On Iterative Learning Control for Nonlinear Time-varying Systems with Input Saturation

Ying Tan and Jian-Xin Xu

Abstract—Input saturation is inevitable in many engineering systems. In this note, the focus is to design the proper iterative learning control algorithms when the desired trajectory cannot be realized within the saturation bound. Two new algorithms are proposed. First algorithm introduces a kind of “reference governor” which reduces the amplitude of the desired trajectory systematically such that the modified desired trajectory becomes realizable. The second algorithm employs some “barrier” functions to prevent the control input violating the hard input constraint. Simulation example and discussion are provided with some insights on how the proposed iterative learning algorithms work.

I. INTRODUCTION

This paper considers a special control problem: the control task in a finite time interval \([0, T]\) repeats itself along the iteration domain (see [25] and references herein). This kind of control problem is frequently encountered in many industrial processes such as assembly lines and chemical batch processes. Intuitively, the information obtained from last iteration would be used to improve the control performance of this iteration. Iterative learning control (ILC) is well-known for its ability to improve the control performance without requiring the precise knowledge of systems as it can fully use information that is available due to repeatable control environment.

Since S. Arimoto published his first paper on ILC ([3]), ILC methodologies have become the focus of many researchers as well as engineers over the last three decades (see, for instance, [13], [20], [2], [6], [14], [11], [5], [8], [7], [5], [15], [21], [24], [1] and references therein), leading to numerous practical implementations of ILC schemes, as well as a better theoretical understanding.

In many engineering problems, input saturation is wildly encountered due to a limited size of the actuators, sensors and interfacing devices. The existence of input saturation may severely decrease the control accuracy or cause oscillations, even lead to system instability. Input saturation/input constraints have been addressed in ILC literature, see, for example, [18], [9], [26] with the following standing assumption

The desired trajectory can be realized within the saturation bound.

Under such an assumption, appropriate iterative learning control algorithms are available to deal with input saturation for linear/nonlinear, time-invariant/time-varying dynamic systems. However, this standing assumption does not always hold in engineering applications. In general, engineering practitioners do not know whether the desired trajectory can be realized or not when the dynamic system is partially unknown. When the standing assumption is not true, or is hard to check, the proposed algorithms in literature cannot be applicable directly.

In this paper, two different ILC algorithms are proposed to deal with the input saturation without requiring the standing assumption.

The first ILC algorithm proposed is somehow related to the well-known “reference governors” that have been used in model predictive control when the desired trajectory is not feasible due to the existence of input/state constraints. By planning the “reference trajectory” carefully, the modified reference trajectory becomes feasible for the system with constraints [4]. The basic principle that makes the reference governor to work is that the dynamic model is completely known so that the prediction can be used to check whether the constraints are violated or not. When the dynamic system is not completely known, the model-based prediction cannot be applied.

In this work, a very simple reference governor is employed. Let us assume there exists some positive constant \(\alpha \in (0, 1]\) such that when the desired trajectory \(x_r(t)\) is modified as \(\bar{x}_r(t) = \alpha x_r(t)\), it is realizable within the saturation bound, i.e., the standing assumption is satisfied. Then the ILC algorithms available in literature, see, for example, [26] are applicable. Obviously, such a constant is not unique. Our aim is to find the largest possible \(\alpha\) so that

Y. Tan is with The Department of Electrical and Electronics Engineering, The University of Melbourne, Parkville, VIC 3010. Y. Tan’s e-mail is yingt@unimelb.edu.au

J.-X. Xu is with The Department of Electrical & Computer Engineering, The National University of Singapore, Singapore 117576, and his email is elexujx@nus.edu.sg.
the best performance would be obtained. It is shown that when the decay rate of the parameter $\alpha$ is sufficiently small, it is possible to find an sub-optimal $\alpha$, at the price of slower convergence speed.

The second algorithm introduces a logarithmic “barrier function” as did in [12] in order to avoid violating the hard input constraints. In [12], barrier function is used to obtain optimal algorithms with the consideration of input/state constraints. However, the system considered in [12] is linear time-invariant which is assumed to be known. The only unknown part of the system is the disturbance which is assumed to be independent of the tracking error. Therefore, the method used in [12] cannot be employed in this paper where the problem of interest is nonlinear systems that are not completely known. Therefore, it cannot directly applicable to partially unknown nonlinear dynamic systems.

The second ILC algorithm assumes that there are one ILC updating law (see, for example, $I_1$ in Section 2) that can achieve the perfect tracking performance without input saturation. This updating law optimizes some unknown objective function by computing its gradient with respect to the (unknown) objective function. This is similar to the well-known concept called “inverse optimality” that has been used in [22, Chapter 3].

When there exists hard constraint on input, the barrier function is then added to this “unknown” objective function to avoid violating the input constraint. By computing the gradient of the the barrier function, the ILC updating law can achieve a good performance when the hard input constraints exist.

This paper is organized as follows. In Section 2 we present problem formulation. Main results are stated in Section 3. Simulation example and discussion are provided in Section 4 followed by conclusions.

II. PROBLEM FORMULATION

We denote the set of real numbers as $\mathbb{R}$. The set of integers is denoted as $\mathbb{N}$. $i$ is the number of iteration and $i \in \mathbb{N}$.

For simplicity, the following single-input-single-output (SISO) nonlinear uncertain dynamics is considered

$$\dot{x} = \eta(x, t) + sat(u(t), u^*)$$

where $\eta : \mathbb{R} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a lumped uncertainty, which is globally Lipschitz continuous in $x^1$. That is, there exists a constant $L \geq 0$, independent of $t$, such that

$$|\eta(t, x_1) - \eta(t, x_2)| \leq L|x_1 - x_2|,$$

for any $x_1, x_2 \in \mathbb{R}$ and $t \geq 0$. The saturation function $sat(u(t), u^*)$ is defined as

$$sat(z, u^*) \triangleq sign(z) \min \{u^*, |z|\}, \forall z \in \mathbb{R}$$

where $u^* > 0$ is the upper bound of the control signal, which comes either from a physical process constraint or from an artificial limiter. This work can be extended to multi-input-multi-output (MIMO) system by following the similar way as in [26].

It is assumed that the desired trajectory $x_r(t)$ $t \in [0, T]$ is generated by the following system

$$\dot{x}_r(t) = \eta(x_r(t)) + u_r(t), \quad (3)$$

where $u_r(t)$ is the desired input satisfying

$$\max_{t \in [0, T]} |u_r(t)| \geq u^*$$. \quad (4)

As part of the system repeatability, we assume the following identical initialization condition:

**Assumption 1** $x_i(0) = x_r(0)$.

Let $e_i(t) = x_r(t) - x_i(t)$ and $q > 0$ is the learning gain. The following learning algorithms have been used for the system in the form of (3) in literature:

$$I_1 : u_{i+1}(t) = u_i(t) + q e_i+1(t) \quad (5)$$

$$I_2 : u_{i+1}(t) = sat(u_i(t), u^*) + q e_i+1(t), \quad (6)$$

for all $t \in [0, T]$. $I_1$ can ensure a perfect tracking performance when there is no input constraint while $I_2$ ensures the perfect tracking performance with input saturation under the standing assumption. When condition (4) holds, neither algorithm can work well as can be seen in the following example:

**Example 1**: Consider system (1) with $\eta(x, t) = 3x \sin t$, $t \in [0, 2\pi]$, the target trajectory $x_r = 1.5 \sin^3 t$, and $x_i(0) = x_r(0) = 0$. By computing, $u_r = 4.5 \sin^2 t \cos t - 4.5 \sin^4 t$ and $\max_{t \in [0, 2\pi]} |u_r(t)| = 5.0172$.

Here, we apply $I_1$ and $I_2$ with $q = 5$ and $u^* = 2$, which implies that the standing assumption does not hold. The tracking performance of two algorithms are shown in Figure 1 and Figure 2 respectively, where $\epsilon_{max} = \max_{t \in [0, T]} |\epsilon(t)|$. This example shows when the condition (4) holds, the tracking error of $I_1$ (or $I_2$) are big.

It is apparent, due to condition (4), it is not possible to obtain the perfect tracking performance. Our aim is to find appropriate ILC algorithms such that

- Perfect tracking performance is achieved when the standing assumption is satisfied.
- A reasonably good tracking performance can be obtained when the standing assumption does not hold.
A. ILC design based on the reference governor

Reference governors are frequently used in model predictive control, when the system have constraints on state and control variables. The aim of using the reference governor is to modify, when necessary, the reference (or desired trajectory) in such a way that the constraints are enforced. The reference governor is not a controller, but is installed in an already designed control system.

In this note the purpose is to modify the desired trajectory \( x_r(t) \) so that the modified trajectory \( \bar{x}_r(t) \) satisfies the standing assumption (accuracy). The ILC updating law \( I_2 \) can be applied to ensure that output of the system (1) can track the modified trajectory. Since the system (1) is not completely known (\( \eta(\cdot, \cdot) \) in the dynamics is unknown), how to design a proper reference governor is challenging.

The reason why the desired trajectory \( x_r(t) \) is not realizable under condition (4) lies in the fact that \( x_r(t) \) is too big to be tracked by a limited input. By reducing the size of desired trajectory properly, it is possible to get an modified trajectory that is realizable. Then applying the updating law \( I_2 \) will lead to perfect tracking performance.

As the system (1) is not completely known, the simplest way to design a reference governor is to reduce the size of the desired trajectory, i.e.,

\[
\bar{x}_r(t) = \alpha^* x_r(t), \alpha \in (0, 1),
\]

where \( \alpha^* \) is the solution of the following optimization problem

\[
\max_{\alpha \in \mathbb{R} > 0} \alpha, \quad \text{s.t.} \quad \begin{cases} \dot{x} = \eta(x, t) + u(t) \\ |u| \leq u^*. \end{cases}
\]

The value of \( \alpha^* \) is unknown, but it is known that for any \( \alpha \leq \alpha^* \), the updating law \( I_2 \) can achieve the perfect tracking performance.

Therefore, the following updating laws is used to modify the reference trajectory, learn \( \alpha^* \) and update control input:

\[
I_3 : \begin{cases} x_{r,k} = \alpha_k x_r(t) \\ \alpha_k = \rho^k, \alpha_0 = 1, \rho \in (0, 1), k = Ni, \\ e_{i,k} = x_{r,k} - x_i, \\ u_{i+1}(t) = sat(u_i(t), u^*) + q\bar{e}_{i+1,k} 
\end{cases}
\]

where \( N \) is called the dwell time that to ensure that the updating law \( I_2 \) can converge provided that \( \alpha \leq \alpha^* \). The updating law \( I_3 \) will be stopped when \( \epsilon_{\text{max}} = \max_{\epsilon(t) \in [0, T]} |\dot{e}(t)| \) is smaller than a pre-defined threshold \( \epsilon \).

There exists a clear time-scale separation, as the updating of \( \alpha_k \) is much slower than that of the control input. Once \( \alpha_k \) is updated, it will wait \( N \) iterations to see whether the new reference \( x_{r,k} \) can be realized within the saturation bound.

This result can be summarized as follows

\textbf{Result 1:} For any \( \nu > 0 \) such that there exist \( \rho^* \in (0, 1), \epsilon > 0, N^* \) such that for any \( \rho > \rho^* \) and \( N \geq N^* \), the proposed \( I_3 \) can ensure that

\[
\limsup_{k \to \infty} |\alpha_k - \alpha^*| \leq \nu \quad \limsup_{i,k \to \infty} |e_{i,k}| \leq \epsilon.
\]

Obviously, \( \rho^* \) needs to be sufficiently close to 1 and \( N^* \) has to be sufficiently large. The proposed ILC algorithm \( I_3 \) can converge to \( \nu \)-neighborhood of \( \alpha^* \). There is a trade-off between the convergence speed and the size of \( \nu \) (accuracy). How to design proper \( \rho \) and \( N \) relies on the particular
performance requirement. Simulation results in Section VI will illustrate the trade-off between convergence speed and accuracy.

Remark 1: The proposed ILC algorithm \( I_3 \) uses an online tuning of the reference governor by checking the tracking performance after \( N^{th} \) iteration. Since the model of the system is not completely known, checking the tracking performance is the only way to evaluate whether the current \( \alpha_k \) can work.

Remark 2: The reference governor used is very simple. It just reduces the size of the desired trajectory while the shape of the desired trajectory does not change. Although it is very easy to be implemented, it may not be the best choice for some applications. For example, there may exist another realizable reference \( x_{r,o}(t) \) such that
\[
\max_{t \in [0,T]} |x_r(t)| \leq u^* \\
\int_0^T |x_r(\tau) - x_{r,o}(\tau)|^2 d\tau \leq \int_0^T |x_r(\tau) - \alpha^* x_r(\tau)|^2 d\tau.
\]
That is this realizable \( x_{r,o}(t) \) can achieve a better tracking performance in some sense. However, since the system is not known to designers, it is difficult to construct \( x_{r,o}(t) \) systematically. If more knowledge of the system is available, it may be possible to find such \( x_{r,o}(t) \). How to design a proper reference governor in order to find \( x_{r,o} \) using additional knowledge of the system will be our future work.

B. ILC design based on introducing the barrier functions

Barrier functions are widely used in nonlinear programming problem:
\[
\min J(x), \quad \text{subject to} \quad c(x) \leq 0, \tag{13}
\]
where \( J : \mathbb{R}^n \to \mathbb{R} \) and \( c : \mathbb{R}^n \to \mathbb{R}^m \) are both smooth. By introducing some type of barrier function \( \psi(\cdot) \), the constrained minimization in (13) will be approximated by an unconstrained optimization involving an interior-point barrier function for the hard constraints,
\[
J_a(x; \mu) = J(x) + \mu \sum_{j=1}^m \Psi(c_j(x)), \tag{14}
\]
where the barrier function \( \Psi : (-\infty, 0) \to \mathbb{R} \) is chosen to be convex, nondecreasing, and \( C^1 \) on the open domain \((-\infty, 0)\). Furthermore, it must satisfy
\[
\lim_{z \to 0^-} \frac{\partial \Psi}{\partial z} = \infty. \tag{15}
\]
Assume that \( x^* \) is a local solution of (13), then the objective function \( J_a(x; \mu) \) has a local minimizer near \( x^* \) for all \( \mu \) sufficiently small. In many cases \( \Psi(z) = -\ln(-z) \) or \( \Psi(z) = -\frac{1}{z} \) are used [19].

In this note, the hard constraint is
\[
u^2 - (u^*)^2 \leq 0. \tag{16}
\]
However, the objective function \( J(u)/J(e) \) is unknown, though it is known that when \( I_1 \) is applied to the system (1), the desired performance (perfect tracking) can be achieved without input saturation.

Here we assume there exists an objective function \( J(u) \) corresponding to \( I_1 \) such that \( \frac{\partial J}{\partial u} = qe(t) \) so that the updating law \( I_1 \) can achieve the optimal performance. With the consideration of the input saturation, the “new” objective function becomes
\[
J_a(u; \mu) = J(u) + \mu \Psi(u) \tag{17}
\]
\[
\Psi(u) = -\ln \left( -\left( u^2 - (u^*)^2 \right) \right). \tag{18}
\]
Using a technique that widely used in nonlinear programming in updating \( \mu \), it yields the following updating law
\[
I_4 : u_{i+1}(t) = u_i(t) + qe_{i+1} + \mu \frac{\partial \Psi}{\partial u}(u_i(t)). \tag{19}
\]
\[
\mu_{i+1} = \rho \mu_i, \tag{20}
\]
where \( \rho \in (0, 1) \).

It should be noted that the proposed \( I_4 \) works well under some conditions of \( J(u) \). For example, one assumption is that \( J(u) \) needs to be twice continuously differentiable for all \( |u| \leq u^* \) in order to ensure that the unconstrained optimization problem (14) well approximates the constrained optimization problem (13). However, \( J(u) \) is not known, it is thus hard to verify that all needed assumptions hold (for assumptions required, see [19] for more details). Nevertheless, even though we cannot ensure that \( I_4 \) is an optimal solution, it is at least a feasible solution, or sub-optimal solution in most cases. In Section IV, we will discuss the performance of \( I_4 \).

IV. SIMULATION EXAMPLE AND DISCUSSION

In this section, Example 1 is still used to illustrate the effectiveness of the proposed two ILC algorithms. The first part of this section is to show the trade-off between the convergence speed and the accuracy. The second part of this section compares four iterative learning algorithms.

A. The performance of \( I_4 \) with different choice of parameters

In this example, we will how the tuning parameters \( \rho, N \) and \( \epsilon \) affect the performance of ILC algorithm \( I_3 \). Three groups of parameters are chosen

Case I : \( \rho = 0.99999, N = 20, \epsilon = 0.01 \)

Case II : \( \rho = 0.99, N = 10, \epsilon = 0.05 \)

Case III : \( \rho = 0.9, N = 5, \epsilon = 0.1 \).
In simulation, the target trajectory is chosen \( x_r = 1.5 \sin^3 t \), and \( x_i(0) = x_r(0) = 0 \) and \( \max_{t \in [0, 2\pi]} |u_r(t)| = 5.0172 \).

Here, we apply \( I_3 \) with \( q = 5 \) and \( u^* = 2 \). The tracking performance of three different cases can be found in Figures 3-5.

Figure 3 shows three modified reference trajectories generated by \( I_3 \). Among them \( X_{r,I} \) is the most close to the desired trajectory \( x_r(t) \). Figure 4 shows that output of the system after iterative processes stop. For Case I, it takes 16780 iterations to converge while for Case II, it takes 880 iterations. But for Case III, only after 220 iterations, the pre-defined threshold is achieved. This example clearly demonstrates the trade-off between the convergence speed and the precision (\( \nu \)-neighborhood of the optimal \( \alpha^* \)). Figure 5 shows control inputs for two different cases (Case I and Case II). The difference between the control input of the Case II and that of the Case III are negligible and could not be seen clearly in the figure. Figure 5 shows that the control input generated in Case I (the slowest convergence speed and the best accuracy) is more oscillatory.

\[ \text{Fig. 3. The modified reference signals of } I_3 \]

B. Comparison of four different ILC algorithms

In order to evaluate the performance of the proposed two ILC algorithms, they are compared with two algorithms in literature (\( I_1 \) and \( I_2 \)). The Example 1 is again used with different desired trajectories and different saturation bound.

For three different trajectories, condition (a) indicates that the standing assumption is not satisfied. On the other hand

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{Case} & x_r & \max_{t \in [0, 2\pi]} |u_r(t)| & a (u_{max}) & b (u_{max}) \\
\hline
A & 1.5\sin^3(t) & 5.0172 & 6 & 2.5 \\
B & 5 \sin(0.5t) & 10.1341 & 11 & 5 \\
C & 0.8t^2 & 65.9561 & 66 & 33 \\
\hline
\end{array}
\]

Condition (b) means that the standing condition is not satisfied.

The tracking error \( e_{I,i} \), computed from four different ILC algorithms under different cases can be seen in the following table. Here the tracking error is evaluated as \( e_{I,i} = \frac{1}{2\pi} \int_0^{2\pi} e_i(t)^2 dt \).

<table>
<thead>
<tr>
<th>Case</th>
<th>( I_1 )</th>
<th>( I_2 )</th>
<th>( I_3 )</th>
<th>( I_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_a )</td>
<td>0.17</td>
<td>4.3 \times 10^{-4}</td>
<td>2.0 \times 10^{-4}</td>
<td>1.1 \times 10^{-6}</td>
</tr>
<tr>
<td>( A_b )</td>
<td>169.64</td>
<td>74.84</td>
<td>0.1521</td>
<td>74.32</td>
</tr>
<tr>
<td>( B_a )</td>
<td>11.26</td>
<td>5.7 \times 10^{-3}</td>
<td>1.6 \times 10^{-3}</td>
<td>6.5 \times 10^{-6}</td>
</tr>
<tr>
<td>( B_b )</td>
<td>1559.8</td>
<td>580.44</td>
<td>2.98</td>
<td>580.22</td>
</tr>
<tr>
<td>( C_a )</td>
<td>7.05</td>
<td>2.8 \times 10^{-3}</td>
<td>3.23 \times 10^{-4}</td>
<td>1.73 \times 10^{-4}</td>
</tr>
<tr>
<td>( C_b )</td>
<td>27.82</td>
<td>26.02</td>
<td>48.92</td>
<td>25.98</td>
</tr>
</tbody>
</table>

Table 1. The tracking performance of 4 different ILC algorithms.
For $I_1$, $I_2$ and $I_4$, the tracking error is evaluated at the 100th iteration (fixed the iteration number). For $I_3$, the number of iteration depends on the choice of the parameters $(\epsilon, \rho N)$. In this example, $\rho = 0.99$, $N = 10$ and $\epsilon = 0.1$. The number of iteration used in 6 different cases is shown in the following table.

<table>
<thead>
<tr>
<th>Number of iterations</th>
<th>$A_a$</th>
<th>$A_b$</th>
<th>$B_a$</th>
<th>$B_b$</th>
<th>$C_a$</th>
<th>$C_b$</th>
</tr>
</thead>
</table>

Table 2. Iteration number needed for $I_3$ under 6 cases.

When the standing assumption is satisfied ($A_a$, $B_a$ and $C_a$), three ILC algorithms $I_2$-$I_4$ work well, especially, $I_4$ converges much faster than $I_2$ and $I_3$. For the algorithm $I_1$, as the input saturation is ignored in the ILC design, even though the standing assumption is satisfied, the tracking performance is not good enough.

When the standing assumption is not satisfied, the performance of $I_2$ and $I_4$ are very similar in all three cases, although $I_4$ is a bit better. $I_3$ works well in Case $A_b$ and Case $B_b$, but it does not work well in Case $C_b$. On the other hand, $I_4$ works pretty well in Case $C_b$, but not good in Case $A_b$ and Case $B_b$.

It is observed that, in both $I_2$ and $I_4$, it is hard to escape from the saturation bound once it enters the saturation bound. This may explain why when the desired trajectory monotonically increases in time, $I_2$ and $I_4$ works well (see, for example Case $B_b$). But when the desired trajectory is not monotonic, the performance of both $I_4$ and $I_2$ are not good enough.

On the other hand, $I_3$ can escape from the saturation bound easily. It is very suitable to deal with the cases when the desired trajectory does not monotonically increase in time in the simulation.

Since the simulation is limited to one example, it just can provide some insight in using the proposed methods. Further effort is need to evaluate the performance of the proposed ILC algorithms more rigorously.

V. CONCLUSIONS

We have presented two iterative learning control algorithms that can work well when the desired trajectory cannot be realizable due to input saturation/constraints.

REFERENCES


