International Journal of Control, Automation and Systems 16(2) (2018) 937-943
http://dx.doi.org/10.1007/s12555-017-0186-z
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

# Improvement of Tracking Control of a Sliding Mode Controller for Robot Manipulators by a Neural Network

**Seul Jung**

**Abstract:** This article presents a neural network control technique to improve the tracking performance of a robot manipulator controlled by the sliding mode control method in a non-model-based framework. The sliding mode controller is a typical nonlinear controller that has been well developed in theory and used in many applications due to its simplicity and practicality. Selection of the gain of the nonlinear function plays an important role in performance as well as stability. When the sliding mode controller is used for the non model-based configuration in robot control, the nonlinear gain should be selected large enough to guarantee the stability. Since the appropriate selection of the gain value is essential and difficult in the sliding mode control framework, a neural network compensator is introduced at the trajectory level to help the fixed gain deal with the stability and performance more intelligently. Stability of the proposed control scheme is analyzed. Simulation studies of following the Cartesian trajectory for a three-link rotary robot manipulator are conducted to confirm the control improvement by the neural network.

**Keywords:** Neural network, reference compensation technique, robot manipulators, sliding mode control.

## 1. INTRODUCTION

The disturbance observer (DOB) technique has been actively used in motion control to achieve the robustness [1, 2]. One weak point of the DOB scheme is that the structure requires the inverse model of the system, which is difficult to obtain, especially for the nonlinear systems [3]. An adaptive control approach with dynamic programming has been presented as well [4, 5].

The sliding mode control method is one of non-model-based control candidates for nonlinear systems that have been well developed theoretically. The performance of the sliding mode control method depends upon the selection of the nonlinear gain suitably to guarantee the global stability. When the sliding mode controller is used for the non model-based configuration in robot control, the gain for the nonlinear function should be selected with care to satisfy the stability. The gain value becomes relatively large compared with the model–based control method and this leads to the chattering behavior of the output.

Therefore, intelligent tools may compensate for the lack of the sliding mode control method. Combining the sliding mode control method with AI technologies will provide the better performance to control systems. Appropriate intelligent tools such as neural network, fuzzy logic, and genetic algorithms can be applied to control systems to satisfy performance specifications or at least to improve the performance.

Combing the neural network with the primary control methods has been actively accomplished to have the better performance in the literature [6–13]. Neural networks have been added to the controlled system as an auxiliary controller to achieve the real-time control performance. Various neural network control schemes have been presented.

The feedback error learning (FEL) control scheme replaces the feed-forward controller with a neural network in the feedback control scheme [6]. The stability analysis of a radial basis function (RBF) network control scheme has been done for a robot manipulator [11]. Control applications with the stability analysis of the FEL scheme for robot manipulator have been extensively presented by reformulating robot dynamic equations with tracking error functions along with simulation results [9].

Neural network control schemes along with the sliding mode control for robot manipulators have been presented [10–12].

In other aspects, differing the compensation location can provide a structural advantage without bothering the modification of the inside controllers while achieving the same goal of the FEL scheme. This scheme is known as the reference compensation technique (RCT) [13]. The real implementation of the RCT scheme has been presented along with the stability analysis of the RCT scheme

Manuscript received April 2, 2017; revised August 24, 2017; accepted October 26, 2017. Recommended by Associate Editor Kang-Hyun Jo under the direction of Editor Euntai Kim. This work has been supported by the basic research funds through the contract of National Research Foundation of Korea (2016R1A2B2012031).

Seul Jung is with the Department of Mechatronics Engineering, Chungnam National University, 99 Daehak-ro, Yuseong-gu, Daejeon 34134, Korea (e-mail: jungs@cnu.ac.kr).

for controlling robot manipulators by neural networks .

In this paper, since the appropriate selection of the gain value in the sliding mode control is essential, a neural network is introduced at the trajectory level by forming the RCT scheme to achieve the effect of selecting the appropriate gain of the sliding mode control indirectly. The stability of the proposed control scheme is analyzed.

Finally, simulation studies of following the Cartesian trajectory of the three-link rotary robot manipulator under nonlinear uncertainties are conducted to confirm the performance of the neuro-sliding mode control scheme. Performances of tracking control by both the sliding mode control and the neuro-sliding mode control scheme are compared.

## 2. ROBOT MANIPULATOR DYNAMICS

In general, the dynamic equation of an n degrees-of-freedom robot manipulator in the joint space coordinates is described by

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau, \tag{1}$$

where $q$ is the $n\times 1$ joint angle vector, $\dot{q}$ is the $n\times 1$ joint angular velocity vector, and $\ddot{q}$ is the $n\times 1$ joint angular acceleration vector, $D(q)$ is the $n\times n$ symmetric positive definite inertia matrix, $C(q,\dot{q})\dot{q}$ is the $n\times 1$ Coriolis and centrifugal torque vector, $G(q)$ is the $n\times 1$ gravitational torque vector, and $\tau$ is the $n\times 1$ joint torque vector.

Define the joint trajectory tracking errors as

$$e=q_d-q,\ \dot{e}=\dot{q}_d-\dot{q},\ \ddot{e}=\ddot{q}_d-\ddot{q}, \tag{2}$$

where $q_d$ is the reference trajectory. From (2), the error surface functions are defined as

$$s=\dot{e}+\lambda e,\ \dot{s}=\ddot{e}+\lambda\dot{e}, \tag{3}$$

where $\lambda$ is a positive constant.

Rearranging $\dot{q}$, $\ddot{q}$ from (3) in terms of $s,\dot{s}$ yields

$$\dot{q}=\dot{q}_d-(s-\lambda e),\ \ddot{q}=\ddot{q}_d-(\dot{s}-\lambda\dot{e}). \tag{4}$$

Substituting $\dot{q}$, $\ddot{q}$ of (4) into the dynamic equation (1) yields the modified dynamic equation of a robot manipulator in terms of $s$.

$$\tau_m(q,\dot{q})-(D(q)\dot{s}+C(q,\dot{q})s)=\tau, \tag{5}$$

where $\tau_m(q,\dot{q})=D(q)(\ddot{q}_d+\lambda\dot{e})+C(\dot{q}_d+\lambda e)+G(q)$ which includes all the uncertainties of a robot manipulator. This term is expected to be cancelled out by the sliding mode controller with the help of a neural network. As a result, equation (5) satisfies the tracking performance as well as the stability when the controller is designed with the function of the error surface $s$.

## 3. NEURAL NETWORK

A radial basis function (RBF) network has been popular in control applications owing to its simplicity and a fast learning structure for a real-time control fashion. A Gaussian function is used as the nonlinear function that measures the Euclidean distance among input data. Although the RBF network has a structural advantage, the inside weights are quite sensitive that the appropriate selection of learning rates and initial weight values are required which is not easy.

Here a RBF-like multilayered perceptron network is used as in [14]. The RBF-like network has only one hidden layer and the output is linear as shown in Fig. 1.

The nonlinear function of the hidden layer is described as

$$\psi_j(e_i)=\frac{1-\exp(-\sum_{i=0}^{N_I}e_i)}{1+\exp(-\sum_{i=0}^{N_I}e_i)}, \tag{6}$$

where $e_i$ is the ith input element and $N_I$ is the number of input elements.

The output of the hidden layer is summed together to form the output.

$$\tau_k=\sum_{j=1}^{N_H}\psi_j w_{jk}+b_k, \tag{7}$$

where $\psi_j$ is $j$th output of the hidden layer and $w_{jk}$ is the weight between the $j$th hidden unit and the $k$th output, $b_k$ is the bias weight of the $k$th output, and $N_H$ is the number of hidden units.

Fig. 1 shows the RBF-like multilayer perceptron network (MLP) that does not have the weights between the input and hidden layer. The advantage of this network is that the number of the update weights is a half of the RBF network. This may lose the flexibility of a nonlinear mapping ability of the MLP network, but the simplified

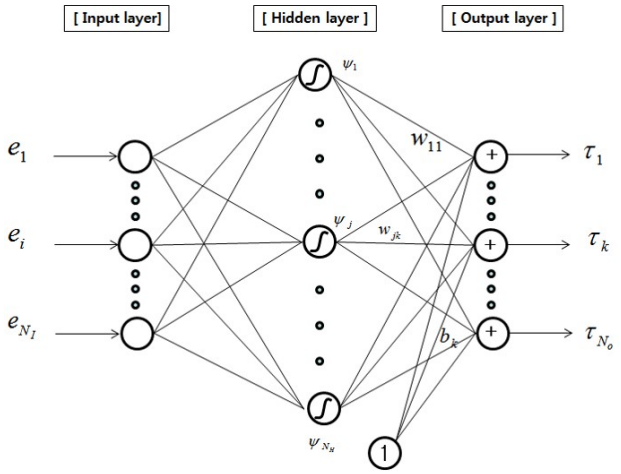



Fig. 1. Neural network structure.

network may perform evenly for the control applications that are not highly nonlinear.

## 4. SLIDING MODE CONTROL

### 4.1. Model-based sliding mode control scheme

The sliding mode control is a well-known nonlinear control method used for both model-based and non model-based control schemes. The error surface is defined as in (3) for the control input to push states to the error surface that guarantees zero.

The control law of the model based-sliding mode control is given as

$$\tau = \hat{\tau}_m + \tau_{smc\_m}, \tag{8}$$

where $\hat{\tau}_m$ is the estimate of model-based control $\tau_m$ and $\tau_{smc\_m}$ is a sliding mode control. Equation (8) is described as

$$\tau = \hat{D}(\ddot{q}_d + \lambda\dot{e}) + \hat{C}(\dot{q}_d + \lambda e) + \hat{G} + K_s sgn(s), \tag{9}$$

where $\hat{D}$, $\hat{C}$, $\hat{G}$ are the estimates of the robot dynamic model and $K_s$ is a gain for the nonlinear function of the sliding mode.

For stability, the Lyapunov function is defined as

$$V = \frac{1}{2}s^T Ds. \tag{10}$$

Differentiating the Lyapunov function V yields

$$\dot{V} = \frac{1}{2}s^T\dot{D}s + s^T D\dot{s}. \tag{11}$$

Substituting $D\dot{s}$ from (5) into (11) yields

$$\begin{aligned}\dot{V} &= \frac{1}{2}s^T\dot{D}s + s^T(\tau_m - Cs - \tau) \\ &= \frac{1}{2}s^T(\dot{D} - 2C)s + s^T(\tau_m - \tau).\end{aligned} \tag{12}$$

Applying the skew symmetry of $\dot{D} - 2C = 0$ to (12) yields

$$\dot{V} = s^T(\tau_m - \tau). \tag{13}$$

Substituting the control law in (9) into (13) yields

$$\dot{V} = s^T(\Delta\tau_m - K_s sgn(s)), \tag{14}$$

where $\Delta\tau_m = \tau_m - \hat{\tau}_m = \Delta D(\ddot{q}_d + \lambda\dot{e}) + \Delta C(\dot{q}_d + \lambda e) + \Delta G$ and $\Delta D = D - \hat{D}$, $\Delta C = C - \hat{C}$, $\Delta G = G - \hat{G}$.

If the estimated model is correct, then $\Delta\tau_m = 0$ leads to the stability as

$$\dot{V} = -s^T K_s sgn(s) < 0. \tag{15}$$

It satisfies the Lyapunov stability condition.

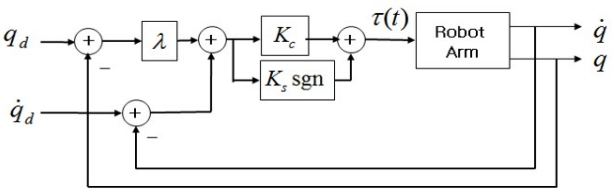



Fig. 2. Non model-based sliding mode control structure.

However, there are always the modeling errors. Then we have the following condition to be satisfied for the stability.

$$k_m > |\Delta\tau_m|, \tag{16}$$

where $k_m$ is the smallest positive eigenvalue of the gain matrix $K_s$.

The gain $K_s$ can be selected to satisfy (16) depending upon the magnitude of $\Delta\tau_m$ for the stability, but we still have the inherent chattering problem.

### 4.2. Non model-based sliding mode control scheme

For the nonmodel-based control case, the control law becomes

$$\begin{aligned}\tau &= \tau_n + \tau_{smc\_n} \\ &= K_c(\dot{e} + \lambda e) + K_s sgn(s),\end{aligned} \tag{17}$$

where $\tau_n$ is the feedback controller, $K_s$ and $K_c$ are a positive gain matrix of the sliding mode controller. The control block diagram is shown in Fig. 2.

From (13), we have

$$\dot{V} = s^T(\Delta\tau_n - K_s sgn(s)), \tag{18}$$

where $\Delta\tau_n = \tau_m - \tau_n = D(\ddot{q}_d + \lambda\dot{e}) + C(\dot{q}_d + \lambda e) + G - K_c(\dot{e} + \lambda e)$.

To satisfy the stability, we have the magnitude condition

$$k_m > |\Delta\tau_n|. \tag{19}$$

This means that the gain $K_s$ has to cover all the dynamics of the robot manipulator. We also know that $k_m$ in (19) should be selected as a much larger value than that in (16). It is clear that a larger gain may cause a chattering problem severely. The fixed controller gain $K_s$ may not satisfy the stability when (19) does not hold.

Therefore, our proposal here is to introduce a neural network to compensate for the dynamics of a robot manipulator to reduce $\Delta\tau_n$ further to minimize the burden of $K_s$ such that (19) can be satisfied under the constant gain $K_s$ with ease in the non-model-based control configuration.

## 5. NEURO-SLIDING MODE CONTROL

### 5.1. Neuro-sliding mode control scheme

A RBF-like neural network is added to the sliding mode control for the non model-based control framework as

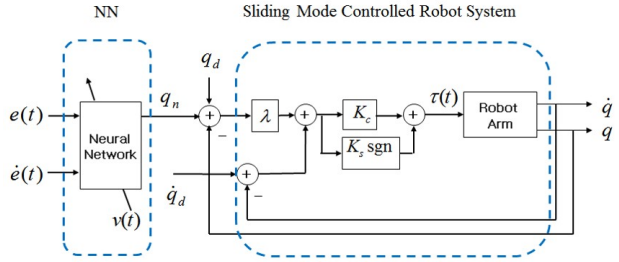



Fig. 3. Neuro-sliding mode control structure.

shown in Fig. 3. A neural network is added at the trajectory level to modify the reference trajectory instead of modifying torque values inside the control structure.

Adding neural network outputs at the trajectory, the control law becomes

$$\begin{aligned}\tau &= K_c(\dot{q}_d - \dot{q} + \lambda(q_d - q + q_n)) + K_s sgn(s)\\ &= K_c(\dot{e} + \lambda e) + K_s sgn(s) + K_c\lambda q_n\\ &= K_c s + K_s sgn(s) + K_c\lambda q_n,\end{aligned} \tag{20}$$

where $K_c$ is the constant controller gain.

Combining (20) with the dynamic equation (5) yields the closed loop error equation as

$$D\dot{s} + (C + K_c)s = \tau_m(q,\dot{q}) - K_c\lambda q_n - K_s sgn(s). \tag{21}$$

Note that (21) tells us that the robot dynamic equation is compensated by both the sliding mode controller and the neural network controller.

## 5.2. Learning process

Learning of the neural network uses the back-propagation algorithm to update weights. Then we have to define the learning signal to form the objective function to be minimized.

From (20), we define the learning signal as

$$v = K_c s + K_s sgn(s) = \tau - K_c\lambda q_n. \tag{22}$$

Since we need to minimize the feedback control error $v$ instead of e, the objective function is formed as

$$E = \frac{1}{2}v^T v, \tag{23}$$

where $v \in R^{n\times 1}$ is considered as the training signal.

The gradient with respect to the weight is used and calculated to minimize the objective function as

$$\Delta w = -\eta\frac{\partial E}{\partial w} = -\eta\frac{\partial E}{\partial v}\frac{\partial v}{\partial w}, \tag{24}$$

where $\eta$ is the learning rate and $w$ is the weight vector. The gradient can be obtained from (22) as

$$\Delta w = -\eta\frac{\partial v}{\partial w}v = \eta'[\frac{\partial q_n}{\partial w}]^T K_c v, \tag{25}$$

where $\eta' = \eta\lambda$ is the learning rate.

Weights are updated at every sampling time as

$$w(t+1) = w(t) + \Delta w(t) + \alpha\Delta w(t-1), \tag{26}$$

where $\alpha$ is the momentum constant.

## 5.3. Stability analysis

Based on the neuro-sliding mode control structure, stability is analyzed. The universal approximation property of the neural network can approximate any nonlinear functions as

$$\tau_m(q,\dot{q}) = \Phi(w,e,\dot{e}) = K_c\lambda W^T\psi + \varepsilon, \tag{27}$$

where $W$ is the weight matrix, $\psi$ is the output vector of the hidden layer and $\varepsilon$ is the approximation error vector.

From (20) and (27), the control law becomes the multiplication of the gain $K_c$ to the addition of the feedback controller and the neural network output.

$$\tau = K_c(\lambda\hat{W}^T\psi + s) + K_s sgn(s). \tag{28}$$

Substituting (27) and (28) into (5) yields the closed loop error equation as

$$D\dot{s} + (C + K_c)s = K_s sgn(s) + K_c\lambda\tilde{W}^T\psi + \varepsilon, \tag{29}$$

where $\tilde{W}^T = W^T - \hat{W}^T$ and $W^T$ is the true weights and $\hat{W}^T$ is the approximation of the true weight.

Define the Lyapunov function as

$$L = \frac{1}{2}s^T Ds + \frac{1}{2}Tr\{\tilde{W}^T\Gamma^{-1}\tilde{W}\}, \tag{30}$$

where $Tr$ is Trace of the matrix and $\Gamma^{-1}$ is positive constant matrix.

Differentiating (30) yields

$$\dot{L} = \frac{1}{2}s^T\dot{D}s + s^T D\dot{s} + Tr\{\tilde{W}^T\Gamma^{-1}\dot{\tilde{W}}\}. \tag{31}$$

From (29), we have

$$D\dot{s} = K_c\lambda\tilde{W}^T\psi + K_s sgn(s) - (C + K_c)s + \varepsilon. \tag{32}$$

Substituting $D\dot{s}$ of (32) into (31) yields

$$\begin{aligned}\dot{L} =& \frac{1}{2}s^T(\dot{D} - 2C)s - s^T K_c s - s^T K_s sgn(s)\\ &+ Tr\{\tilde{W}^T(\Gamma^{-1}\dot{\tilde{W}} + K_c\lambda\psi s^T)\} + s^T\varepsilon.\end{aligned} \tag{33}$$

Using the skew symmetry property of a robot manipulator $\dot{D} - 2C = 0$ simplifies (33) as

$$\begin{aligned}\dot{L} =& -s^T K_c s - s^T K_s sgn(s)\\ &+ Tr\{\tilde{W}^T(\Gamma^{-1}\dot{\tilde{W}} + K_c\lambda\psi s^T)\} + s^T\varepsilon.\end{aligned} \tag{34}$$

To guarantee the stability for $\dot{L} < 0$, we select the update law for the weights as

$$\dot{\hat{W}} = \dot{W} - \dot{\hat{W}} = -\dot{\hat{W}} = -\Gamma\lambda K_c \psi s^T. \quad (35)$$

Substituting the update law $\dot{\hat{W}} = \Gamma\lambda K_c \psi s^T$ into (34) yields

$$\dot{L} = -s^T K_c s - s^T K_s sgn(s) + s^T \varepsilon. \quad (36)$$

To satisfy $\dot{L} < 0$ for the stability in (36), we can select the sliding mode gain $K_s$ large enough to satisfy the following condition, which can be easily done such that

$$|s^T K_s sgn(s)| > |s^T \varepsilon|. \quad (37)$$

Equation (37) can be represented as

$$k_m |s| > |s^T \varepsilon|. \quad (38)$$

It can further simplified as

$$k_m > |\varepsilon|. \quad (39)$$

Therefore, the stability is a matter of selection of the gain $k_m$ to satisfy the relationship of (39).

Comparing with the equation (16), we see here that the neural network helps the sliding mode controller to guarantee the stability although the modeling error is not known.

## 6. SIMULATION

### 6.1. Desired trajectory

A three-link rotary robot manipulator is controlled in the Cartesian space. The robot is a type of industrial robot that has the mass of each link such as $m1 = 30$ Kg, $m2 = 17.4$ Kg, $m3 = 4.8$ Kg, and its length as $l1 = 0.66$ m, $l2 = 0.43$ m, $l3 = 0.43$ m. Initial joint angles of the robot are set to $q = [0.6378 \;\; 0.8260 \;\; -1.2445]^T$ as shown in Fig. 4. The robot is commanded to follow the circular trajectory in the Cartesian space.

Desired trajectory is slanted by 45 degrees in x and z axis as shown in Fig. 4.

$$\begin{aligned} x_d &= 0.408 + \cos(-\pi/4)(0.2\cos(t)), \\ y_d &= 0.408 + 0.2\sin(t), \\ z_d &= 0.66 - \sin(-\pi/4)(0.2\cos(t)). \end{aligned} \quad (40)$$

Friction terms are added to each joint to see the effect of nonlinear uncertainties. The friction model becomes

$$f_i = sign(q_i) * (k_1 * abs(q_i) + k_2), \quad (41)$$

where $k_1$, $k_2$ are friction coefficients.

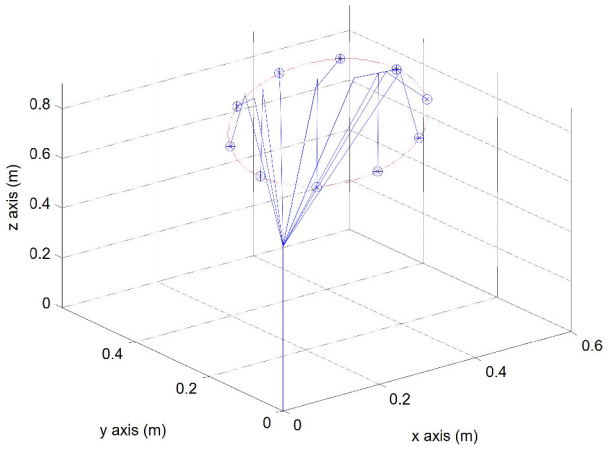



Fig. 4. Desired trajectory.

### 6.2. Sliding mode control

Sliding mode gains are set to $\lambda = 10$, $K_c = daig[1,1,1,]$ and $K_s = diag[100,100,100]$, respectively for three joints. Gains are selected to have the stable response. It turned out that other lower gains gave the unstable responses.

Fig. 5 shows the tracking performances by the sliding mode controllers. In the Cartesian space, the robot controlled by the sliding mode controller follows the desired trajectory well as shown in Fig. 5(a). The corresponding joint tracking performances are plotted in Fig. 5(b). The joint angle errors are also plotted in Fig. 5(c). We see that the tracking errors of joint 2 and 3 are larger than that of joint 1.

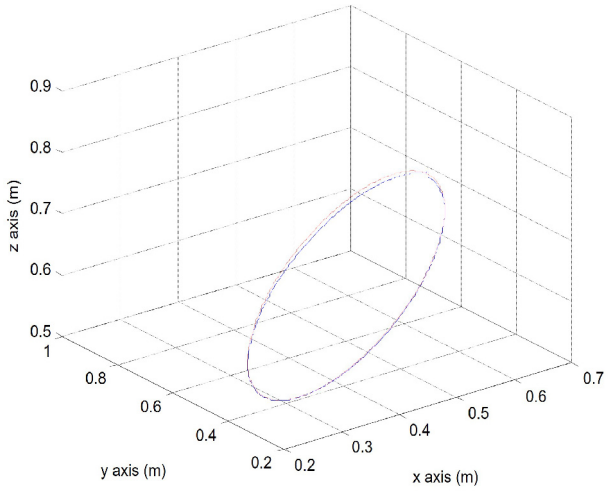
### 6.3. Neuro-sliding mode control

The same task is coducted by the neuro-sliding mode control method here. The same controller gains are used. The neural network has 6 hidden units. The learning rate of 0.0002 is tested.

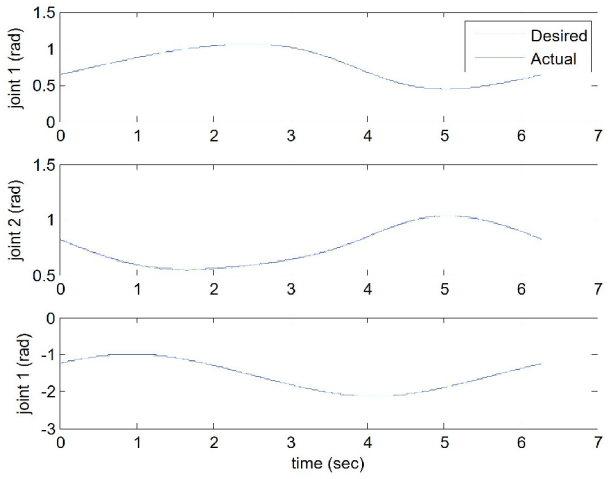
Tracking performances by the neuro-sliding mode control method are shown in Fig. 6. Comparing Fig. 6(a) with Fig. 5(a) shows the better tracking performance by the neuro-sliding mode control scheme. The corresponding joint angles and errors are shown in Fig. 6(b) and (c), respectively. We see that the tracking performance by the neuro-sliding mode control method has been improved, espeicially joint 2 error has been reduced.

We also plotted neural network output signals in Fig. 7. Compensating signals are small since controller gains are multiplied to neural network outputs to generate the torque. Compensating signal at joint 2 is relatively large at the beginning since we have observed the larger error in Fig. 6(c).

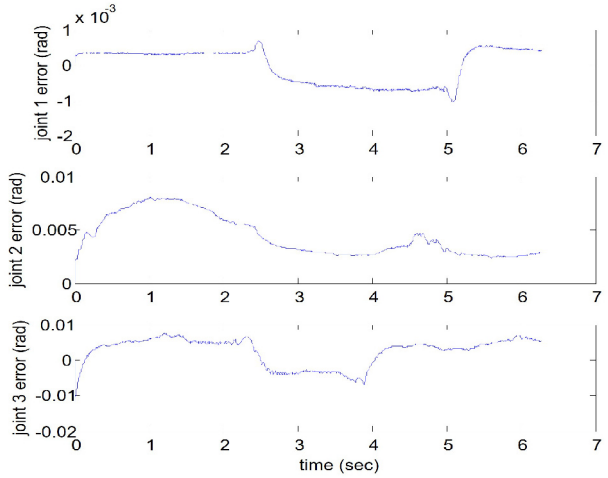
For the numerical comparison, we have summarized the errors of the Cartesian tracking performances in Table 1. The root-mean-square errors of the joint tracking errors for the sliding mode control and the neuro-sliding mode




(a) Tracking performancein Cartesian space.

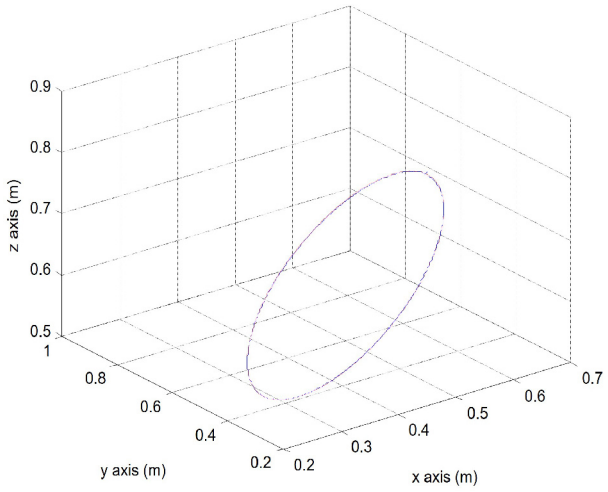



(b) Joint tracking performance.

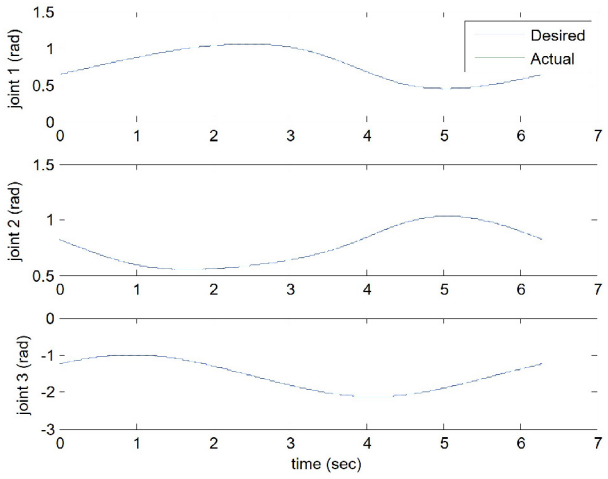



(c) Joint tracking error.

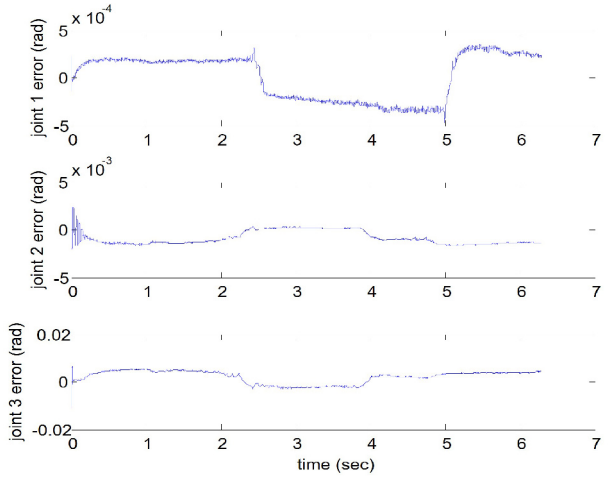
Fig. 5. Sliding mode control performance.




(a) Tracking performancein Cartesian space.




(b) Joint tracking performance.




(c) Joint tracking error.

Fig. 6. Neuro-sliding mode control performance.

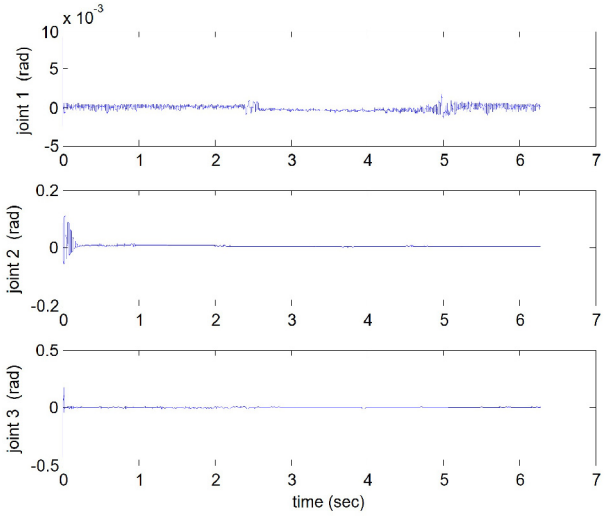



Fig. 7. NN compensating signals.

Table 1. Cartesian tracking error.

| | Sliding mode control | Neuro-sliding mode control |
|---|---|---|
| Error (m) | 0.0062 | 0.0018 |

Table 2. Joint tracking error.

| | Sliding mode control | Neuro-sliding mode control |
|---|---|---|
| Joint 1(rad) | 4.9740e-04 | 2.3973e-04 |
| Joint 2(rad) | 0.0048 | 0.0011 |
| Joint 3(rad) | 0.0046 | 0.0034 |

control are compared in Table 2. We clearly see that the neuro-sliding controller works better.

## 7. CONCLUSION

The neuro-sliding mode control of the reference compensation technique has been tested for a non-model based control framework. The effect of selecting appropriate gain values in the sliding mode control has been achieved by introducing a neural network at the trajectory level. The RBF network-like MLP has been used to compensate for uncertainties in a three-link robot manipulator. Stability of the neuro-sliding mode control scheme has been analyzed. Simulation studies showed that the tracking performance of the neuro-sliding mode control scheme was better than that of the sliding mode control scheme alone.

## REFERENCES

[1] E. Sariyildiz and K. Ohnishi, "On the explicit robust force control via disturbance observer," *IEEE Trans. Industrial Electronics*, vol. 62, no. 3, pp. 1581-1589, 2015. [click]

[2] K. Kong and M. Tomizuka, "Nominal model manipulation for enhancement of stability robustness for disturbance observer-based systems," *International Journal of Control, Automation, and Systems*, vol. 11, no. 1, pp. 12-20, 2013.

[3] S. D. Lee and S. Jung, "Analysis of time constant effect in the Q filter for designing a disturbance observer: balancing control of a single-wheel robot," *Journal of The Institute of Electronics and Information Engineers*, vol. 53, no. 10, pp. 1711-171, 2016.

[4] W. Gao and Z. P. Jiang, "Adaptive dynamic programming and adaptive optimal output regulation of linear system," *IEEE Trans. on Automatic Control*, vol. 61, no. 12, pp. 4164-4169, 2016. [click]

[5] W. Gao and Z. P. Jiang, "Nonlinear and adaptive suboptimal control of connected vehicles: A global adaptive dynamic programming approach," *Journal of Intelligent & Robotic Systems*, vol. 85, no. 3-4, pp. 597-611, 2017. [click]

[6] H. Gomi and M. Kawato, "Learning control for a closed loop system using feedback error learning," *Proc. of the IEEE International Conf. on Decision and Control*, pp. 3289-3294, 1990.

[7] R. J. Wai and R. Muthusamy, "Fuzzy-neural network inherited sliding-mode control for robot manipulator including actuator dynamics," *IEEE Trans. on Neural Network and Learning Systems*, vol. 24, no. 2, pp. 274-287, 2013. [click]

[8] H. Xin and Q. Chen, "Full-order neural sliding model control of robotic manipulator with unknown dead-zone," *Proc. of Chinese Automation Conference*, pp. 1664-1669, 2015.

[9] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, 1999.

[10] R. G. Rodriguez and V. P. Vega, "Tracking control of robot manipulators using second order neuro sliding mode," *Latin American Applied Research*, vol. 39, no. 4, pp. 285-294, 2009.

[11] H. Zhang, M. Du, and W. Bu, "Sliding mode controller with RBF neural network for manipulator trajectory tracking," *IAENG International Journal of Applied Mathematics*, vol. 45, no. 4, 2015.

[12] T. Liu and S. Yin, "An improved neural network adaptive sliding mode control used in robot trajectory tracking control," *International Journal of Innovative Computing, Information and Control*, vol. 11, no. 5, pp. 1655-1666, 2015.

[13] S. Jung and T. C. Hsia, "Neural network inverse control techniques for PD controlled robot manipulator," *Robotica*, vol. 19, no. 3, pp. 305-314, 2000.

[14] S. Jung, "Stability analysis of reference compensation technique for controlling robot manipulators by neural network," *International Journal of Control, Automation, and Systems*, vol. 15, no. 2, pp. 952-958, 2017. [click]