Abstract—Time optimal output transition for input-constrained systems is a frequent requirement in many actuation systems. In the case of known and/or linear systems, techniques exist to determine the corresponding control input either analytically and/or experimentally. In other cases however, is a sufficiently precise model is available. Against this background this paper proposes a learning algorithmic approach to solve the issue for unknown nonlinear saturated systems. The proposed approach searches the fastest transition by setting a time optimal but unfeasible trajectory and using learning techniques to recover the nearest feasible one in some sense. To achieve this goal, the proposed approach performs twice a learning process. On one side, it determines by iterative learning control for each suggested trajectory either the optimal input if the trajectory is feasible or an error information if this is not the case. On the other side, a model free learning model generator is used to determine the best feasible reference trajectory.

I. INTRODUCTION

This paper is concerned with the time-optimal output transition of an unknown nonlinear plant with saturated inputs. The problem behind this interest occurs frequently in industrial applications, especially in drives: the system is complex, mostly nonlinear, saturations may occur at different places, and minimum transition time is required. Examples of this problem are for instance compressor valves as discussed in a former paper [1].

In general, the transition problem we are addressing consists in going from a point A to a point B in time optimal manner as hinted in Fig. 1.

In principle, the problem as stated is unsolvable, as the computation of a time optimal trajectory requires a model. To solve this conflict, there are essentially two possible paths.

First, the model can be derived by identification, a possibly difficult task in a nonlinear setup, or the alternative, it is possible to determine experimentally the correct trajectory exploiting a suitable combination of iterative learning control and reference governor, these two can be designed to be model free.

Iterative learning control is a control method designed to continuously improve the tracking performance in the case of repetitive operation by using the information on the tracking error of a previous run for the next one [2], [3] and [4]. In the last years different approaches have been developed which allow including some optimality conditions, but it is essentially a tracking and not an optimization setup. In its general case, ILC does not require a model, and, if it converges, it leads to the correct feed-forward signal. ILC offers more advantages; in particular it is easily implementable and represents an intuitive solution for control problems by imitating the human learning process. The use of the ILC relies on a few basic assumptions, which a system has to fulfill [3], and some of which have already been relaxed (e.g. same starting point for each iteration [5]).

The ILC is of particular interest when no model is available and only a specific trajectory is given, in particular for nonlinear systems.

One of the important conditions for convergence of ILC is the feasibility of the desired trajectory. Applying the ILC to an unknown system, which is the main advantage of the ILC, it is not possible to guarantee that the trajectory is feasible. However, this task can be taken over by reference governors, as they have been proposed in [6], [7], [8] to determine feasible trajectories, albeit for known systems.

A similar framework has been already discussed in several other papers. One possible approach for the output transition problem is the Terminal Iterative Learning Control (TILC) [9]. This kind of ILC uses the error information of a single point (e.g. the error at $t_1^*$ in Fig. 1) in combination with the impulse response of the system to learn an input sequence to reach the desired point at the specified instant of time. An alternative to use the measurements to calculate the error is to use a model description of the system and to estimate the error [10]. However all versions of the TILC need a model and are not suited for unknown system control tasks.

Another possibility is offered by the Norm Optimal ILC (NOILC) [11], [12], but in spite of its name this ILC is only optimal in the sense of the error evolution. This kind of ILC is based on an analytical general result of error norm and uses system information to calculate the next input sequence to the system. An extension of these algorithms

---

H. Trogmann and L. del Re are with the Institute for Design and Control of Mechatronical Systems, Johannes Kepler University Linz, 4040 Linz, Austria hannes.trogmann@jku.at
for time optimal control is not straightforward, because the optimization occurs in the iteration domain, while the time optimal optimization - as its name says - happens in the time domain.

There have been other interesting contributions related to linear systems with saturated inputs, see e.g. [13], albeit again based on system information. A model free algorithmic approach for the same task as we intend to solve has been proposed by [14], albeit explicitly relying on linearity.

In this work, we propose an algorithmic approach which essentially relies on decoupling the two problems - determining the ideal trajectory to obtain the time optimal solution and tracking it. The connection point is given by the convergence of the ILC: we start with an infeasible but definitely time optimal trajectory, and use the convergence properties of the ILC to detect the nearest feasible approximation. Where feasible means that there exist an input \( u^* \) for the required output transition for which the error becomes zero. The convergence of the whole setup is not critical, as ILC and reference governor iterate in different domains. Of course, the whole system could be described and treated as a single hybrid system, but in this paper we stick to the sequential structure for the inner loop of the presented control method and learn properties for some cases.

This paper is organized as follows: the basic ILC and the corresponding convergence analysis are presented in Section II, followed by trajectory update method in Section III and the complete control scheme in Section IV. The last two sections cover the results of a simulation example, the conclusions as well as an outlook on future work.

II. Iterative Learning Control

As stated above, we intend to use ILC not only to track the signal, but also to detect unfeasibility of the requirement, so we shall recall some basic results of ILC for nonlinear, input affine systems. To this end, consider the following nonlinear, input affine system,

\[
\begin{align*}
\dot{x} &= f(x) + g(x) u \\
y &= h(x)
\end{align*}
\]

(1)

with \( f : \mathbb{R}^n \rightarrow \mathbb{R}^n, B \in \mathbb{R}^{n \times m}, h : \mathbb{R}^n \rightarrow \mathbb{R}^p \) and \( x \in \mathbb{R}^n, u \in \mathbb{R}^m \) and \( y \in \mathbb{R}^p \). In the literature [2], [3] a PD-type ILC has been shown to offer good convergence properties for many applications similar to our setup:

\[
v = (1 - \rho) u_k + \rho u_0 + \gamma e_k + \theta \dot{e}_k
\]

(2)

\[
u_{k+1} = \text{sat}(v, u_{\text{max}}) \cdot \text{sign}(v)
\]

with \( \rho \in [0, 1] \) as a forgetting factor, \( u_{\text{max}} \) the maximal input value, \( \gamma \) proportional and \( \theta \) derivative gains, \( e_k = y_d - y_k \) and \( u_0 \) the initial value for \( u_k \). We shall use this ILC structure for the inner loop of the presented control method and learn \( v \) without using any information on the system.

For the following discussion, we need to ensure that the controlled system fulfills some conditions which are fulfilled by a large number of real systems, in particular

- Lipschitz (at least locally)
- the input is bounded
- minimum phase.

We shall define the bound on \( u(t) \) by

\[
|u(t)| \leq B_B.
\]

(3)

This ensures that it is not necessary or possible that the input will be \( \infty \) (a higher input that the maximal has no additional effect to the system). The Lipschitz condition is fundamental for the use of an ILC

\[
\begin{align*}
\|f(x) - f(y)\| &\leq k_f \|x - y\| \\
\|g(x) - g(y)\| &\leq k_g \|x - y\| \\
\|h(x) - h(y)\| &\leq k_h \|x - y\|
\end{align*}
\]

(4)

with \( k_f, k_g \) and \( k_h \) the Lipschitz constants. Using the Lipschitz constants - just bounds on the rates of change - usually yields more conservative choices of \( \gamma \) and \( \theta \) and thus a slower convergence rate than a model based ILC could do. Minimum phase is also a condition assumed in the design of the ILC update law in (2). Again, this condition is not always fulfilled, but in the case of fast actuators the mechanical design is tailored to fast movements and non-minimum phase behavior would be a poor design choice. Additional requirements, like realizable and piecewise differentiable, for desired output \( y_d \) of the systems have to be fulfilled too.

A. Convergence analysis

The complete convergence analysis for a PD-Type can be found in [3], but the main results for us are that the convergence speed mainly depends on the Lipschitz constants and the selected values for \( \gamma \) and \( \theta \) in (2). To this end, we start with the expression for the errors needed in the update law (2),

\[
e_k = y_d - y_k = h(x_d) - h(x_k) = \delta h(x_k)
\]

(5)

and its derivative

\[
\begin{align*}
\dot{e}_k &= \dot{y}_d - \dot{y}_k = h(x_d) - h(x_k) \\
& = \frac{\partial h(x_d)}{\partial x_d} \dot{x}_d - \frac{\partial h(x_k)}{\partial x_k} \dot{x}_k + \frac{dh(x_d)}{dt} - \frac{dh(x_k)}{dt} \\
& = \delta h_{x_k} \dot{x}_d + h_k(x_k) \delta \dot{x}_k + \delta h_k
\end{align*}
\]

(6)

with \( \delta h_{x_k} = h_k(x_d) - h_k(x_k), \quad \delta h_k = h_k(x_d) - h_k(x_k) \) and \( \delta \dot{x}_k = \dot{x}_d - \dot{x}_k = f(x_d) - f(x_k) + B_d u_d - B_d u_k \). Differently to the classical convergence analysis we shall not be concerned with showing that the error is monotonically decreasing, but that the difference between the applied input \( u_k \) and the "desired" input \( u_d \) goes to zero. For this it is necessary to calculate the input difference of the consecutive iteration,
\[
\delta u_{k+1} = u_d - u_k = u_d - (1 - \rho) u_k - \rho u_0 - \gamma e_k - \theta \dot{e}_k \quad (7)
\]

by inserting (5) and (6) into (7) and some calculus, the difference becomes

\[
\delta u_{k+1} = (1 - \theta h_k(x_k) B_k) \delta u_k + \rho \delta u_0 - \gamma \delta h(x_k) - \theta [\delta h_k (f(x_d) + B_d u_d) + h_k (\delta f_k + B_k u_d) + \delta h_k]
\]

(8)

with \( \delta f_k = f(x_d) - f(x_k), \delta B_k = B_d - B_k, \delta u_0 = u_k - u_0. \)

By taking the \( \lambda \)-norm of (8) in combination with the assumption that the system is Lipschitz (for each \( \delta \) function a Lipschitz constant is given) and the bounded input, the relation between actual and consecutive input difference become,

\[
\| \delta u_{k+1} \|_{\lambda} \leq \mu \| \delta u_k \|_{\lambda} + a_0 \| \delta x_k \|_{\lambda} + \rho \| \delta u_0 \|_{\lambda}
\]

(9)

where

\[
\mu = (1 - \theta h_k(x_k) B_k)
\]

\[
a_0 = B_y k_{hx} + B_\theta (k_{hx} B_{sd} + k_{hr} B_{hx} (k_{hx} + k_{y1} B_{sd}))
\]

The remaining disturbing part in (9) is \( \delta x_k, \) for which it is possible to get an assessment by calculating the result analytically and afterwards take the \( \lambda \)-norm of the result (including the Lipschitz terms),

\[
\| \delta x_k \|_{\lambda} \leq \| x_k (0) \| + B_y O (\lambda^{-1}) \| \delta u_k \|_{\lambda} / (1 - a_2 O (\lambda^{-1})}
\]

(10)

with \( O (\lambda^{-1}) \) as terms depending on \( \lambda \) and \( a_2 = k_f + k_{y1} B_{sd}. \) Finally it is possible to prove the convergence of the ILC. The convergence proof is based on the difference between the desired input sequence and the learned input sequence and is

\[
\| \delta u_{k+1} \|_{\lambda} \leq \bar{\mu}_k \| \delta u_k \|_{\lambda} + \bar{\varepsilon}
\]

(11)

with the Lipschitz and bound constant depending factors

\[
\bar{\mu}_k = \mu_k + B_y O (\lambda^{-1}) / (1 - a_2 O (\lambda^{-1})}
\]

\[
\bar{\varepsilon} = \rho \| \delta u_0 \|_{\lambda} + a_0 \| x_k (0) \| / (1 - a_2 O (\lambda^{-1})}
\]

\[
a_0 = B_y k_{hx} + B_\theta (k_{hx} B_{sd} + k_{hr} B_{hx} (k_{hx} + k_{y1} B_{sd}))
\]

\[
a_2 = k_f + k_{y1} B_{sd}. \)

Again, the convergence can be ensured by choosing the adequate values of the update law (2) only on the bases of the assumed limits without using any system knowledge.

**B. Effect of an unfeasible trajectory**

As it can be seen in (2) the ILC depends via the error \( e_k \) on the desired output trajectory, and the ILC output will increase or decrease at each time point until the corresponding error becomes zero. With the choice highlighted above, it can be ensured that the consecutive error diminish and converge to a "perfect" tracking of the desired trajectory. However, if the trajectory is unfeasible for some points the consecutive error will not decrease as the output of the update law (2) is saturated and the convergence result

\[
\| e_{k+1} \| \leq q \| e_k \| \quad 0 < q < 1
\]

will not hold. If the system is unknown, and thus also the saturation values, the failure of convergence of the error can be used to detect a (local) lack of feasibility.

**III. Trajectory update**

**A. Algorithmic presentation**

The key idea of this work is to start with the fastest (and as consequence time-optimal) trajectory, the step (the solid line in 1). However, this trajectory will be unfeasible for most cases at least at some points \( t' \), and the reference governor is expected to recover feasibility by reducing the requirements at the time \( t' \). In terms of the error dynamics, this means enforcing the convergence of the error \( e_k(t') \) not (only) by increasing the output, but (also) by reducing the reference output.

We shall first present the algorithm and then make some remarks on the two convergence problems involved.

The method is based on Theorem 1 in [2] and on the similar Theorem 2 in [15]. Both theorems show that feasibility is required for the convergence of the trajectory. Only then the ILC will be able to learn an input sequence to achieve

\[
\lim_{k \to \infty} \| e_k \| = 0;
\]

(13)

Conversely, if the trajectory or parts of the trajectory are unfeasible, the error in this time points will not decrease for further iterations.

```c
if modulo(t, Tu)==0
    e_new = actual error;
    for i=1:length(e_new)
        change = f/(abs(e_new(i)-e_old(i))); cor = abs(e_new(i)+change);
        val(i) = (1-cor/2)*old_value(i)+cor/2+y(i);
    end
    old_value = val;
e_old = e_new;
end
```

Listing 1. Trajectory update algorithm

The basis of the algorithm is the detection of the unfeasible trajectory points, this has been solved by using (12) and transforming it into a barrier-function (line 4) with a factor \( f \) as tuning parameter. The parameter can be used to create a very sharp update condition, so that an update is only done for points which have a very low (near to 0) error gradient. All variables in the algorithm are in vector form containing
the values corresponding to the single time points over a complete iteration of the ILC. Due to the use of a barrier-function the trajectory is adapted if

\[ \Delta e_i(t) \leq \varepsilon \quad 0 \leq t \leq T \]  

(14)

To prevent that the algorithm changes the trajectory when the error is already zero, the correction term is multiplied with the actual error (line 5) and will be zero if the actual error is zero. One critical issue remains: if the error is already zero over an update iteration, \( \Delta e \) would be zero too and the barrier-function returns an infinite value, as consequence the update factor for this point would not be defined. In this particular case, the update factor is set to zero.

\[ \bar{y}_d, i = 1 \]

\[ \bar{y}_d, i > 1 \]

\[ \text{ILC} + \Sigma \]

\[ \text{Update Trajectory} \]

\[ e \]

![Fig. 2. Idea of the update procedure](image)

In Fig. 2 the basic scheme of the algorithm and update procedure is shown. During the first update iteration the ILC receives the step as the desired output trajectory. In some parts of the trajectory the error will decrease, in some other the error will remain constant or be higher after the iterations due to the saturations. For the areas where \( \Delta e \neq 0 \) is true nor the condition \(|\Delta e| \leq \varepsilon \) is fulfilled, the update block uses the algorithm presented to lower the specific trajectory points and create a new desired trajectory for the ILC. This loop is active as long as the residual error exists without a change in the error.

As soon as the reference becomes feasible (dash dotted line in Fig. 1) the ILC is able to invert the system and to track the reference. So the trajectory is not designed a priori but will be generated by an additional loop of the control method out of the initial reference. This allows minimizing the transition time. Differently available reference governors, the feasible set is not known a priori.

B. Convergence analysis of the update loop

There are two different convergence questions to be considered in the case of the update loop: convergence to a feasible solution and convergence to the time optimal solution.

1) Convergence to a feasible solution: We first assume that the inner loop converges for the feasible points, i.e. given a feasible trajectory \( y_{d,i}(t) \)

\[ \lim_{k \to \infty} e_i(t) = y_{d,i}(t) - y_k(t) = 0 \]  

(15)

whit \( y_k(t) \) the system output at the iteration \( k \) and \( y_{d,i}(t) \) the actual trajectory generated by the reference governor. Let us assume now that the actual reference \( y_{d,i} \) produced by the reference generator is not feasible and thus leads via the ILC update law (2) to a saturation at the time \( t_s \) of the system described by (1). Then the system will locally perform as

\[ x(t_s) = f(x(t_s)) + g(x(t_s)) \text{sat}(u(t_s)) \]

\[ y = h(x(t_s)) \]

which means that a change at the input will not change the output any-further. The same can occur with the states, the system input will still change but have no effect on the states (or a part of them). Let us assume that the saturation leads to an error, i.e. \( \lim_{k \to \infty} e_i(t_s) = y_{d,i}(t_s) - y_k(t_s) \neq 0 \) but \( \lim_{k \to \infty} e_i(t_s) = y_{d,i}(t_s) - y_k(t_s) = 0 \). Looking at (2) and the following equations it is easy to see that under quite mild conditions if the error cannot be reduced by changing the output, it can be reduced by changing the reference. In other words, reducing the value of \( y_{d,i}(t_s) \) (in the case of a positive error) leads to re-establish the convergence conditions and thus to establish a feasible trajectory.

2) Convergence to the optimal solution: In general, as no information on the system is assumed besides minimum phase and the Lipschitz conditions mentioned above, a proof of this convergence will not be possible. However, it is easy to see that for instance for linear systems with positive Hankel coefficients the solution will converge to the time optimal solution, and similar considerations are valid also for the equivalent nonlinear systems.

IV. COMBINING THE LOOPS

![Fig. 3. Control scheme of the combined loops](image)

- Block scheme of the complete process
- Conditions to combine the both loops
- possible drawbacks

The combination of both loops is a critical step in the method as both loops try to reduce the tracking error, the inner one by changing the control input to the system, the outer one by changing the desired trajectory. However, it is easy and sensible to design the operation of the two systems in two different scales of time, the outer loop waiting for the inner one to converge (or stop doing it). The number of iterations of the inner loop required before the outer loop changes the reference will depend on the tuning of the two controls, but a simultaneous tuning has not been considered here, also in view of the generality of the application class.
V. Example

Most applications of ILC concern positioning problems of highly dynamic mechanical systems. As an example we use a second order mechanical system to demonstrate the possibility to reduce the necessary time of the output transition from A to B. For simplicity a linear model of an elevator has been used, but the method does not depend in any way on linearity.

\[
\begin{align*}
\dot{x} &= \begin{bmatrix} 0 & 1 \\ -25 & -10 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} F \\
y &= \begin{bmatrix} 1 & 0 \end{bmatrix} x,
\end{align*}
\]

with \(F\) as the input and the position of the system as output. To make it more realistic the input is limited \(|F| \leq 500\) N.

As controller the PD-type ILC in Section II has been used and \(\gamma\) was set to 0.4 and \(\theta\) to 50 respectively. For the design process no information about the system has been used, the important is that the input is bounded and the system is Lipschitz with known values. The ILC is able to control the system and to achieve a good result during the first 2000 iteration, due to the remaining error, caused mainly by the unfeasible trajectory, the ILC starts to diverge and the result becomes afterwards poor (Fig. 4 in the upper graph). To evaluate the tracking and control quality of the ILC with and without the trajectory update, two error measures

\[
\|e_k\|_2 = \sqrt{e_k^T e_k}
\]

and

\[
\sup |e_k| = \max |e_k(t)| \quad 0 \leq t \leq T
\]

with \(k\) as iteration index were introduced. In Table I the resulting error of the two methods is compared, the change in the error being the consequence of the update law. The first case, ILC without external loop, is depicted in Fig. 4. The error development seems to suggest that the ILC works well and has converged to the desired trajectory. However, a closer look to the corresponding input sequence shows that the result is not optimal and the trajectory is only approximately the desired one.

Using now the reference update loop, we obtain the results shown in Fig. 4. The settings of the ILC are the same as before, the differences can clearly be seen in the error development and input sequence, which are caused by the trajectory update. As it can be seen by comparing Fig. 4 and Fig. 5 the error is now reduced and the transition time has decreased from 1.28s (without adaption) to 1.13s (with adaptation, and nearly the optimal time compared with a bang bang controller 1.12s). From the form of the update it is clear that the error becomes lower after the update, but what
more important is the influence on the ILC which decreases over the iterations after the update has occurred. As it can be seen in Fig. 6 the update provokes a small step in the error development and the ILC changes the input sequence due to new system information to lower the error of the system. Based on the influence in the error development by the update method it is possible to detect when a real update of the trajectory has occurred. In Fig. 6 this effect can be seen.

![Error Development](image)

**Fig. 6.** Effect of the trajectory update - The ILC gets new information and is able to reduce the error in further iterations.

**VI. CONCLUSION AND OUTLOOK**

In this paper a method has been presented which allows using an ILC to determine a time optimal output transition of an unknown nonlinear system. In combination with the proposed trajectory update algorithm no a feasible trajectory for the ILC is necessary at the beginning, it is even not necessary to have a set of feasible trajectories. Due to this combination of ILC and trajectory update it is possible to create a control system able to realize time and energy optimal trajectories for unknown systems. The future research in this field will consist of improving the update algorithm, by using a varying update time. The problem is shown in Fig. 6, the first trajectory updates would not be necessary, because the error is still decreasing, afterwards the update should occur more often due to a higher convergence rate of the ILC. Another interesting aspect is to use optimization methods for the trajectory update procedure.

**VII. ACKNOWLEDGMENTS**

The authors gratefully acknowledge the sponsoring of this work by the JKU HOERBIGER Research Institute for Smart Actuators (JHI).

**REFERENCES**


