Proceedings of Machine Learning Research vol 120:1–10, 2020

Constraint Management for Batch Processes Using Iterative Learning Control and Reference Governors

Aidan Laracy AIDAN.LARACY@UVM.EDU
College of Engineering and Mathematical Sciences, University of Vermont, Burlington, VT USA

Hamid Ossareh HAMID.OSSAREH@UVM.EDU
College of Engineering and Mathematical Sciences, University of Vermont, Burlington, VT USA

Abstract

This paper provides a novel combination of Reference Governors (RG) and Iterative Learning Control (ILC) to address the issue of simultaneous learning and constraint management in systems that perform a task repeatedly. The proposed control strategy leverages the measured output from the previous iterations to improve tracking, while guaranteeing constraint satisfaction during the learning process. To achieve this, the plant is modeled by a linear system with uncertainties. An RG solution based on a robust Maximal Admissable Set (MAS) is proposed that endows the ILC algorithm with constraint management capabilities. An update law on the MAS is proposed to further improve performance.

Keywords: Iterative Learning Control, Reference Governor, Constraint Management, Batch Processes, Maximal Admissible Sets

1. Introduction

Initially proposed in Arimoto et al. (1984), Iterative Learning Control (ILC) is a method of control used for systems that perform the same task repeatedly. Similar to how humans learn from previous experiences, ILC controllers use information from previous iterations, or batches, of the task to improve tracking performance. Its applications have been explored in high speed trains Yu et al. (2018), hard disk drives J. Xu et al. (2001), robotics Marchal et al. (2014), and numerous other systems performing a repetitive task.

One of the major challenges with ILC is enforcing input, output, or state constraints. To provide examples of such constraints, consider a robotic arm on an assembly line. Typical constraints for this example are actuator saturation, position and angle constraints, and constraints on power consumption. Clearly, constraint violation could lead to the injury of factory workers, damaged machinery in the surrounding area, or damaged components of the arm.

Several schemes have been proposed in the literature to handle constraint management of systems controlled by ILC. In Zhang et al. (2016), a data-driven ILC scheme is used for systems with unknown models that have input, output, and rate of change of input constraints. A quadratic program is used to optimize the control signal, and input-output data is used to estimate system matrices as the ILC learns. Sebastian et al. (2018) uses ILC for linear time-varying systems with input and output constraints, where an output feedback loop based on barrier functions is used for output constraint management. In Ruikun and Ronghu (2017), input saturation is considered for nonlinear MIMO systems. To do this, a P-type ILC is used containing a saturated control term, a feedback term, and a system uncertainty estimate term. Jin et al. (2014) uses a convex optimization-based ILC for iteration-varying systems with output constraints. This is done by defining a cost function to optimize the learning term of the ILC algorithm. The above papers either do not consider output or state constraints, or use nonlinear or quadratic programming to update the control signal.

In this paper, we propose an alternative solution for constraint management of systems controlled by ILC. The solution is based on the Reference Governor (RG) algorithm, and is motivated by the fact that the traditional RG for linear systems, as described in Gilbert and Kolmanovsky (1995b); Kolmanovsky et al. (2014); Osorio and Ossareh (2018); Liu et al. (2018); Ossareh (2019); Li et al. (2019), can naturally handle input, output, and state constraints, and it does so by solving a simple linear program with an explicit solution. This leads to a much more computationally-efficient algorithm compared to other existing optimization-based methods.

The main novelty of the proposed solution is that we apply RG in the iteration (batch) domain instead of the time domain; in other words, we treat the batch number as the independent variable, instead of the timestep within each batch. Furthermore, instead of assuming no knowledge of the model of the system, as is the case in ILC, or assuming perfect knowledge of the model, as is the case with RG, we assume an uncertain model of the system, wherein the system matrices are not known exactly. We use this modeling uncertainty to create a robust Maximal Admissible Set (MAS), and propose an RG based on this MAS to endow the ILC controller with constraint management capabilities. Since the robust MAS may lead to a conservative response, a method is provided to update the MAS as the system learns to further improve performance.

Finally, note that a variation of RG has been previously proposed in Tan et al. (2011) to handle input saturation in systems with ILC controllers. The RG proposed in Tan et al. (2011) reduces either the amplitude or the frequency of the reference signal so that it can be realized within the saturation bounds of the system. While the RG described in said paper does eventually compute an optimal input signal which enforces the constraints, there may be violations as the system learns. Our paper differs from Tan et al. (2011) in that it handles state and output constraints in addition to input saturation, and it can guarantee constraint satisfaction during the learning process.

2. Preliminaries

This section reviews the basics of ILC as seen in Arimoto et al. (1984); J. Xu et al. (2001); Yu et al. (2018); Moore (1998), and RG as it appears in Gilbert and Kolmanovsky (1995b); Kolmanovsky et al. (2014).

2.1. Iterative Learning Control (ILC)

ILC is a control method used for systems that perform a repeated task, e.g., a robotic arm in an assembly line, where the arm is to track some reference trajectory. Consider the discrete-time linear model describing the dynamics of the system:

$$\begin{aligned} x_k(t+1) &= Ax_k(t)+Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{1}$$

where $t \in \mathbb{Z}^+$ is the discrete time index, $k \in \mathbb{Z}^+$ is the iteration or batch number, $x_k(t) \in \mathbb{R}^n$ is the state of the system in batch k at time t, $u_k(t) \in \mathbb{R}^m$ is the input, and $y_k(t) \in \mathbb{R}^m$ is the output. A, B, and C are system matrices of appropriate dimensions. For simplicity, we assume that the system starts from zero initial conditions at every iteration, i.e., $x_k(0) = 0$ for all k. To illustrate the above variables with an example, consider a robotic manipulator whose end effector needs to follow a given path. The batch number k would represent one full run of the robot attempting to follow the path, and t represents the discrete-time (e.g., sampled time) during that run. The state $x_k(t)$ would be the internal states of the robot at a given time in a given batch, be that the torque being applied to a joint, or the angle and angular velocity of a joint.

Let $r(t)$ be a desired reference trajectory, defined on the time interval from $t = 0$ to some finite time $t = T$. The goal of ILC is to update the input $u_k(t)$ so that $y_k(t)$ converges to $r(t)$ as k tends to infinity

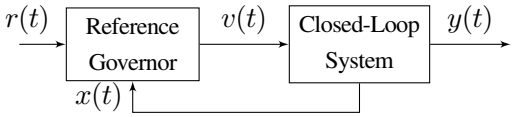

Figure 1: Reference governor block diagram.

(i.e., the goal is to make the system learn from the previous iterations). This can be achieved, for example, using a simple Arimoto ILC update law:

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{2}$$

where $e_k(t) = r(t) - y_k(t)$ is the tracking error in iteration (or batch) k, and $\gamma \in \mathbb{R}^+$ is the "learning coefficient". A larger γ will lead to faster convergence to the reference signal, but can cause the system to become unstable. For stability and convergence criteria of this algorithm, please see Moore (1998). Note that many variations of the ILC algorithm have been proposed, including those that use different learning coefficients for each input channel, and those with more complex update laws. For the sake of simplicity, we only consider the update law (2) in this paper.

As a final remark, note that ILC has traditionally been a "model-free" control technique in that, akin to classical controllers (e.g., PID), a model of the plant is not required inside the controller for implementation.

2.2. Reference Governors (RG)

RG is a method of constraint management that modifies the *reference signal* to a closed-loop control system, and is an add-on scheme to a traditional feedback control system. Since the inner loop dynamics of the system are not modified, RG is ideal for constraint management of "black-box" systems or systems with legacy controllers.

Consider Figure 1, in which the "closed-loop system" is described by the multi-input multi-output discrete-time stable linear system:

$$x(t+1) = Ax(t) + Bv(t) \tag{3}$$

where the state x is subject to the following polytopic constraints:

$$x(t) \in \mathbb{X} \triangleq \{x : Sx \leq s\} \tag{4}$$

Vector inequalities here and throughout the paper are to be interpreted element-wise. Note that constraints on states, outputs, and actuator commands can all be expressed using (4).

The RG employs the so-called maximal admissible set (MAS), denoted by O_∞, which is the set of all initial conditions and constant control inputs that satisfy (4) for all time:

$$O_\infty = \{(x_0, v_0) : x(0) = x_0,\, v(t) = v_0,\, x(t) \in \mathbb{X},\, \forall t \in \mathbb{Z}^+\} \tag{5}$$

As seen in (5), $v(t) = v_0$ is held constant for all t. Using this assumption, the evolution of the state $x(t)$ can be expressed explicitly as a function of $x(0) = x_0$ and v_0:

$$x(t) = A^t x_0 + (I - A^t)(I - A)^{-1} B v_0 \tag{6}$$

Therefore, MAS in (5) can be characterized by a polytope defined by an infinite number of inequalities:

$$O_\infty = \{(x_0, v_0) : SA^t x_0 + S(I - A^t)(I - A)^{-1} B v_0 \leq s,\, \forall t \in \mathbb{Z}^+\} \tag{7}$$

It is shown in Gilbert and Kolmanovsky (1995b); Gilbert and Tan (1991) that, to make this set finitely determined (i.e., be described by a finite number of inequalities), the steady-state value of $x(t)$, denoted by $x(\infty) := (I-A)^{-1}Bv_0$, must be constrained to the interior of the constraint set, i.e.,

$$S(I-A)^{-1}Bv_0 \leq (1-\epsilon)s \tag{8}$$

where $\epsilon \in \mathbb{R}^+$ is a small number. After introducing (8) in (7), it can be shown that there exists a finite prediction time j^*, where the inequalities corresponding to all future prediction times ($t > j^*$) are redundant.

Combining (7) and (8), O_∞ can be represented by a polytope of the form:

$$O_\infty = \{(x_0, v_0) : G_x x_0 + G_v v_0 \leq g\} \tag{9}$$

where the matrices G_x, G_v, and g are finite dimensional.

The above MAS is computed offline. In real-time, RG employs the MAS to select an optimal control input that will not cause a constraint violation at any future time. The RG update law that achieves this is:

$$v(t) = v(t-1) + \lambda(r(t) - v(t-1)) \tag{10}$$

where $\lambda \in [0,1]$. To select λ, the RG solves the following linear program at every timestep:

$$\begin{aligned} \underset{\lambda \in [0,1]}{\text{maximize}} \quad & \lambda \\ \text{s.t.} \quad & \big(x(t),\, v(t-1) + \lambda(r(t) - v(t-1))\big) \in O_\infty \end{aligned} \tag{11}$$

where $x(t)$, $r(t)$, and $v(t-1)$ are known parameters at time t. If the reference $r(t)$ is feasible, then the solution to (11) is $\lambda = 1$ and, therefore, $v(t) = r(t)$. If, however, the reference $r(t)$ is not feasible, then $\lambda < 1$.

A few important properties of RG are as follows. First, if the initial condition $(x(0), v(0))$ is inside O_∞, then the solution $\lambda = 0$ is always feasible in the optimization problem, and the constraints will never be violated. Thus, the RG formulation is recursively feasible. Second, for a bounded reference $r(t)$, $v(t)$ is a convex combination of $r(t)$ and $v(t-1)$, which implies that $v(t)$ is bounded as well. Lastly, for a constant $r(t)$, $v(t)$ converges in finite time.

3. Main Results

3.1. Control Method

As mentioned in the Introduction, this paper investigates a method of control that combines ILC and RG to enforce the constraints during the ILC learning process. Recall from Section 2.2 that the traditional RG algorithm governs the reference signal to a closed-loop system to enforce the constraints (see Figure 1). In this paper, this idea is preserved, but instead of governing the reference at each discrete time-step, the entire reference signal is governed at each iteration. In other words, the RG is implemented on the iteration (i.e., k) domain, as opposed to the time (i.e., t) domain.

To elaborate, a high-level block diagram of the proposed control strategy in shown in Figure 2, where the "Plant" is described by system (1), and the signals r, x_k, y_k, and u_k in the figure represent the lifted versions of $r(t)$, $x_k(t)$, $y_k(t)$, and $u_k(t)$ as defined below:

$$r = \begin{bmatrix} r(1) \\ \vdots \\ r(T) \end{bmatrix}, x_k = \begin{bmatrix} x_k(1) \\ \vdots \\ x_k(T) \end{bmatrix}, y_k = \begin{bmatrix} y_k(1) \\ \vdots \\ y_k(T) \end{bmatrix}, u_k = \begin{bmatrix} u_k(0) \\ \vdots \\ u_k(T-1) \end{bmatrix} \tag{12}$$

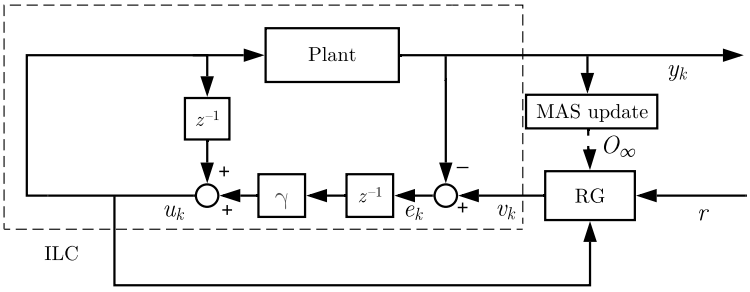

Figure 2: A block diagram of the proposed strategy. All signals are lifted signals as defined in (12). The plant is given by $y_k = H_y u_k$, where H_y is given in (13). Note: z^{-1} denotes a one-step delay in the iteration (i.e., k) domain.

Here, $r,\ y_k,\ u_k \in \mathbb{R}^{mT}$ and $x_k \in \mathbb{R}^{nT}$, where m is the number of inputs/outputs of the plant, n is the number of states, and T is the number of discrete time steps in each batch. Lifting system (1) with this notation, y_k and x_k can be expressed as $y_k = H_y u_k$ and $x_k = H_x u_k$, where H_y and H_x are given by:

$$H_y = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & & & \\ CA^{T-1}B & CA^{T-2}B & \dots & CB \end{bmatrix}, H_x = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & & & \\ A^{T-1}B & A^{T-2}B & \dots & B \end{bmatrix} \tag{13}$$

Now consider the ILC law in (2), with the reference $r(t)$ replaced by the governed reference $v_k(t)$. After lifting this update law and a handful of algebraic manipulations, a state-space model for the closed-loop iteration-domain dynamics of the ILC algorithm can be formulated as:

$$\begin{aligned} u_{k+1} &= (I - \gamma H_y)u_k + \gamma v_k \\ x_k &= H_x u_k, \quad y_k = H_y u_k \end{aligned} \tag{14}$$

where $v_k \in \mathbb{R}^{mT}$ is the lifted version of $v_k(t)$. Next, suppose the goal is to enforce the constraint $x_k(t) \in \mathbb{X}$ on system (1). Using the relation $x_k = H_x u_k$, we recast this constraint in terms of the lifted system:

$$H_x u_k \in \underbrace{\mathbb{X} \times \mathbb{X} \times \cdots \times \mathbb{X}}_{T \text{ terms}} \tag{15}$$

where $\times$ denotes the Cartesian product. The iteration-domain RG proposed in this paper is designed based on the lifted system (14) (treating u_k as the state) with constraint (15). This requires the computation of the MAS, $O_\infty \subset \mathbb{R}^{2Tm}$, for (14), (15). Note that, as explained in Section 2.2, computing the MAS requires tightening the constraint on the steady-state value of the state. It can be shown that this is possible for system (14) if the eigenvalues of $I - \gamma CB$ are inside the unit disk. In situations where this condition fails because $CB = 0$ (e.g., the relative degree of the system is greater than 1), the definition of the lifted output y_k in (12) can be slightly modified to overcome this issue (see Moore et al. (1993) for details).

Finally, the iteration-domain RG update law is as follows:

$$v_k = v_{k-1} + \lambda(r - v_{k-1})$$

where $\lambda \in [0,1]$ is obtained by solving the following linear program after every iteration:

$$\begin{aligned} \underset{\lambda \in [0,1]}{\text{maximize}} \quad & \lambda \\ \text{s.t.} \quad & \big(u_k,\ v_{k-1} + \lambda(r - v_{k-1})\big) \in O_\infty \end{aligned} \tag{16}$$

Since the proposed RG algorithm is essentially a standard RG applied to the lifted system, it enjoys the properties described in the following proposition.

Proposition 1 *Suppose the initial condition of the system satisfies* $(u_0,v_0)\in O_\infty$. *Then, formulation* (16) *is recursively feasible, guarantees constraint satisfaction for all* t *and* k, *and guarantees convergence of* v_k *and, hence,* y_k *as* k *tends to infinity.*

Note that the iteration-domain O_∞ has a much higher dimension than a time-domain O_∞, because it has to account for the entire time-history of the signal in a given iteration. One may be led to believe that this would cause large computation times, but due to the structure of the linear program used in the RG, this computation is still tractable, as illustrated in Section 3.2.

3.2. Robust RG/ILC Formulation

As mentioned in Section 2.1, ILC has traditionally been a model-free control technique. The RG, on the other hand, is a model-based technique that requires a faithful model of the plant in order to enforce the constraints. To resolve this apparent discrepancy, we now present a modification of the strategy presented in Section 3.1 to account for uncertainties.

To deal with modeling uncertainties using RG, a robust MAS, denoted by O_∞^{robust}, must be created. To accomplish this, the methods outlined in Pluymers et al. (2005) and Gilbert and Kolmanovsky (1995a) may be used. Specifically, Pluymers et al. (2005) presents a method for generating MAS robust to systems with "polytopic uncertainties", where the actual system matrices are unknown but lie inside the convex hull of known matrices. Certain aspects of this method make it rather computationally expensive, and considering the dimension of the matrices we will be dealing with, this method is intractable. Gilbert and Kolmanovsky (1995a) presents an alternative, more computationally-tractable method for creating the robust MAS, by assuming that the system is affected by set-bounded disturbances. The main idea is to "shrink" the MAS to account for the worst case realization of the disturbances at any given time.

We take a simpler approach in our paper to create the robust MAS. Specifically, suppose a nominal (possibly inaccurate) model of the system is given. We construct an O_∞ for this nominal model using the approach presented in Section 2.2. This leads to a characterization of a "non-robust" O_∞ with the form shown in (9). To robustify this set, we radially shrink it as follows:

$$O_\infty^{robust}=\{(x,v):G_x x+G_v v\leq \beta g\} \tag{17}$$

where $0<\beta<1$ is a parameter that adjusts the amount of shrinking that the MAS experiences. It must be chosen small enough to capture the effects of modeling uncertainties and disturbances, but not too small so as to avoid making the response overly conservative (we call this "over-governing", as described below).

To illustrate the above ideas with a numerical example, consider system (1) with known B and C matrices, and an uncertain A matrix. Below are the nominal A, B, and C used to create O_∞, as well as the actual A matrix, A_{actual}.

$$A=\begin{bmatrix}0.0438 & -0.4387\\ 0.4387 & 0.7018\end{bmatrix},\ A_{actual}=\begin{bmatrix}0.0438 & -0.4000\\ 0.4387 & 0.8000\end{bmatrix},\ B=\begin{bmatrix}0.4387\\ 0.2982\end{bmatrix},\ C=\begin{bmatrix}0.5 & 0.5\end{bmatrix}$$

Using the above matrices, the robust MAS, O_∞^{robust}, is created as discussed above with an output constraint of $-1\leq y_k(t)\leq 1$, and a β of 0.8. For this O_∞^{robust}, the G_x, G_v, and g matrices are 960×30, 960×30, and 960×1, respectively. The RG/ILC algorithm in Section 3.1 is then implemented with this robust

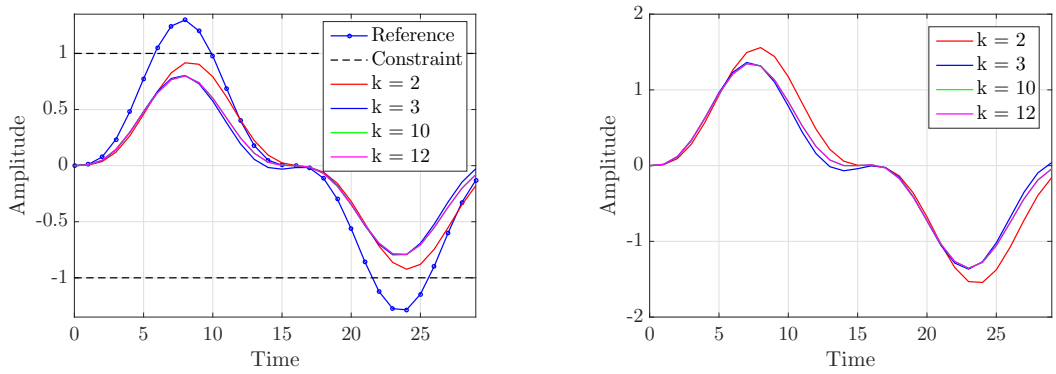

Figure 3: Simulation results. Left: system output, $y_k(t)$, for various iterations. The dashed constant lines show the imposed constraints. Right: The control input, $u_k(t)$, for each iteration respectively

MAS and the ILC learning coefficient of $\gamma = 2$. A numerical simulation is performed in MATLAB using a laptop computer equipped with an Intel Core i7 CPU and 16 GB of RAM. The desired reference trajectory for the simulation is assumed to be $r(t) = 1.3\sin^3(0.2t)$. Figure 3 shows the output and control input of the simulated system. As can be seen in the figure, in each iteration, the constraints are satisfied for all t. Also, after $k = 10$ iterations, the output has converged (i.e., does not change significantly with further iterations) and the learning is complete. Note that the RG linear program in (16) was implemented using an explicit algorithm (similar to Liu et al. (2018)). The mean computation time of this algorithm was 4.5 ms for this example, which shows that the proposed scheme is computationally tractable.

Notice that the output response is overly conservative, as evidenced by the gap between the output and the constraint, even at higher iterations. This implies that optimal tracking has not been achieved. The reason for this is that the O_∞^{robust} is too conservative (i.e., the system has been made too robust to modeling errors). We refer to this phenomenon as "over-governing", which occurs when the RG predicts that constraints will be violated when, in fact, the system is safe. We provide a solution to remedy this issue in the next subsection.

3.3. Addressing the issue of over-governing

Recall that the robust MAS, O_∞^{robust}, was created by radially shrinking a MAS created using a nominal model of the system. Thus, to overcome the issue of over-governing described above, we reverse this operation and gradually enlarge (i.e., radially expand) O_∞^{robust} as follows: after every N iterations, with N being a tunable parameter that will be discussed later, the value of β in (17) is incremented towards 1. To be more specific, recall that the constraint that we wish to impose on the system is given by $x_k(t) \in \mathbb{X}$, where $\mathbb{X} \triangleq \{x : Sx \leq s\}$. Now, let us introduce the following two parameters: let e_k^s be the smallest distance of the state from the constraint in iteration k, that is $e_k^s = \min_t \min_i (s_i - S_i x_k(t))$, where the subscript i denotes the i-th row. As an illustration, the constraint, s, is visualized by the constant dashed lines in Figure 3 and e_k^s can be viewed as the smallest gap between the output and the constraint. Let e_k^r be the maximum value of the tracking error in iteration k, that is $e_k^r = \max_t \max_i |r_i(t) - y_{ki}(t)|$, where i as before denotes the i-th row. Using this notation, β is updated after every N-th iteration as follows:

if $e_k^s > \alpha$ **then**
  if $e_k^r > \xi$ **then**
    $\beta \leftarrow \beta + \rho$

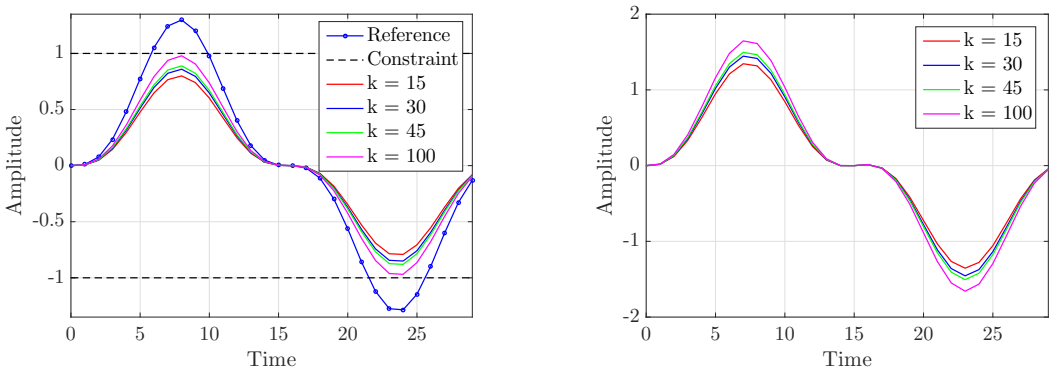

Figure 4: Left: The output y_k of the simulated system at various iterations. Right: The input u_k of the simulates system at various iterations.

where ξ and α are user-defined threshold values, and ρ is the user-defined incremental update of O_∞^{robust}. As mentioned previously, N is the number of iterations between updates of O_∞^{robust}. This is selected sufficiently large to allow the transient of the learning response to die out before making an update to the MAS. As can be seen in Figure 3, the transient of the learning response dies out around $k=10$, so it is advised to choose $N>10$. An N smaller than this may lead to constraint violation, as the learning may not be completed. An N larger than this will ensure that learning is complete, but overly large values of N will slow down the MAS updates.

To further explain the rationale behind the above algorithm, we note that over-governing is determined by the distance of the output from the constraint (i.e., e_k^s is large) in situations in which the output does not track the reference (i.e., e_k^r is large). In these situations, the update algorithm above will continually update β to reduce the effect of over-governing. Note that the condition $e_k^s>\alpha$ is required to ensure that the updates of β do not lead to an over-relaxation of the set, and the condition $e_k^r>\xi$ is introduced to ensure that the set is not relaxed when the tracking performance is already within an acceptable level.

Note that larger values of ρ could lead to over-relaxation of the robust MAS and, hence, constraint violation. To prevent this, we recommend to select ρ as follows: $\rho\leq\frac{\alpha}{\|s\|_\infty}$.

Finally, we recommend to select the parameters α and ξ to be 1 to 5% of $\|s\|_\infty$ (i.e. the value of the constraint). Smaller values would lead to smaller updates and therefore slower convergence (since ρ is recommended to be less than $\frac{\alpha}{\|s\|_\infty}$). Larger values of α and ξ will lead to faster convergence, but may lead to over-relaxation and therefore constraint violation. Thus, a trade-off must be made.

Figure 4 shows the over-governed system in Section 3.2 implemented with the MAS updating algorithm from above. For this simulation, N is set to 15 iterations, γ is set to 2, and $\alpha=\rho=\xi=0.03$. At iteration 15, the system is being over-governed, but eventually is within α of the constraint as the MAS is updated.

4. Conclusion

In this paper, a novel combination of ILC and RG was formed to pose a solution to constraint management for ILC. Specifically, the standard RG formulation was modified to govern the iteration domain dynamics of an ILC algorithm. The RG was endowed with robustness properties through a robust maximal admissible set. As the algorithm learns, this set is updated to allow better tracking of the reference signal inside constraint boundaries. Future work includes a data-driven approach to estimating the system being controlled as the algorithm learns, and extending the work to other constraint management schemes such as command governors.

References

S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operation of robots by learning. *Journal of Robotic systems*, 1(2):123–140, 1984.

E. G. Gilbert and I. Kolmanovsky. Discrete-time reference governors for systems with state and control constraints and disturbance inputs. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 2, pages 1189–1194 vol.2, Dec 1995a.

E. G. Gilbert and I. Kolmanovsky. Discrete-time reference governors for systems with state and control constraints and disturbance inputs. In *Proc. IEEE Conference on Decision and Control*, volume 2, pages 1189–1194, Dec 1995b.

E. G. Gilbert and K. T. Tan. Linear systems with state and control constraints: the theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36(9):1008–1020, Sep. 1991.

J. Xu, T. H. Lee, and H. Zhang. Comparative studies on repeatable runout compensation using iterative learning control. In *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, volume 4, pages 2834–2839 vol.4, June 2001.

X. Jin, Z. Wang, and R. H. S. Kwong. Convex optimization based iterative learning control for iteration-varying systems under output constraints. In *11th IEEE International Conference on Control Automation (ICCA)*, pages 1444–1448, June 2014.

I. Kolmanovsky, E. Garone, and S. Di Cairano. Reference and command governors: A tutorial on their theory and automotive applications. In *2014 American Control Conference*, pages 226–241, June 2014.

N. Li, I. V. Kolmanovsky, and A. Girard. A reference governor for nonlinear systems with disturbance inputs based on logarithmic norms and quadratic programming. *IEEE Transactions on Automatic Control*, pages 1–1, 2019.

Y. Liu, J. Osorio, et al. Decoupled reference governors for multi-input multi-output systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1839–1846. IEEE, 2018.

P. C. Marchal, O. Sörnmo, B. Olofsson, A. Robertsson, J. Gómez Ortega, and R. Johansson. Iterative learning control for machining with industrial robots. *IFAC Proceedings Volumes*, 47(3):9327 – 9333, 2014. ISSN 1474-6670. 19th IFAC World Congress.

K. L. Moore. Multi-loop control approach to designing iterative learning controllers. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, volume 1, pages 666–671 vol.1, Dec 1998.

K. L. Moore, M. Johnson, and M. J. Grimble. *Iterative Learning Control for Deterministic Systems*. Springer-Verlag, Berlin, Heidelberg, 1993. ISBN 0387197079.

J. Osorio and H. R. Ossareh. A stochastic approach to maximal output admissible sets and reference governors. *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 704–709, 2018.

H. R. Ossareh. Reference governors and maximal output admissible sets for linear periodic systems. *International Journal of Control*, 0(0):1–13, 2019.

B. Pluymers, J. A. Rossiter, J. A. K. Suykens, and B. De Moor. The efficient computation of polyhedral invariant sets for linear systems with polytopic uncertainty. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 804–809 vol. 2, June 2005.

Z. Ruikun and C. Ronghu. Iterative learning control for a class of mimo nonlinear system with input saturation constraint. In *2017 36th Chinese Control Conference (CCC)*, pages 3543–3347, July 2017.

G. Sebastian, Y. Tan, D. Oetomo, and I. Mareels. Iterative learning control for linear time-varying systems with input and output constraints. In *2018 Australian New Zealand Control Conference (ANZCC)*, pages 87–92, Dec 2018.

Y. Tan, J.X. Xu, M. Norrlöf, and C. Freeman. On reference governor in iterative learning control for dynamic systems with input saturation. *Automatica*, 47(11):2412 – 2419, 2011. ISSN 0005-1098.

Q. Yu, X. Bu, R. Chi, and Z. Hou. Modified p-type ilc for high-speed trains with varying trial lengths. In *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1006–1010, May 2018.

R. Zhang, Z. Hou, R. Chi, and Z. Li. Data-driven iterative learning control for i/o constrained lti systems. In *2016 35th Chinese Control Conference (CCC)*, pages 3166–3171, July 2016.