REMARKS ON QUBIT NEURON-BASED QUANTUM NEURAL SERVO CONTROLLER

Kazuhiko Takahashi

Doshisha University, 1-3 Miyakodani Tatara Kyotanabe Kyoto, Japan
katakaha@mail.doshisha.ac.jp

ABSTRACT

This paper presents a servo-level controller using a quantum neural network and investigates its characteristics for control systems. A multi-layer quantum neural network that uses qubit neurons as an information processing unit is used to design three types of neural-network-based servo controllers: a direct controller, parallel controller and self-tuning controller. Computational experiments to control a nonlinear discrete time plant are conducted in order to evaluate the learning performance and capability of the quantum neural-network-based servo controllers. The results of the computational experiments confirm both the feasibility and effectiveness of the quantum neural-network-based servo controllers.

Keywords: Quantum neural network, Qubit neuron, Servo controller, Real-coded genetic algorithm

1. INTRODUCTION

After Feynman (1982) introduced the possibility of using quantum mechanical systems for reasonable computing, Deutsch (1989) proposed the first quantum computing model. Subsequently, several quantum computing algorithms (Shor 1994, Grover 1996) have been proposed. Since Kak (1995) originally presented the concept of quantum neural computing, interest in artificial neural networks based on quantum theoretical concepts and techniques (hereafter called quantum neural networks) increased because of the belief that quantum neural networks may provide a new understanding of certain brain functions and also help solving classically intractable problems (Ezhov and Ventura 2000). In quantum computing, 'qubits' (an abbreviation for quantum bits) are the counterparts of the 'bits' of classical computers, and they are used to store the states of circuits in quantum computations. The quantum neural network that utilizes qubit neurons as an information processing unit was proposed by Matsui, Takai, and Nishimura (2000), where the qubit neuron model is the one in which neuron states are connected to quantum states, and the transitions between neuron states are based on operations derived from quantum logic gates. The high learning capability of the quantum neural network with qubit neurons was demonstrated in several basic benchmark tests and applications (Mitrpanont and Srisuphab 2002; Kouda, Matsui, Nishimura, and Peper 2005; Araujo, Oliveira, and Soares 2010; Takahashi, Kurokawa, and Hashimoto 2011). During the past quarter of the century, many studies about the application of both flexibility and learning capability of neural networks to control systems have been conducted worldwide (Narendra and Parthictsarathy 1990). Although many types of neural network-based control systems have been proposed (Hagan, Demuth, and De Jesus 2002, Balderud and Giovanini 2008), the possibility of applying quantum neural networks to servo-level controller applications has not been adequately investigated.

This paper proposes a servo controller design using a quantum neural network and investigates its characteristics for control systems. In Section 2, a multi-layer quantum neural network with qubit neurons is described and designs of three types of quantum neural-network-based servo controllers are presented (hereafter called

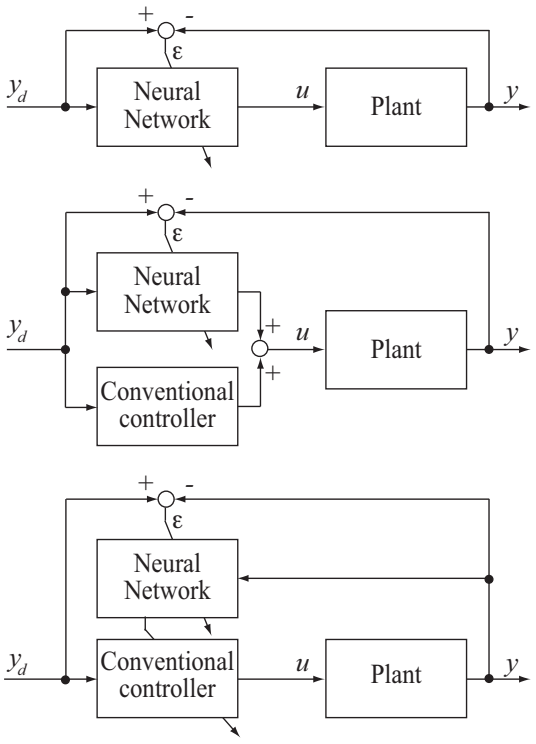

Figure 1: Schematic figure of a neural-network-based servo controller, where y_d is the desired output, y is the plant output, u is the plant input and ϵ is the output error between the desired output and the plant output (top: direct controller; middle: parallel controller; bottom: self-tuning controller.)

quantum neural servo controllers.) In this study, a direct controller, parallel controller and self-tuning controller, as shown in Fig. 1 are presented. In Section 3, computational experiments for controlling a discrete time nonlinear plant are conducted to evaluate the feasibility of the quantum neural network for servo-level controller applications.

2. DESIGN OF A QUANTUM NEURAL SERVO CONTROLLER

2.1 Neural Servo Controller

In this section, the basic idea of designing neural servo controllers is described. To design neural servo controllers, the following single-input single-output discrete time plant is considered as a controlling target plant:

$$y(k+d) = F_p[y(k), \cdots, y(k-n+1), u(k), \cdots, u(k-m-d+1)], \quad (1)$$

where y is the plant output, u is the plant input, n and m are the plant orders, k is the sampling number, d is the dead time of the plant and $F_p(\cdot)$ is the function that expresses plant dynamics. This design uses the following assumptions: the upper limit orders of the plant are known and the dead time of the plant are known. The plant output $y(k)$ depends on the past plant input and output. The plant orders determine the period in which the plant output depends on them. This period is usually shorter than the trial period. By considering the desired plant output $y_d(k)$, the output error $\epsilon(k)$ can be defined: $\epsilon(k) = y_d(k) - y(k)$.

Direct controller: In the direct controller, the output from the neural network u_n is input to the plant directly as shown at the top of Fig. 1: $u(k) = u_n(k)$. Substituting Eq. (1) into the output error and then setting it to zero, an input vector $\boldsymbol{x}_D(k)$ of the neural network can be defined as follows:

$$\boldsymbol{x}_D(k) = [\ y_d(k+d) \quad y(k) \quad \cdots \quad y(k-n+1) \quad u(k-1) \quad \cdots \quad u(k-m-d+1)\ ]^{\mathrm{T}}. \quad (2)$$

Parallel controller: In the parallel controller, the plant input is composed of the outputs obtained from the neural network u_n and a conventional controller u_c as shown in the middle of Fig. 1: $u(k) = u_n(k) + u_c(k)$. Substituting Eq. (1) into the output error and then setting it to zero, an input vector $\boldsymbol{x}_P(k)$ of the neural network can be defined as follows:

$$\boldsymbol{x}_P(k) = [\ y_d(k+d) \quad y(k) \quad \cdots \quad y(k-n+1) \quad u_n(k-1) \quad \cdots \quad u_n(k-m-d+1) \quad u_c(k) \quad \cdots \quad u_c(k-m-d+1)\ ]^{\mathrm{T}}. \quad (3)$$

Self-tuning controller: In the self-tuning controller, the plant input is calculated by a conventional feedback and/or feedforward controller and its parameters are adjusted by the neural network as shown at the bottom of Fig. 1. When a digital PID control law is utilized as the conventional controller, the plant input is defined as follows:

$$\begin{aligned} u(k) = u(k-1) &+ K_P(k)\{\epsilon(k) - \epsilon(k-1)\} \\ &+ K_I(k)\epsilon(k) \\ &+ K_D(k)\{y(k) - 2y(k-1) + y(k-2)\}, \end{aligned} \quad (4)$$

where K_P(k), $K_I(k)$ and $K_D(k)$ are the proportional gain, integral gain and differential gain, respectively. In this controller, the gain parameters are given by the neural network as follows: $K_P(k) = u_{n_1}(k)$, $K_I(k) = u_{n_2}(k)$ and $K_D(k) = u_{n_3}(k)$. Substituting Eq. (1) into the output error and then setting it to zero, an input vector $\boldsymbol{x}_S(k)$ of the neural network can be defined as follows:

$$\boldsymbol{x}_S(k) = [\ y_d(k+d) \quad y(k) \quad y(k-1) \quad y(k-2) \quad \cdots \quad y(k-n+1) \quad u(k-1) \quad \cdots \quad u(k-m-d+1) \quad \epsilon(k) \quad \epsilon(k-1)\ ]^{\mathrm{T}}. \quad (5)$$

2.2 Multi-layer Quantum Neural Network with Qubit Neurons

In quantum computing, the two quantum states express one bit of information: $|0\rangle \in \mathbf{C}$ corresponds to the classical computer's bit 0 and $|1\rangle \in \mathbf{C}$ corresponds to bit 1. Here the symbol $|\ \rangle$ is a part of the Dirac notation. The qubit state $|\psi\rangle$ maintains a coherent superposition of states: $|\psi\rangle = a\,|0\rangle + b\,|1\rangle$, where a and b are complex numbers called probability amplitudes that satisfy $|a|^2 + |b|^2 = 1$. The operations of the rotation gate and controlled NOT gate in quantum computations can be expressed by the quantum state rewritten using the phase ϕ. The rotation gate, which is a phase-shift gate that transforms the phase of quantum states, can be expressed as $f(\phi_1 + \phi_2) = f(\phi_1)f(\phi_2)$, where $f(\phi) = \mathrm{e}^{\mathrm{i}\phi}$ (i is an imaginary unit). The controlled NOT gate, which is the phase reverse operation defined with respect to the controlled input parameter γ, can be given by $f(\frac{\pi}{2}\gamma - \phi)$, where $\gamma = 1$ and $\gamma = 0$ correspond to reversal rotation and non-rotation, respectively. Although the phase of the probability amplitude of the quantum state $|1\rangle$ is reversed when $\gamma = 0$, this case can be treated as non-rotation because its observed probability is invariant. By considering these gates, the state of the j-th qubit neuron model in the r-th set z_j^r is defined as follows:

$$\begin{cases} z_j^r = f[\dfrac{\pi}{2}\delta_j^r - \arg(v_j^r)] \\ v_j^r = \displaystyle\sum_l f(\theta_{l,j}^r) f(z_l^{r-1}) - f(\lambda_j^r) \end{cases}. \quad (6)$$

Here, δ_j^r is the reversal parameter corresponding to the controlled NOT gate, $\theta_{l,j}^r$ is the phase parameter corresponding to the phase of the rotation gate and λ_j^r is the threshold parameter having a range $[0, 1]$.

The multi-layer quantum neural network is designed by combining qubit neurons in layers. In the input layer (indicated by superscript r of I), the network input x_l in the range $[0, 1]$ is first converted into quantum states with a phase in the range $[0, \frac{\pi}{2}]$, and then the output, given

by $z_l^I = f(\frac{\pi}{2}x_l)$, is fed into the neurons present in the hidden layer (indicated by superscript r of H). In the hidden and output layers, the outputs from neurons are given by Eq. (6). By considering the probability of the state in which $|1\rangle$ is observed from the j-th neuron in the output layer (indicated by superscript r of O), the output from the network u_{QN_j} is defined as follows:

$$u_{QN_j}(\omega, \boldsymbol{x}) = |\mathrm{Im}(z_j^O)|^2. \tag{7}$$

Here, the vector ω is composed of parameters $\theta_{l,j}^r$, δ_j^r and λ_j^r ($r = I, H, O$), and the vector $\boldsymbol{x}$ is composed of input x_l. In practical applications, the outputs from the quantum neural network are converted from the range $[0, 1]$ into the range $[u_{n_{min}}, u_{n_{max}}]$ with a gain and shift factors: $u_{n_j} = c_0(u_{QN_j} - u_{QN_0})$ where c_0 is the gain factor, and u_{QN_0} is the shift factor.

2.3 Training by Real-coded Genetic Algorithm

The training of the quantum neural servo controller is performed by searching the optimal parameters $\theta_{l,j}^r$, δ_j^r and λ_j^r of the quantum neural network so as to minimize the cost function, $J(\omega) = \frac{1}{2}\sum_k \epsilon^2(k)$. A back-propagation algorithm, which is based on the steepest descent method to minimize the cost function: $\omega_{p+1} = \omega_p - \eta\frac{\partial J(\omega_p)}{\partial \omega_p}$ where η is the learning factor and p is the iteration number, can be applied for training the quantum neural servo controller. However, its learning occasionally falls into a local minimum, and the information of a plant Jacobian $\frac{\partial y}{\partial u}$ is required in order to calculate the derivative of the cost function. In this study, a real-coded genetic algorithm (Akimoto, Sakuma, Ono, and Kobayashi 2009) is utilized in training the quantum neural servo controller. Thus, the parameter values of the vector $\boldsymbol{w}$ are used directly as gene parameters of an individual. The real-coded genetic algorithm is composed of a multi-parental crossover and a generation alternation model. A real-coded ensemble crossover is used as the multi-parental crossover. The real-coded ensemble crossover is a generalization of the enhanced unimodal normal distribution crossover, and has some probability distribution in the multi-parental crossover operation in order to avoid the asymmetry and bias of children distribution. In the real-coded ensemble crossover, the new individuals (children), $\boldsymbol{g}_c$, are generated using multi-parental individuals, $\boldsymbol{g}_j$ ($j = 1, 2, \cdots, N + K$: N is the dimension of the problem; in this study it is the dimension of the vector $\boldsymbol{w}$):$\boldsymbol{g}_c = \boldsymbol{g}_0 + \sum_{j=1}^{N+K}\nu_j(\boldsymbol{g}_j - \boldsymbol{g}_0)$, where $\boldsymbol{g}_0$ indicates the center of gravity of the parents and ν_j is the stochastic variable that follows the probability distribution $\varphi(0, \frac{1}{N+K})$. In a generation alternation model, the just generation gap which replaces parents with children completely in every generation, is utilized. In the just generation gap, the numbers of population, parents and children are recommended to be $(15 \sim 50)N$, $N + K$ and $10N$, respectively. To evaluate the individuals, a fitness function of the q-th individual at the p-th generation is defined by the reciprocal of the cost function $J(\omega_p^q)$.

3. COMPUTATIONAL EXPERIMENTS

The quantum neural servo controllers are numerically investigated using a discrete time nonlinear plant in the computational experiments. The equation of the plant in which the second-order system is dominant is as follows:

$$y(k+1) = F_s[-\sum_{i=1}^{3} a_i y(k-i+1) + \sum_{i=1}^{2} b_i u(k-i+1) + c_{non}y^2(k)], \tag{8}$$

where a_3 and c_{non} are the coefficients of the parasitic term and the nonlinear term and the function $F_s(\cdot)$ has the nonlinear characteristic of saturation:

$$F_s(x) = \begin{cases} 1 & (x \geq 1) \\ x & (-1 < x < 1) \\ -1 & (x \leq -1) \end{cases}$$

In the experiments, the values of 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 2010).

In designing the quantum neural servo controller, the plant was assumed to be a linear second order plant: $d = 1$, $n = 2$ and $m = 1$. Thus, the number of qubit neurons in the input layer was 4, 6 and 7 for the direct controller, parallel controller and self-tuning controller, respectively. In each controller, the number of qubit neurons in the hidden layer was set to 4. In the parallel controller, P-control law was utilized as a conventional controller: $u_c = K_P\epsilon(k)$, where $K_P = 0.5$ in the computational experiment. The training conditions were as follows: the dimension of the problem N were 30, 38 and 54 for the direct controller, parallel controller and self-tuning controller, respectively, The value of K was 1, total number of generations was 2000 and probability distribution φ was uniform in the range [-1, 1]. In the training process, the desired plant output $y_d(k)$ was a rectangular wave in order to take account of frequency richness. The number of samples within one cycle of the rectangular wave was 50 and the amplitude of the wave was changed randomly in the range [-0.5, 0.5]. In the open test, the desired plant output, the rectangular wave, had an amplitude of ± 0.4 and the number of samples within one cycle varied from 50 to 60, 30 and 40 in order.

As a reference for comparing the results of the quantum neural servo controller, the following conventional multi-layer neural network that uses sigmoid functions as an information processing unit was utilized to design the neural servo controller (hereafter called sigmoid neural servo controller).

$$u_{MN_j}(\boldsymbol{w}, \boldsymbol{x}) = s[\sum_i w_{2_{ji}} s(\sum_l w_{1_{il}} x_l)], \tag{9}$$

where x_l is the input to the lth neuron in the input layer, u_{MN_j} is the output of the jth neuron in the output layer, $w_{i_{jl}}$ ($i = 1, 2$) is the weight that includes the threshold, $s(\cdot)$ is the sigmoid function, and the vectors $\boldsymbol{w}$ and

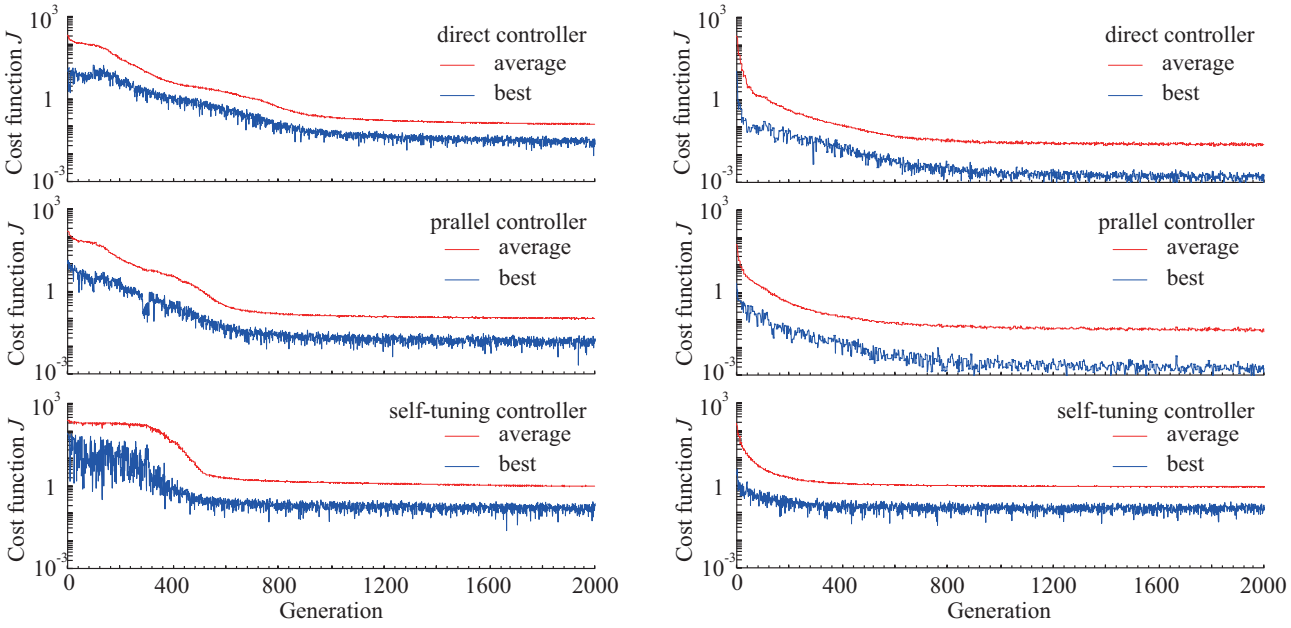

Figure 2: Example of neural servo controllers' training process (left: quantum neural servo controller; right: sigmoid neural servo controller; top: direct controller; middle: parallel controller; bottom: self-tuning controller.)

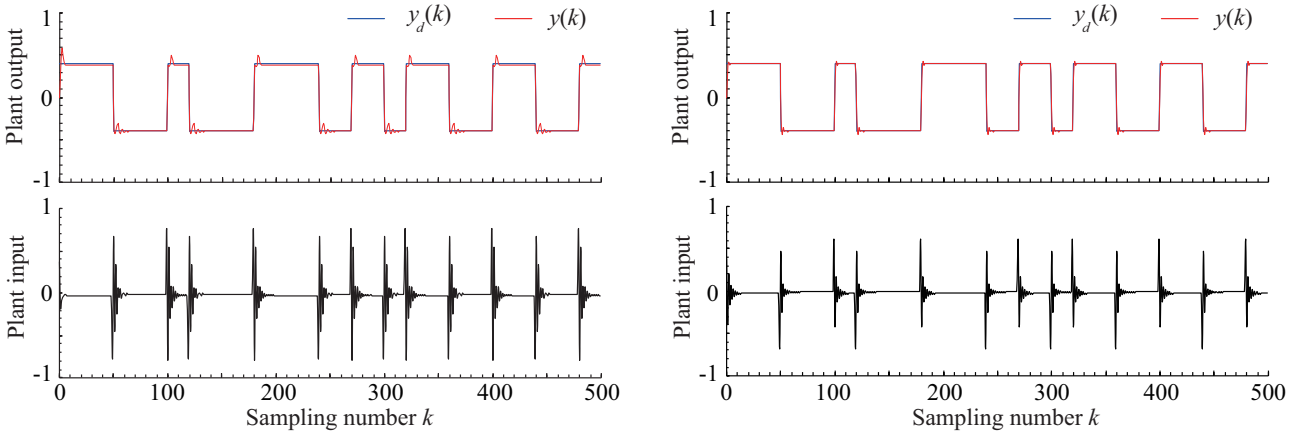

Figure 3: System response controlled by the direct neural servo controller (left: quantum neural servo controller [J = 0.224]; right: sigmoid neural servo controller [J = 0.243]; top: desired and plant outputs; bottom: plant input.)

$\boldsymbol{x}$ are composed of the weight $w_{i_{jl}}$ $(i = 1, 2)$ and input x_l respectively. The learning of the sigmoid neural servo controller is performed by the real-coded genetic algorithm in order to minimize the cost function, $J(\boldsymbol{w}) = \frac{1}{2}\sum_k \epsilon^2(k)$. The training conditions used here were the same as that in the quantum neural servo controller. The number of neurons in the hidden layer was set to 6 in each controller so that the dimension of the problem in the sigmoid neural controller was almost the same as that in the quantum neural controller. Thus N was 31, 41 and 58.

Figure 2 shows an example of the training process. Here, the horizontal axis represents the generation and the vertical axis represents the cost function. As shown in Fig. 2, the cost function decreased as the generation progressed. Although the averaged cost functions are almost same in the quantum neural servo controller and the sigmoid neural servo controller, the cost functions of the sigmoid neural servo controller with the best individual are smaller than those of the quantum neural servo controllers in the direct and parallel controllers.

Figures 3, 4 and 5 are examples of responses controlled by the direct controller, parallel controller and self-tuning controller, respectively, after the neural servo controller's training converged. In each controller, the individual that has the best fitness function was used. Because the plant output tracks the desired output, as shown in each figure, the quantum neural servo controllers can achieve the control task of making the nonlinear plant follow the desired output. By comparing the three types

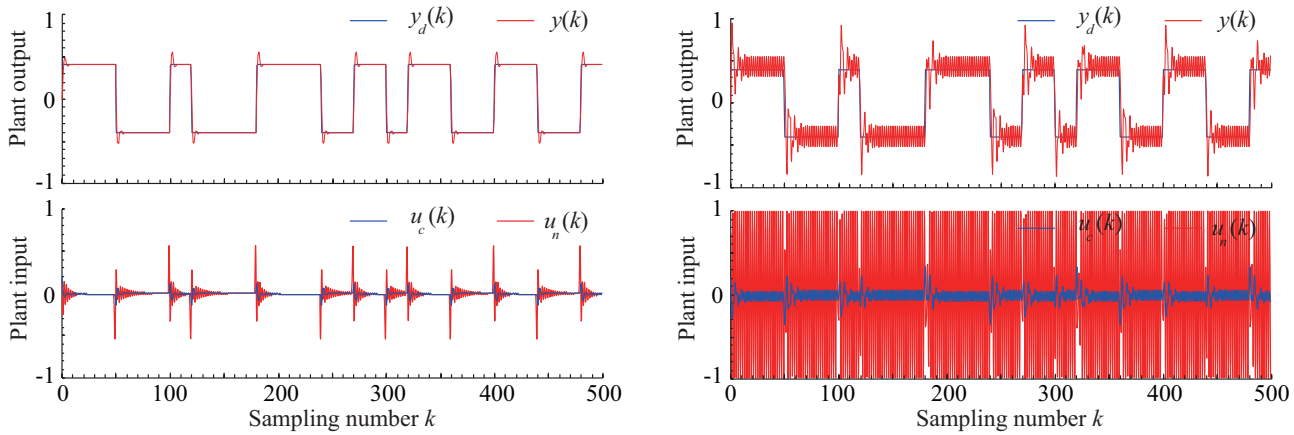

Figure 4: System response controlled by the parallel neural servo controller (left: quantum neural servo controller [J = 0.695]; right: sigmoid neural servo controller [J = 7.631]; top: desired and plant outputs; bottom: plant input.)

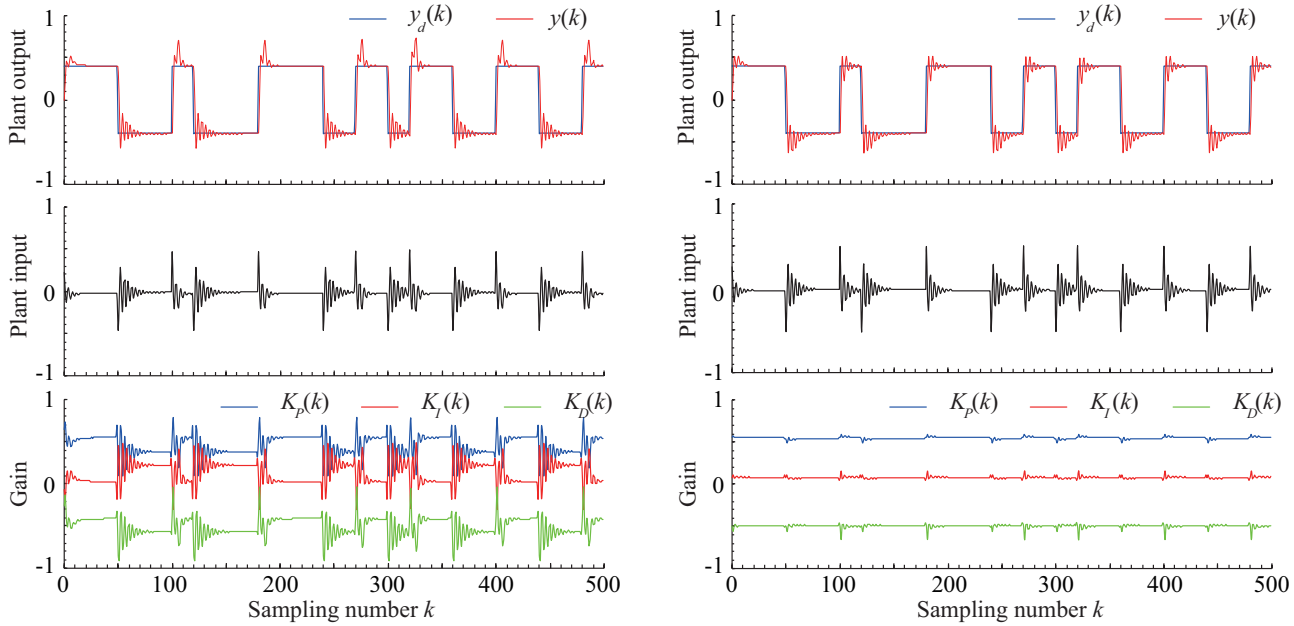

Figure 5: System response controlled by the self-tuning neural servo controller (left: quantum neural servo controller [J = 5.779]; right: sigmoid neural servo controller [J = 5.336]; top: desired and plant outputs; middle: control input; bottom: PID gain parameters.)

of quantum neural servo controllers, we observe that the direct controller has the lowest cost function in the open test. In the direct controller, the control performance of the quantum neural servo controller is almost the same as that of the sigmoid neural servo controller. In the parallel controller, shown in Fig. 4, the output from the quantum neural network is dominant in the plant input (the output from the conventional controller is almost zero). In the self-tuning controller, although the gain parameters are tuned by the output from the quantum neural network, as shown in the bottom of Fig. 5, the output error increases rather than the direct and parallel controllers because the control performance depends on the PID law. These results indicate the feasibility of the quantum neural servo controllers and show that the real-coded genetic algorithm has an advantage in the learning of the quantum neural servo controller because it does not require the Jacobian information of the plant in the training process.

4. CONCLUSION

This paper presented a servo controller designs using a quantum neural network and investigated their characteristics for control systems. The multi-layer quantum neural network that uses qubit neurons as an information processing unit was used to design three types of neural-network-based servo controller: the direct controller, parallel controller and self-tuning controller. Computational

experiments to control the nonlinear discrete time plant were conducted in order to evaluate the learning performance and capability of the quantum neural servo controllers. The results of the computational experiments confirmed both the feasibility and effectiveness of the quantum neural servo controllers.

REFERENCES

Akimoto, Y., Sakuma, J., Ono, I. and Kobayashi, S., 2009. Adaptation of Expansion Rate for Real-coded Crossovers. *Proceedings of 11th Annual Conference on Genetic and Evolutionary Computation*, 739–746.

Araujo, R. A., Oliveira, A. L. I. and Soares, S. C. B., 2010. A Quantum-Inspired Hybrid Methodology for Financial Time Series Prediction. *Proceedings of the 2010 International Joint Conference on Neural Networks*, 1–8.

Balderud, J. and Giovanini, L., 2008. Adaptive Control and Signal Processing Literature Survey. *International Journal of Adaptive Control and Signal Processing*, 22(3), 318–321.

Deutsch, D., 1989. Quantum Computational Networks. *Proceedings of the Royal Society of London*, Series A, 425, 73–90.

Ezhov, A. A. and Ventura, D., 2000. Quantum Neural Networks. *Future Directions for Intelligent Systems and Information Sciences*, 213–234.

Feynman, R., 1982. Simulating Physics with Computers. *International Journal of Theoretical Physics*, 21, 467–488.

Grover, L. K., 1996. A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, 212–219.

Hagan, M. T., Demuth, H. B. and De Jesus, O., 2002. An Introduction to the Use of Neural Networks in Control Systems. *International Journal of Robust and Nonlinear Control*, 12(11), 959–985.

Kak, S. C., 1995. On Quantum Neural Computing. *Information Science*, 83, 143–163.

Kouda, N., Matsui, F., Nishimura, H. and Peper, F., 2005. Qubit Neural Network and Its Learning Efficiency. *Neural Computing and Application*, 14(2), 114–121.

Matsui, N., Takai, M. and Nishimura, H., 2000. A Network Model Based on Qubit-like Neuron Corresponding to Quantum Circuit. *Electronics and Communications in Japan*, 83(10), 67–73.

Mitrpanont, J. L. and Srisuphab, A., 2002. The Realization of Quantum Complex-Valued Backpropagation Neural Network in Pattern Recognition Problem. *Proceedings of the 9th International Conference on Neural Information Processing*, 1, 462–466.

Narendra, K.S. and Parthictsarathy, K., 1990. Identification and Control of Dynamics System Using Neural Networks. *IEEE Transactions on Neural Networks*, 1(1), 4–27.

Shor, P. W., 1994. Algorithms for Quantum Computation: Discrete log and factoring. *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*, 124–134.

Takahashi, K., Kurokawa, M. and Hashimoto, M., 2011. Controller Application of a Multi-Layer Quantum Neural Network Trained by a Conjugate Gradient Algorithm. *Proceedings of the 37th Annual Conference of the IEEE Industrial Electronics Society*, 2278–2283.

Yamada, T., 2010. Transformation of Neural Network Weight Trajectories on 2D Plane for A Learning-type Neural Network Direct Controller. *Artificial Life and Robotics*, 15, 413–416.