Implementation of a Neural Network Controller on a DSP for Controlling an Inverted Pendulum System on an X-Y Plane

Sung S. Kim*, Geun H. Lee**, and Seul Jung***

*Agency for Defense Development, Daejeon, Korea, ** Mechatronics Engineering Department, Chungnam National University, Daejeon, Korea (Tel: 82-42-821-7232; e-mail: sadthink@paran.com).
*** Mechatronics Engineering Department, Chungnam National University, Daejeon, Korea, (Tel: 82-42-821-6876; e-mail: jungs@cnu.ac.kr)

Abstract: This paper presents the hardware implementation of a neural network controller for controlling an inverted pendulum system on an x-y plane robot. The inverted pendulum system can move on an x-y plane while balancing the angle of the pendulum. Neural network algorithm is implemented on a cost effective DSP board in association with an FPGA chip. The reference compensation technique of neural network control scheme is used for on-line learning and control of the inverted pendulum system. Experimental results of tracking the circular trajectory while balancing the pendulum demonstrate to confirm the successful performance of the neural network hardware.

1. INTRODUCTION

Recently, interests in the intelligent system area have enormously been increased. Researches on intelligent systems become a very important subject in a variety of engineering fields such as control systems, robot systems, and signal processing systems. One of the next frontier research topics seems to develop intelligent systems that apply the intelligence to the systems to solve very complicated problems. Although the research history on intelligent systems has not been long compared with that of other control subjects such as adaptive control, optimal control, and robust control, many progresses have been made by enormous attraction by researchers.

After 1980s, the era of intelligent control begins and many successful results have been presented. Neural network and fuzzy logics are two major tools for developing intelligent control systems. Fuzzy logics are more practical and can be relatively implemented as hardware since real time calculation can be easily satisfied although it suffers from the difficulty of determining rules.

Starting from developing the theoretical analysis of how to use neural network in control systems as an online learning and control tool, simulation results have been presented. In recent days, however, an actual hardware implementation of neural network controllers becomes more feasible and it has been implemented to confirm the theoretical analysis. Successful demonstrations of achieving real-time control applications have been presented.

As a test-bed system for confirming neural network control schemes, the inverted pendulum control system has been considered as a prototype example of nonlinear system control applications whose structure is a single-input multiple-output(SIMO) where one single input force has to control both the angle of the pendulum and the position of the cart at the same time (Hung et al., 1997, Mogana et al., 1998). Recently, numerous examples of more challenging inverted pendulum systems as extensions of the one dimensional pendulum system have been presented. Control of two-degrees-of-freedom inverted pendulum or a spherical pendulum moving on the x-y plane has been proposed and successfully demonstrated (Kim et al., 2004). Further extensions of simple pendulum models include an acrobat(Spong et al., 1995), and the Furuta pendulum. Increasing interest in inverted pendulum systems extends the category to a more interesting and challenging 3D inverted pendulum problem(Shen et al., 2004).

In this paper, the hardware implementation of a neural network controller for controlling an inverted pendulum system on an x-y plane robot is presented. The inverted pendulum system has two degrees-of-freedom to move on an x-y plane while balancing the angle of the pendulum. In our previous researches, control of the inverted pendulum system on x-y plane has been successfully demonstrated with expensive DSP systems (Jung et al., 2004). Here, we develop cost effective neural network control hardware. Neural network learning and control algorithm is implemented on a cost effective DSP board and the PID control algorithm is implemented on an FPGA chip to form the reference compensation technique scheme of neural network. Experimental results of commanding to track the circular trajectory while balancing the pendulum show successful demonstration to confirm the performance of the neural network hardware.

2. INVERTED PENDULUM SYSTEM ON AN X-Y PLANE
The inverted pendulum system is shown in Fig. 1. The pendulum can move on the x-y plane controlled by an x-y table robot. It is a more challenging task compared to one axis inverted pendulum system where coupling effects between two axes do not exist. The goal is to maintain balancing the pendulum while tracking the desired trajectory \( x, y \) by regulating two control inputs, \( u_x, u_y \).

The concept of the RCT is to compensate the system controlled by predetermined controllers by closing another outer loop. The neural network output signal \( \phi_j \) compensate at the desired trajectory \( q_d \) to modify the control input \( u \) by minimizing the output error, \( e = q_d - q \).

The detailed control structure for the pendulum system is shown in Fig. 2. Neural network outputs are added to tracking errors to form PID controller outputs. A control input \( u_{sd} \) for the pendulum angle and a control input \( u_{sp} \) for the cart position are summed together to generate an input force \( u_x \) to the system.

The pendulum angle error of the x axis is defined as

\[
e_{x\theta} = \theta_{sd} - \theta_x,
\]

where \( \theta_{sd} \) is the desired angle of the pendulum and \( \theta_x \) is the actual angle of the pendulum.

Then a PID control input for the angle control is given by

\[
u_x = k_{sp\theta}e_{x\theta}(t) + k_{sd\theta}\int e_{x\theta}(t)dt + k_{sd\theta}\dot{e}_{x\theta}(t) + k_{sp\theta}\phi_{x1} + k_{sd\theta}\phi_{x2} + k_{sd\theta}\phi_{x3},
\]

where \( k_{sp\theta}, k_{sd\theta}, k_{sd\theta} \) are PID gains for the pendulum control and \( \phi_{x1}, \phi_{x2}, \phi_{x3} \) are neural network outputs.

The mobile pendulum position error is defined by

\[
e_{xp} = x_d - x,
\]

where \( x_d \) is the desired cart position and \( x \) is the actual position of the cart.

The PID control input for the position control is

\[
u_{sp} = k_{px}e_{x}(t) + k_{dx}\int e_{x}(t)dt + k_{dx}\dot{e}_{x}(t) + k_{px}\phi_{x4} + k_{dx}\phi_{x5} + k_{px}\phi_{x6},
\]

where \( k_{px}, k_{dx}, k_{px} \) are PID gains for the cart control and \( \phi_{x4}, \phi_{x5}, \phi_{x6} \) are neural network outputs.

The overall control input is the sum of two PID controller outputs, \( u_{sd} \) in (2) and \( u_{sp} \) in (4)

\[
u_x = u_{sp} + u_{sd}.
\]

In the same manners, the control input for the y axis is given by
where
\[ u_y = u_{yp} + u_{y0}. \] (6)

In the same way, we have the control input for the cart position
\[ u_p = k_{yp} e_p(t) + k_{yp} \int e_p(t)dt + k_{yp} \phi_p + \Phi_p, \] (9)
where \( \Phi_p = k_{\phi_p} \phi_x + k_{\phi_p} \phi_y + k_{\phi_p} \phi_z \).

If the system dynamic equation is represented as \( f(\theta, \dot{\theta}, \phi, p, \dot{p}) \), then combining the system dynamic equation with (8) and (9) yields
\[ K_\theta \dot{\theta} + K_\phi \dot{\phi} = f(\theta, \dot{\theta}, \phi, p, \dot{p}) - \Phi, \] (10)
where \( \Phi = \Phi_\theta + \Phi_p \), \( K_\phi = [k_{\phi\theta}, k_{\phi\phi}], K_{\phi\phi} = [k_{\phi\phi}, k_{\phi\phi}] \), and \( e = [e_\theta, e_\phi]^T \).

To learn the inverse dynamic of the system, we set the training signal as
\[ v = K_\phi e + K_{\phi\phi} \int edt + K_{\phi\phi} \phi. \] (11)

When the error converges, that is, when the training signal \( v \) converges to zero, the neural network output becomes \( \Phi \cong f(\theta, \dot{\theta}, \phi, p, \dot{p}) \) so the inverse dynamic control can be accomplished.

Next is to develop on-line learning algorithm, the back-propagation algorithm for the neural controller. Define the objective function to be minimized as
\[ E = \frac{1}{2} v^2. \] (12)

Differentiating (12) with the weight vector \( w \) yields
\[ \frac{\partial E}{\partial w} = \frac{\partial E}{\partial v} \frac{\partial v}{\partial w} = v \frac{\partial v}{\partial w} = -v \frac{\partial \Phi}{\partial w}, \] (13)
where
\[ \frac{\partial \Phi}{\partial w} = k_{\phi\theta} \frac{\partial \dot{\theta}}{\partial w} + k_{\phi\phi} \frac{\partial \dot{\phi}}{\partial w} + \frac{\partial \phi}{\partial w} + k_{\phi\phi} \frac{\partial \dot{\phi}}{\partial w} + k_{\phi\phi} \frac{\partial \phi}{\partial w} + \frac{\partial \phi}{\partial w} \] (14)

In details, for each output, the weight adjustment \( \Delta w_{jk} \) we have
\[ \Delta w_{jk} = \eta \delta_k O_j, \] (15)
where \( \eta \) is the learning rate, \( O_j \) is the \( j \)th output of the hidden layer, and \( \delta_k \) is
\[ \delta_k = -\frac{\partial E}{\partial S_k} = \frac{\partial E}{\partial v} \frac{\partial v}{\partial S_k} = -v \frac{\partial v}{\partial \phi_k} \frac{\partial \phi_k}{\partial S_k}. \] (16)
where $S_k$ is the $k$th summation of the output layer and $\phi_k$ is the $k$th output of the output layer. The gradient $\frac{\partial v}{\partial \phi_k}$ can be obtained from equation (13) as

$$
\frac{\partial v}{\partial \phi_1} = k_{\theta} \phi, \quad \frac{\partial v}{\partial \phi_2} = k_{\phi} \phi, \quad \frac{\partial v}{\partial \phi_3} = k_{\phi} \phi, \quad \frac{\partial v}{\partial \phi_4} = k_{\phi} \phi, \quad \frac{\partial v}{\partial \phi_5} = k_{\phi} \phi, \quad \frac{\partial v}{\partial \phi_6} = k_{\phi} \phi.
$$

(17)

The weights are updated as

$$
\Delta w(t) = \eta \frac{\partial \Phi}{\partial w} + \alpha \Delta w(t-1),
$$

(18)

$$
w(t + 1) = w(t) + \Delta w(t),
$$

(19)

where $\alpha$ is the momentum constant for helping the faster convergence of the error.

5. EXPERIMENTS

5.1 Experimental Setups

Fig. 4 shows the experimental setup for regulating the inverted pendulum system. Initially the pendulum angle is set to zero position by the encoder sensor.

Fig. 5 shows the corresponding control hardware for the pendulum which consists of a DSP controller and a motor driver. The motor drivers receive PWM signals from remotely located controllers through a serial communication and change it to currents. All calculations are done in the DSP controller shown in Fig. 5.

5.2 Circular Trajectory Tracking Task

The experiment is to test the desired trajectory tracking control of the pendulum system. The pendulum system is commanded to track desired circular trajectories while balancing the pendulum. The radius of the circle is 0.2m. Fig. 6 shows the tracking result. The pendulum tracks the trajectory well while maintaining the balance. The corresponding angle tracking error is within ±0.02 rad as shown in Fig. 7. We see a bit larger angle error in y axis since the y axis carries the x axis. The relative position tracking error is within 3 cm as shown in Fig. 8.
6. CONCLUSION

The successful hardware implementation of the neural network controller has been presented. A cost effective DSP controller has been designed to calculate the on-line learning algorithm of two neural networks. Although there exist coupling effects between two axes, neural network was able to decouple and compensate for uncertainties. The neural controller successfully regulates the position of the cart while balancing the pendulum which is quite difficult since the system is coupled and nonlinear.

ACKNOWLEDGEMENT

This research was financially in part supported by the Ministry of Commerce, Industry, and Energy (MOICE) and Korea Industrial Technology Foundation (KOTEF) through the Human Resource Training Project for Regional Innovation.

REFERENCES