ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 193-198

# A Sliding Mode Controller Using Neural Networks for Robot Manipulator

Hajoon Lee*, Dongkyung Nam, Cheol Hoon Park

Division of Electrical Engineering
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology
Daejeon, 305-701, Korea

**Abstract**. This paper proposes a new sliding mode controller using neural networks. Multilayer neural networks with the error back-propagation learning algorithm are used to compensate for the system uncertainty in order to reduce tracking errors and control torques. The stability of the proposed control scheme is proved with the Lyapunov function method. Computer simulation shows that the proposed neuro-controller yields better control performance than the conventional sliding mode controller in the view of tracking errors and overall control torque.

## 1 Introduction

In a robot system there are many uncertainties such as dynamic parameters (eg., inertia and payload conditions), dynamic effects (eg., complex nonlinear frictions), and unmodeled dynamics. Traditional linear controllers have many difficulties in treating these uncertainties. To overcome this problem, sliding mode control has been widely used as one of the precise and robust algorithms [1, 2]. The most distinguished property of the sliding mode control lies in its robustness. Loosely speaking, when a system is in a sliding mode, it is insensitive to dynamic uncertainties and external disturbances. One of the significant drawbacks of the sliding mode control is that it has a discontinuous switching function, which raises some theoretical as well as practical issues. A theoretical issue is the existence and the uniqueness of solutions and a practical one is the chattering phenomena due to imperfections in switching devices [3]. The chattering is generally undesirable because it involves extremely high control activities and may excite high-frequency dynamics neglected in modeling [4].

This work has been supported by EESRI(R-2003-B-411), which is funded by MOCIE(Ministry of commerce, industry and energy).

*E-mail: newth@nnmi.kaist.ac.kr

ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 193-198

By this reason, we use a continuous approximation of the discontinuous switching function and thus the so-called boundary layer approach. However, there is a trade-off between the steady-state error and boundary layer thickness [4].

In this paper, a sliding mode controller using neural networks is proposed to solve the boundary layer problem. The multilayer neural networks with error back-propagation learning algorithm [5] are used to learn the uncertainties in order to reduce the tracking errors and control torques. It adaptively generates additional input torques to reduce the tracking errors and control torques.

The following section describes the design method of the proposed sliding mode controller using neural networks. Simulation results for a two-link planar robot manipulator are shown in Section 3. Conclusion is given in Section 4.

## 2 Design of a sliding mode controller using neural networks

The dynamics of a rigid robot manipulator can be written as follows [6]:

$$\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{1}$$

where $\mathbf{q} \in \mathcal{R}^n$ is the joint angle vector, $H(\mathbf{q})$ is the $n \times n$ manipulator inertia matrix (which is symmetric positive definite), and $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \in \mathcal{R}^n$ represents centripetal and Coriolis torques. $\mathbf{g}(\mathbf{q}) \in \mathcal{R}^n$, $\boldsymbol{\tau} \in \mathcal{R}^n$ represent gravitational torques, applied joint torques, respectively. Let us define $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_d$ as a tracking error vector where $\mathbf{q}_d$ is a desired joint angle vector. In order to make the system track $\mathbf{q}(t) \equiv \mathbf{q}_d(t)$, we define a sliding surface $\mathbf{s}$,

$$\mathbf{s} = \dot{\tilde{\mathbf{q}}} + \boldsymbol{\Lambda}\tilde{\mathbf{q}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_r \tag{2}$$

where $\boldsymbol{\Lambda}$ is a positive definite matrix, and the reference velocity vector $\dot{\mathbf{q}}_r = \ddot{\mathbf{q}}_d - \boldsymbol{\Lambda}\tilde{\mathbf{q}}$ [4]. Let us define the Lyapunov function as follows:

$$V(t) = \frac{1}{2}\mathbf{s}^T\hat{\mathbf{H}}\mathbf{s}. \tag{3}$$

By differentiating (3), we can show

$$\dot{V} = \frac{1}{2}\mathbf{s}^T\dot{\hat{\mathbf{H}}}\mathbf{s} + \mathbf{s}^T\hat{\mathbf{H}}\dot{\mathbf{s}} \tag{4}$$

$$= -\frac{1}{2}\mathbf{s}^T(\dot{\mathbf{H}} - \dot{\hat{\mathbf{H}}})\mathbf{s} + \mathbf{s}^T(\boldsymbol{\tau} - \mathbf{C}\dot{\mathbf{q}}_r - \mathbf{g} - \mathbf{H}\ddot{\mathbf{q}}_r) - \mathbf{s}^T(\mathbf{H} - \hat{\mathbf{H}})\dot{\mathbf{s}} \tag{5}$$

where the skew symmetry of $(\dot{\mathbf{H}} - 2\mathbf{C})$ has been used to eliminate the term $\frac{1}{2}\mathbf{s}^T\dot{\mathbf{H}}\mathbf{s}$. The total control input $\boldsymbol{\tau}$ as shown in Figure 1 is

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{eq} + \Delta\boldsymbol{\tau} + \boldsymbol{\tau}_{nn} \tag{6}$$

where $\boldsymbol{\tau}_{eq}$ is the equivalent control input, $\boldsymbol{\tau}_{nn}$ is the output of the neural network, and the chattering control input $\Delta\boldsymbol{\tau}$ is as follows:

$$\Delta\boldsymbol{\tau} = -\mathbf{K}_1\sigma(\mathbf{s}/\Phi) - \mathbf{K}_2\mathbf{s} \tag{7}$$

ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 193-198

where $\mathbf{K}_1$ and $\mathbf{K}_2$ are positive diagonal matrices and $\sigma(\mathbf{s}/\Phi)$ is defined as follows:

$$\sigma(\mathbf{s}/\Phi) = \begin{cases} sgn(\mathbf{s}) & \text{if } \|\mathbf{s}\| \geq \Phi, \\ \mathbf{s}/\Phi & \text{if } \|\mathbf{s}\| \leq \Phi \end{cases} \tag{8}$$

where $sgn$ is the sign function, and $\Phi$ is the boundary layer thickness. $\boldsymbol{\tau}_{eq}$ is the control input vector which would make $\dot{V}$ equal 0 if the dynamics were exactly known

$$\boldsymbol{\tau}_{eq} = \hat{\mathbf{H}}\ddot{\mathbf{q}}_r + \hat{\mathbf{C}}\dot{\mathbf{q}}_r + \hat{\mathbf{g}}. \tag{9}$$

We then have

$$\dot{V} = \mathbf{s}^T \left( \Delta\boldsymbol{\tau} + \boldsymbol{\tau}_{nn} + \tilde{\mathbf{H}}\ddot{\mathbf{q}}_r + \tilde{\mathbf{C}}\dot{\mathbf{q}}_r + \tilde{\mathbf{g}} + \frac{1}{2}\dot{\tilde{\mathbf{H}}}\mathbf{s} + \tilde{\mathbf{H}}\dot{\mathbf{s}} \right) \tag{10}$$

where the modeling errors $\tilde{\mathbf{H}}$, $\tilde{\mathbf{C}}$, and $\tilde{\mathbf{g}}$ are $\tilde{\mathbf{H}} = \hat{\mathbf{H}} - \mathbf{H}$, $\tilde{\mathbf{C}} = \hat{\mathbf{C}} - \mathbf{C}$, and $\tilde{\mathbf{g}} = \hat{\mathbf{g}} - \mathbf{g}$, respectively. Let us define the total uncertainty $\mathbf{u}$ as follows,

$$\mathbf{u} = \tilde{\mathbf{H}}\ddot{\mathbf{q}}_r + \tilde{\mathbf{C}}\dot{\mathbf{q}}_r + \tilde{\mathbf{g}} + \frac{1}{2}\dot{\tilde{\mathbf{H}}}\mathbf{s} + \tilde{\mathbf{H}}\dot{\mathbf{s}}. \tag{11}$$

From (4), (10) and (11), as shown in Figure 1,

$$\frac{1}{2}\dot{\hat{\mathbf{H}}}\mathbf{s} + \hat{\mathbf{H}}\dot{\mathbf{s}} = \Delta\boldsymbol{\tau} + \boldsymbol{\tau}_{nn} + \mathbf{u}. \tag{12}$$

We can obtain the cost function of the neural network as follows,

$$E = \left\| \Delta\boldsymbol{\tau} - \frac{1}{2}\dot{\hat{\mathbf{H}}}\mathbf{s} - \hat{\mathbf{H}}\dot{\mathbf{s}} \right\|^2 = \left\| -\mathbf{u} - \boldsymbol{\tau}_{nn} \right\|^2. \tag{13}$$

If the neural network successfully learned the total uncertainty $\mathbf{u}$, the error dynamic equation becomes

$$\mathbf{u} + \boldsymbol{\tau}_{nn} = \boldsymbol{\epsilon} \tag{14}$$

where $\boldsymbol{\epsilon}$ denotes a functional reconstruction error. Therefore, the derivative of the Lyapunov function (10) becomes

$$\begin{aligned} \dot{V} &= \mathbf{s}^T(\mathbf{u} + \boldsymbol{\tau}_{nn} + \Delta\boldsymbol{\tau}) = \mathbf{s}^T(\boldsymbol{\epsilon} - \mathbf{K}_1\sigma(\mathbf{s}/\Phi) - \mathbf{K}_2\mathbf{s}) \\ &\leq -\lambda_{\min}(\mathbf{K}_2)\|\mathbf{s}\|^2 + \|\boldsymbol{\epsilon}\| \cdot \|\mathbf{s}\| - \mathbf{s}^T\mathbf{K}_1\sigma(\mathbf{s}/\Phi) \end{aligned} \tag{15}$$

where $\lambda_{min}(\mathbf{K}_2)$ denotes the minimum eigenvalue of $\mathbf{K}_2$. If $\|\mathbf{s}\| \geq \|\boldsymbol{\epsilon}\|/\lambda_{min}(\mathbf{K}_2)$,

$$\dot{V} \leq -\mathbf{s}^T\mathbf{K}_1\sigma(\mathbf{s}/\Phi). \tag{16}$$

It means that the surface $\|\mathbf{s}\| = \|\boldsymbol{\epsilon}\|/\lambda_{min}(\mathbf{K}_2)$ will be reached in a finite time. Bounds on $\mathbf{s}$ can be directly translated into bounds on the tracking error vector

ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 193-198

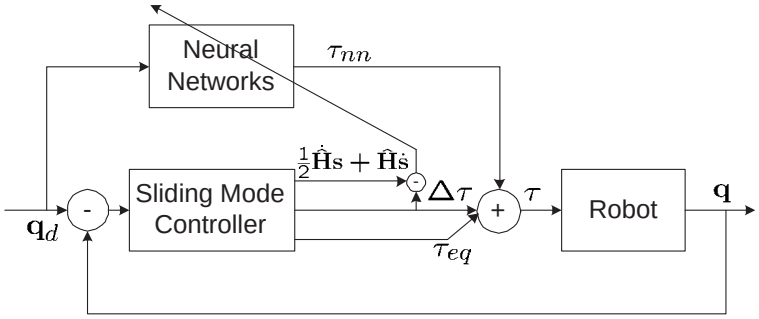


Figure 1: The proposed sliding mode controller using neural network.

$\tilde{\mathbf{q}}$. By equation (2), the tracking error vector $\tilde{\mathbf{q}}$ is obtained from $\mathbf{s}$ through a first-order lowpass filter [4]. From $\|\mathbf{s}\| \leq \|\boldsymbol{\epsilon}\|/\lambda_{min}(\mathbf{K}_2)$ we get

$$\|\tilde{\mathbf{q}}\| \leq \frac{\|\boldsymbol{\epsilon}\|}{\lambda_{min}(\mathbf{K}_2)\lambda_{min}(\boldsymbol{\Lambda})}, \quad \|\dot{\tilde{\mathbf{q}}}\| \leq \frac{\|\boldsymbol{\epsilon}\|}{\lambda_{min}(\mathbf{K}_2)}\left(1 + \frac{\lambda_{max}(\boldsymbol{\Lambda})}{\lambda_{min}(\boldsymbol{\Lambda})}\right). \tag{17}$$

If the neural network successfully learned uncertainties so that $\boldsymbol{\epsilon} \to 0$, it means that $\mathbf{s} \to 0$, and the tracking error vector $\tilde{\mathbf{q}} \to 0$ and $\dot{\tilde{\mathbf{q}}} \to 0$. Also, $\Delta\boldsymbol{\tau} \to 0$ from (7).

# 3 Simulation

In order to show the performance of the proposed controller, we performed the computer simulation using a two-link planar robot manipulator, whose dynamics can be written explicitly as [6]

$$\begin{bmatrix} H_{12} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} -C_{12}\dot{q}_2 & -C_{12}(\dot{q}_1 + \dot{q}_2) \\ C_{12}\dot{q}_1 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} g_1 g \\ g_2 g \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{18}$$

where $H_{11} = (m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(q_2)$, $H_{12} = H_{21} = m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2)$, $H_{22} = m_2 l_2^2$, $C_{12} = m_2 l_1 l_2 \sin(q_2)$, $g_1 = (m_1 + m_2) l_1 \cos(q_2) + m_2 l_2 \cos(q_1 + q_2)$, $g_2 = m_2 l_2 \cos(q_1 + q_2)$, $f_1 = c_1 sgn(\dot{q}_1) + v_1 \dot{q}_1$, $f_2 = c_2 sgn(\dot{q}_2) + v_2 \dot{q}_2$, and $g$ is the acceleration of gravity. $f_1$ and $f_2$ are nonlinear frictions. The parameter values are selected as $c_1 = c_2 = 0.2$, $v_1 = 3$, $v_2 = 2$, $l_1 = 1[m]$, $l_2 = 0.8[m]$, $m_1 = 1[kg]$ and $m_2 = 0.8 \to 1.5[kg]$ where the payload $m_2$ is changed at 5 seconds after simulation begins. The desired trajectories are $\mathbf{q}_d(t) = \begin{bmatrix} \frac{\pi}{6}(1 - \cos(2\pi t)) & \frac{\pi}{4}(1 - cos(2\pi t)) \end{bmatrix}^T$. The initial states are chosen as $\mathbf{q}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, $\dot{\mathbf{q}}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$. The control parameters used in the proposed method are $\Phi = 0.2$, $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{I}$, $\boldsymbol{\Lambda} = 10\mathbf{I}$ where $\mathbf{I}$ is an identity matrix. We used the two-layer neural network with eight hidden neurons and $q_{d1}, q_{d2}, \dot{q}_{d1}, \dot{q}_{d2}$ as the inputs of the neural network. The outputs of the neural

ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 193-198

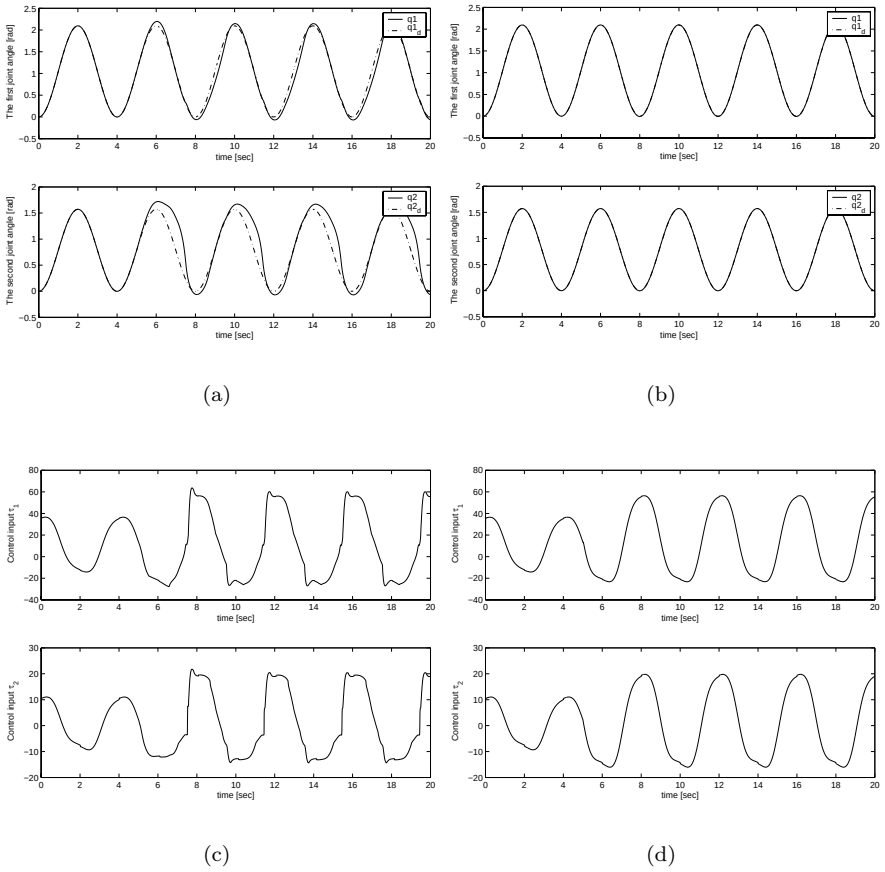


Figure 2: Tracking trajectories and control torques. (a) The tracking trajectory of the conventional sliding mode controller. (b) The tracking trajectory of the proposed controller. (c) The control torque of the conventional sliding mode controller. (d) The control torque of the proposed controller.

network generate the input torques to compensate for the uncertainties. Initial weights are chosen as small random values bounded by $\pm 0.05$. The fourth-order Runge-Kutta method with $10[ms]$ sampling time is used to solve the differential equation (18). Figure 2 shows the tracking performances of the conventional sliding mode controller using boundary layer method [4] and the proposed controller, where the dashdot lines show the desired target trajectories and the solid lines represent the controlled results. Table 1 shows the comparative performances of the conventional sliding mode controller and the proposed controller. The tracking performance of the proposed controller is better than that of the conventional controller in tracking errors and overall control torque.

Table 1: Numerical comparisons between the conventional sliding mode controller and the proposed controller.

| | The conventional sliding mode controller | | The proposed controller | |
|---|---|---|---|---|
| | First link | Second link | First link | Second link |
| Tracking error* | 18.4336 | 102.576 | 0.0109 | 0.0042 |
| Control torque* | 1,900,990 | 282,340 | 1,843,480 | 302,413 |

* This is a sum-squared value.

# 4 Conclusion

In this paper, the sliding mode controller using neural networks is proposed for a robot manipulator. The proposed method used the neural network to compensate for the system uncertainties. According to the simulation results, the neural network, if trained successfully on-line, compensates for the uncertainties, and reduces tracking errors and overall control torque. As a further work, an experiment for more realistic robot manipulator using the proposed controller is under investigation.

# References

[1] J. Y. Hung, W. Gao and J. C. Hung, "Variable structure control: a survey," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, pp. 2-22, Feb. 1993.

[2] Minho Lee and Hyeung-Sik Choi, "A robust neural controller for underwater robot manipulators," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1465-1470, Nov. 2000.

[3] H. K. Khalil, *Nonlinear systems*, 3rd ed., Prentice-Hall, 2002.

[4] J. E. Slotine and W. Li, *Applied nonlinear control*, Pentice-Hall, 1991.

[5] D. E. Rumelhart and J. L. McClelland, *Parallel distributed processing: explorations in the microstructure of cognition*, vol. 1, MIT Press, 1986.

[6] C. Y. Su and T. P. Leung, "A sliding mode controller with bound estimation for robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 208-214, Apr. 1993.