Fuzzy Logic Based Tuning of Sliding Mode Controller for Robot Trajectory Control

Se-Hee Ryu and Jahng-Hyon Park
School of Mechanical Engineering
Hanyang University, Seoul, Korea
jpark@email.hanyang.ac.kr

Abstract

Sliding control guarantees system performance in the presence of modeling uncertainties and parameter inaccuracies. However this is obtained at the price of high control activity that may exacerbate chattering. As one way to alleviate this problem a continuation method using boundary layer around switching is usually used. In this case selecting the control bandwidth and boundary layer thickness is a crucial problem for the trade-off between sliding performance and chattering. The parameter tuning is usually done by trial-and-errors in practice causing significant effort and time. An auto-tuning method based on fuzzy rules is suggested in this paper. To this end, the tracking error index and chattering index are defined first and then the fuzzy rules are designed to compromise the error and chattering indices. To demonstrate the efficiency of the proposed method a robotic control system is simulated and tested. It is shown that the proposed algorithm is effective to facilitate the parameter tuning for sliding mode controllers.

1 Introduction

Sliding mode control (SMC) is a model-based robust control method capable of compensating system modeling inaccuracy and external disturbances. However this SMC requires instantaneous change of control input that may excite high-frequency dynamics neglected in the course of modeling such as unmodeled structural modes, time delays, and so on [1]. This causes fast, finite-amplitude oscillations known as ‘chattering’, which would result in loud noise, high wear of moving mechanical parts and thus should be definitely eliminated [2].

To alleviate the chattering problem, a continuation method using boundary layer [1-4] has been widely used. The discontinuous control law given by a sign function is replaced by a saturation function. This smoothing of control discontinuity inside boundary layer assigns a low-pass filter structure to the local dynamics of sliding surface thus reduces chattering. In this case control performance and actuator chattering are greatly affected by the values of SMC design parameters, control bandwidth (\(\lambda\)) and boundary layer thickness (\(\phi\)). Therefore those parameters should be tuned properly for an efficient sliding mode control. However since there are too many factors that should be considered in tuning process, it is hard to accomplish this task by an analytical method. Therefore in practice a trial-and-error method is used instead.

In this paper as an alternative way an auto-tuning method of sliding mode control parameters using fuzzy logic is suggested. In order to check the performance of the controller, the tracking error index and the chattering index are defined first. Then the fuzzy rules are designed to adjust the control parameters for a trade-off between the error and chattering indices. In this method the system response is monitored at every constant time period and based on it the control parameters are updated using the fuzzy rules.

For validation of the proposed algorithm trajectory controls of an industrial robot manipulator were tried through experiments and the results show the effectiveness of the algorithm.

2 Tuning of Sliding Mode Control Parameters Using Fuzzy Logic

2.1 Design of Sliding Mode Controller

A sliding controller design that will be used in the auto-tuning algorithm is discussed briefly in the following. For a second-order system, the equation of motion is generally given by

\[
\dot{x} = f + u
\]

where \(u\) is control input, \(x\) is system output, and \(f\) is the system dynamics not exactly known but estimated as \(\hat{f}\). The estimation error on \(f\) is assumed to be bounded by some function \(F\) as follows.

\[
|\hat{f} - f| \leq F
\]
In order to have the system track desired trajectory \( x_d(t) \), we define a sliding surface \( s = 0 \) and obtain the control input to satisfy sliding condition as follows.

\[
\frac{1}{2} \frac{d}{dt} s^2 \leq -\eta |s|
\]  

(3)

where \( \eta \) is a constant related to reaching time to sliding mode and \( s \) is defined as follows.

\[
s = (\frac{d}{dt} + \lambda) \bar{x} = \dot{x} + \lambda \bar{x}
\]  

(4)

where \( \bar{x} = x - x_d \).

If we take sliding mode control input is as follows,

\[
u = \dot{\hat{u}} - k \text{sgn}(s)
\]  

(5)

where \( \dot{\hat{u}} = f - \hat{\dot{f}} + \bar{x} - \lambda \bar{x} \), then \( \dot{s} \) becomes as follows.

\[
\dot{s} = f - \hat{f} - k \text{sgn}(s)
\]  

(6)

And the left term of the above eq. (3) can be expanded as follows.

\[
\frac{1}{2} \frac{d}{dt} s^2 = \dot{s} \cdot s = (f - \hat{f}) s - k |s|
\]  

(7)

From eq. (2) and (7) it can be concluded that the control gain \( k \) in eq. (5) should be set as follows to satisfy the sliding condition, eq. (3).

\[
k = F + \eta
\]  

(8)

If the \( F \) is a known function we can use it to calculate the varying control gain to keep the stability of system. But in many practical cases it is difficult to calculate proper control gain each time, thus a large enough constant can be used instead. On the other hand as stated in the introduction the control input in eq. (5) has discontinuity leading to undesirable chattering. To eliminate it a boundary layer is introduced around the sliding surface and discontinuous control law is replaced by a saturation function that approximates the sign function \[13]. In this case the control input in eq. (5) is changed as follows.

\[
u = \dot{\hat{u}} - k \cdot \text{sat}(s / \phi)
\]  

(9)

Then system trajectory inside the boundary layer can be expressed in terms of the variable \( s \) as follows.

\[
\dot{s} = -k \frac{s}{\phi} + f - \hat{f}
\]  

(10)

In the above the variable \( s \) can be viewed as the output of a first-order filter whose input is the model uncertainty. Thus chattering can be eliminated as long as high-frequency unmodeled dynamics are not excited. Conceptually, the structure of the error dynamics can be summarized by Fig. 1. Since \( \lambda \) is the break-frequency of filter in eq. (4), it must be chosen as small with respect to high-frequency unmodeled dynamics (such as unmodeled structural modes or neglected time delays). Furthermore the boundary layer thickness \( \phi \) can be tuned so that eq. (10) also represents a first-order filter of bandwidth \( \lambda \) using the following relation.

\[
k \frac{\phi}{\phi} = \lambda
\]  

(11)

\[
f - \hat{f} \xrightarrow{\text{1st order filter}} s \xrightarrow{\text{eq. (10)}} \dot{s} \xrightarrow{\text{choice of } \phi} (\frac{1}{(p+\lambda)^n}) \xrightarrow{\text{definition of } s} \ddot{x}
\]  

Fig. 1 Structure of the closed-loop error dynamics

In this case the following equality is obtained from eqs. (8) and (11).

\[
k = F + \eta = \lambda \phi
\]  

(12)

2.2 Definition of the performance measure

The control design parameters to be tuned are \( \lambda \) and \( \phi \). The purpose of tuning is to make the system to follow the reference input accurately without chattering. Therefore it is appropriate to evaluate the sliding mode control performance from the viewpoint of both control error and actuator chattering. For this two indices indicating the extents of the error and chattering are respectively introduced as follows.

\[
E = \frac{c_e \int_{T} |x - x_d| dt}{T} \quad (13)
\]

\[
C = \frac{c_i \int_{T} \int_{T} \frac{1}{\tau \tau_{max}} |x - x_d| dt}{T \tau_{max}} \quad (14)
\]

where \( T \) is the tuning time interval, \( \tau_{max} \) is the maximum actuator output, \( C_e, C_c \) are proportional constants. The index \( E \) in eq. (13) is for the evaluation of control error. This definition reflects the fact that the error should be evaluated based on the magnitude of reference input signal. Another index \( C \) defined in eq. (2) is for the chattering evaluation and means the average absolute second order derivative value of actuator outputs during time interval \( T \) divided by maximum actuator output. This definition reflects the idea that the degree of chattering can be approximatively estimated as the average change rate of the slope of actuator output time history relative to actuator performance capability. As a result smaller values of \( E, C \) are preferred for better tracking and less chattering, respectively.

On the other hand we introduce a sensitivity variable that will be used in fuzzy rules as follows.
\[ \frac{\Delta E}{\Delta \lambda} : \text{sensitivity variable for } \lambda \text{ update} \]  

(15)

where \( \Delta \lambda = \lambda_{\text{new}} - \lambda_{\text{old}} \), \( \Delta E = E_{\text{new}} - E_{\text{old}} \).

Here \( \Delta \lambda \), \( \Delta E \) represent the differences between the current parameter or index value and the former one before tuning. The number \( \Delta E/\Delta \lambda \) is sensitivity indicating the degree of change of error index relative to that of \( \lambda \) value. If it takes large value, then it can be inferred that the control performance is greatly affected by the tuning of parameter value. Therefore in this case small change in \( \lambda \) value should be tried.

2.3 Auto-tuning Algorithm for Control Parameters

Before constituting parameter auto-tuning algorithm it is worthwhile to consider the actual parameter tuning procedure done by a human control designer. At first the user should tune the control gain to guarantee system stability, and then tune \( \lambda \), \( \phi \) values to make \( E \), \( C \) values small enough. In this procedure if the system performance is very sensitive to the change of parameter values, then the amount of parameter change is taken very small, and otherwise vice versa. In this paper auto-tuning algorithm is suggested according to the same procedure.

We herein introduce fuzzy rules and based on the rule the controller should update \( \lambda \), \( \phi \) values automatically using \( E \), \( C \) values to evaluate current system performance and \( \Delta E/\Delta \lambda \) to consider parameter tuning history. The flowchart for the algorithm is displayed in Fig. 2.

![Flowchart of parameter tuning scheme](image)

Fig. 2 Flowchart of parameter tuning scheme

2.4 Fuzzy tuning rules

The fuzzy rules suggested here are mainly composed of two parts as follows.

1. \( k \) tuning rule for stability

2. \( \lambda \), \( \phi \) tuning rule for the avoidance of chattering

For the simplicity all the rules are constituted using \( \lambda \) tuning process. For example for the rule 1 we take the method of tuning only \( \lambda \) value to obtain proper control gain based on eq. (12). And for the rule 2 we also take the method of tuning \( \lambda \) value to obtain less \( E \), \( C \) values. In this case \( \phi \) value is tuned automatically to keep constant based on the following two equalities.

\[ k = \lambda \phi = (\lambda + \Delta \lambda)(\phi + \Delta \phi) \]  

(16)

\[ \Delta \phi = \frac{-\Delta \lambda \phi}{\lambda + \Delta \lambda} \]  

(17)

If large \( \lambda \) is used, a respectable tracking performance can be obtained despite poor dynamic model. At the same time, however, \( \lambda \) should be set lower than the frequency of unmodeled dynamics to avoid chattering. In mechanical systems \( \lambda \) is typically limited by the factors such as structural resonance modes, neglected time delays, sampling rate \([1]\). However because there are so many other factors that could affect the selection of \( \lambda \) such as magnitude and frequency of reference input, payload, etc. \([5-7]\), a trial and error method is commonly employed. Therefore we constitute fuzzy rules using not only \( E \), \( C \) values but also \( \Delta E/\Delta \lambda \) to consider tuning history.

An example for Rule 1 is constituted as follows.

"If \( |E| \) is positive very big, then tune \( \lambda \) without considering chattering" \( \Rightarrow \)

"If \( \Delta E/\Delta \lambda \) is positive big, take \( \Delta \lambda \) as negative small"

In this rule the control gain is tuned for system stability without considering chattering.

And an example for Rule 2 is constituted as follows.

"If \( |E| \) is not positive very big, then tune \( \lambda \) with considering chattering" \( \Rightarrow \)

"If \( \Delta E/\Delta \lambda \) is negative small, take \( \Delta \lambda \) as positive big" & "If \( C \) is positive big, then change \( \lambda \) into \( \lambda_i/2 \)

These rules can be explained graphically considering the procedure of finding \( \lambda \) value minimizing \( E \) value starting from an arbitrary initial condition as shown in Fig. 3. If the initial value of \( \lambda \) is set too high, then the search process finds a local minimum accompanying chattering. To solve this problem Rule 2 should include a guideline that the initial value should be changed according to the chattering. The controller takes new \( \lambda \), \( \phi \) values based on the above rules as follows.

\[ \lambda_{\text{new}} = \lambda_{\text{old}} + \sum_{i=0}^{n} \Delta \lambda_{\text{new}} \]  

(18)

where \( \lambda_{\text{new}} = \lambda_{\text{old}} \) or \( \lambda_{\text{old}}/2 \) (based on the rule)

\[ \phi_{\text{new}} = \phi_{\text{old}} + \Delta \phi \]  

(19)
A part of rules to consider the tuning history are listed in Table 1 and the membership functions are shown in Fig. 4.

![Fig. 3 Process of finding optimal \( \lambda \) value](image)

![Fig. 4 Fuzzy membership functions for control bandwidth update (a) input variable : \( \Delta E/\Delta \dot{\lambda} \) (b) output variable : \( \Delta \dot{\lambda} \)](image)

### Table 1. Fuzzy rules for control bandwidth update

<table>
<thead>
<tr>
<th>Input</th>
<th>NB</th>
<th>NM</th>
<th>NS</th>
<th>Z</th>
<th>PS</th>
<th>PM</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta \dot{\lambda}_{m+1} )</td>
<td>PS</td>
<td>PM</td>
<td>PB</td>
<td>Z</td>
<td>NS</td>
<td>NM</td>
<td>NB</td>
</tr>
</tbody>
</table>

\[ \Delta E_n/\Delta \dot{\lambda} \]

3 Validation of the Algorithm through Experiments

#### 3.1 Modeling of a Robot Manipulator

In this section the sliding mode control algorithm is implemented on a six-link industrial robot shown in Fig. 5. Only axes 2 and 3 are used out of the six links for simplicity. In order to incorporate a varying disturbance to the system the break of axis 4 is loosened, so the loads to axes 2 and 3 change acting as external disturbances as the robot moves. The robot is modeled as follows neglecting friction.

\[
\begin{bmatrix}
H_{x1} & H_{x2} \\
H_{x3} & H_{x4} \\
H_{x5} & H_{x6} \\
H_{x7} & H_{x8}
\end{bmatrix}
\begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2 \\
\dot{q}_3 \\
\dot{q}_4
\end{bmatrix}
+ \begin{bmatrix}
-h_1 \ddot{q}_1 \\
-h_2 \ddot{q}_2 \\
-h_3 \ddot{q}_3 \\
-h_4 \ddot{q}_4
\end{bmatrix}
= \begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2 \\
\dot{q}_3 \\
\dot{q}_4
\end{bmatrix}
\]

(20)

where

- \( H_{x1} = m_1 l_1^2 + I_x + m_2 (l_2^2 + l_1^2) \sin \theta \cos \theta \)
- \( H_{x2} = m_2 l_2 \cos \theta \sin \theta \cos \theta \)
- \( H_{x3} = m_3 l_3 \sin \theta \sin \theta \cos \theta \)
- \( H_{x4} = m_4 l_4 \sin \theta \sin \theta \cos \theta \)
- \( H_{x5} = I_x + l_1^2 m_2 \)
- \( C_{x1} = -m_2 l_2 \cos \theta \sin \theta \sin \theta \cos \theta \)
- \( C_{x2} = -m_3 l_3 \sin \theta \sin \theta \sin \theta \cos \theta \)
- \( C_{x3} = -m_4 l_4 \sin \theta \sin \theta \sin \theta \cos \theta \)
- \( G = mgd^2 + mgd \cos \theta \)

3.2 Experimental Results

The sliding mode control law is given as follows.

\[
\begin{bmatrix}
\tau_x \1 \\
\tau_x \2 \\
\tau_x \3 \\
\tau_x \4
\end{bmatrix} =
\begin{bmatrix}
H_{x1} & H_{x2} & 0 & 0 \\
H_{x3} & H_{x4} & 0 & 0 \\
H_{x5} & H_{x6} & 0 & 0 \\
H_{x7} & H_{x8} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2 \\
\dot{q}_3 \\
\dot{q}_4
\end{bmatrix}
+ \begin{bmatrix}
-h_1 \ddot{q}_1 \\
-h_2 \ddot{q}_2 \\
-h_3 \ddot{q}_3 \\
-h_4 \ddot{q}_4
\end{bmatrix}
\]

(21)

where \( s_x = \dot{q}_1 - \dot{q}_1 + \dot{\lambda} (\theta_x - \theta_x) \)

\[
\begin{bmatrix}
\dot{q}_1 \\
\dot{q}_2 \\
\dot{q}_3 \\
\dot{q}_4
\end{bmatrix} = \begin{bmatrix}
\dot{r}_x \1 \\
\dot{r}_x \2 \\
\dot{r}_x \3 \\
\dot{r}_x \4
\end{bmatrix}
\]

(22)

In the above case the tuning time interval was set as one second so the parameters are updated every second based on the fuzzy rules. For the controller the sampling rate was set as 0.001 second.

For a multi degree-of-freedom system if we simply apply the previous algorithm to each variable separately, a coupling effect may prevail and prevent proper parameter tuning. To solve this problem in this paper both the \( \lambda \) values are taken as the same value and tuned.

During the tracking control, \( \lambda \)’s are updated regularly based on the error and chattering index. At the beginning of a control task, a proper value of \( \lambda \) is not known. Therefore, the control task may start with too large or too small value of \( \lambda \). In the former case, the control bandwidth is high so the tracking error may be very small but the chattering may be significantly high. In the latter case, chattering may not be a problem but the low bandwidth may cause large tracking error. Both of these should be avoided and an appropriate value of \( \lambda \) should be found automatically. In the proposed algorithm, the parameter is supposed to change according to the control performance during the previous tuning period.
Fig. 6 Auto-tuning history for low initial value of $\lambda$ (a) control bandwidth   (b) boundary layer thickness (c) error index   (d) chattering index

Fig. 7 Tracking control result for low initial value of $\lambda$ (a) tracking error before tuning   (b) tracking error after tuning   (c) actuator torque before tuning   (d) actuator torque after tuning

Fig. 8 Auto-tuning history for high initial value of $\lambda$   (a) control bandwidth   (b) boundary layer thickness (c) error index   (d) chattering index
The tuning algorithm was implemented on the robot control and the experimental results are obtained as discussed in the following. The history of tuning of each parameter is shown in Figs. 6 and 8, and the tracking error and actuator force before and after tuning are shown in Figs. 7 and 9.

Fig. 6 shows the auto-tuning results when the initial $\lambda$ is too low. As expected, the initial tracking error was very large and it gradually gets smaller but with increase in the chattering index. However, the chattering is not significant at all and the tracking error was improved very much as can be seen in Fig. 7.

When the initial $\lambda$ was too large, the chattering index was high and it became smaller as the tuning was proceeded as shown in Figs. 8 and 9. The error level remained about the same and the overall control performance was improved.

In both of the above cases, the parameter $\lambda$ was converged to a same value about 65, which tells that the parameter tuning is consistently well done irrespective of the initial condition and the modeling error.

4 Conclusions

In this paper auto-tuning algorithm of sliding mode control parameters using fuzzy logic was suggested and its validity was shown experimentally for a robot control system. In that procedure modeling uncertainties such as resonance mode, time delay, friction and disturbances to the system were given, and the controller actively tuned the control parameters considering the monitored system responses for each case. This is an automation of the selection procedure of the optimal control bandwidth ($\lambda$) and boundary layer thickness ($\phi$) using fuzzy logic for robust control without chattering. It is expected that this algorithm can replace the trial and error method in sliding mode controller design in practice.

References