cost function decreased as the training progressed and the quantum neural self-tuning controller can make the plant output follow the desired plant output by tuning the scale factors in the fuzzy logic controller, as shown in Fig. 5. This result indicates the usefulness of applying the quantum neural self-tuning controller to many types of conventional controller.

4 CONCLUSIONS

This paper presented an adaptive-type self-tuning controller based on a multi-layer quantum neural network which uses qubit neurons as an information processing unit, and investigated its characteristics for control systems. We proposed a design of the quantum-neural-network-based self-tuning controller which conducts the training of the quantum neural network as an online process. For a conventional controller in which parameters are tuned by the quantum neural network, a digital PID control law was utilized. Computational experiments for the control of a single-input single-output non-linear discrete-time plant were conducted to evaluate the learning performance and capability of the adaptive-type quantum neural self-tuning controller. To investigate the use-

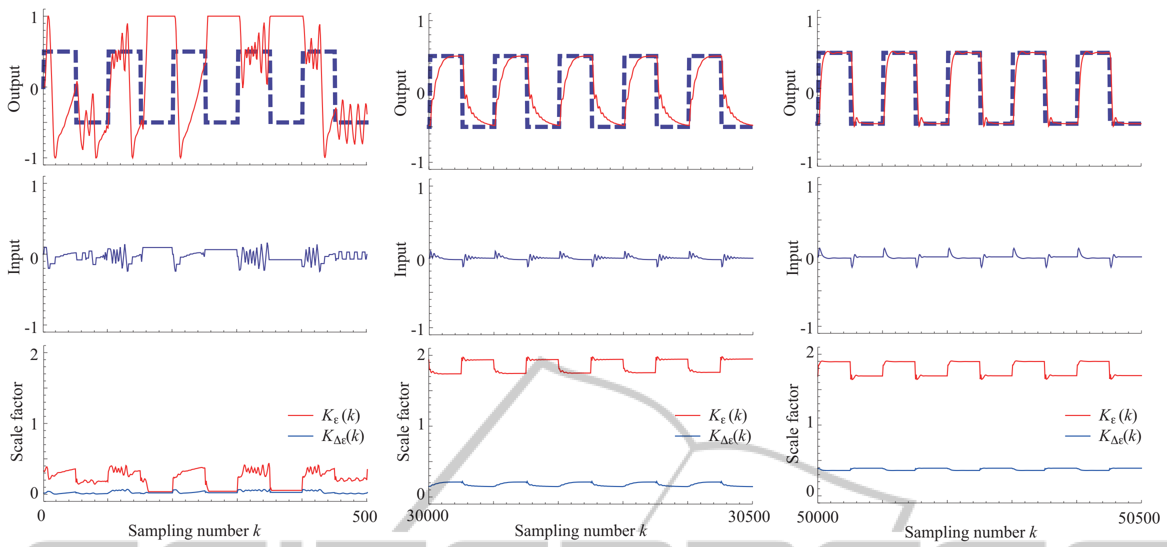

Figure 5: Example of the adaptation process controlled by the quantum neural self-tuning controller (top: plant output where the dotted line indicates the desired plant output $y_d(k)$ and the rigid line indicates the plant output $y(k)$; middle: control input from the fuzzy logic controller; bottom: output from the quantum neural network where the red line indicates $K_\varepsilon(k)$ and the blue line indicates $K_{\Delta\varepsilon}(k)$).

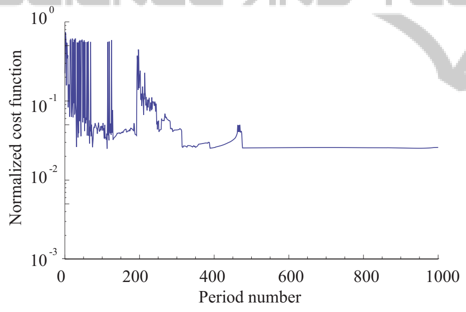

Figure 6: Relationship between the period number of the desired plant output and the normalized cost function (conventional controller: fuzzy logic controller).

fulness of the quantum neural self-tuning controller, the controller was also applied to tune the parameters of a fuzzy logic controller. The results of the computational experiments confirmed both the feasibility and effectiveness of the adaptive-type quantum neural self-tuning controller.

REFERENCES

Ezhov, A. A. and Ventura, D. (2000). Quantum neural networks. In *Future Directions for Intelligent Systems and Information Sciences*, pages 213–234. Physica-Verlang.

Hagan, M. T., Demuth, H. B., and Jesus, O. D. (2002). An introduction to the use of neural networks in control systems. *International Journal of Robust and Nonlinear Control*, 12(11):959–985.

Kouda, N., Matsui, N., Nishimura, H., and Peper, F. (2005). Qubit neural network and its learning efficiency. *Neural Computing and Application*, 14(2):114–121.

Manju, A. and Nigam, M. J. (2012). Applications of quantum inspired computational intelligence: A survey. In *Artificial Intelligence Review*, pages 1–78. Springer Netherlands.

Meireles, M. R. G., Almeida, P. E. M., and Simoes, M. G. (2003). A comprehensive review for industrial applicability of artificial neural networks. *IEEE Transactions on Industrial Electronics*, 50(3):585–601.

Sommer, G. (2001). *Geometric Computing with Clifford Algebra*. Springer.

Takahashi, K., Kurokawa, M., and Hashimoto, M. (2011). Controller application of a multi-layer quantum neural network trained by a conjugate gradient algorithm. In *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 2278–2283.

Yamada, T. (2011). Discussion of neural network controllers from the point of view of inverse dynamics and folding behaviour. In *Proceedings of SICE Annual Conference 2011*, pages 2210–2215.

Zhou, R., Qin, L., and Jiang, N. (2006). Quantum perceptron network. In *Proceedings of the 16th International Conference on Artificial Neural Networks*, volume 1, pages 651–657.