University of Vermont
UVM ScholarWorks

Graduate College Dissertations and Theses | Dissertations and Theses

2020

The Iteration Domain Reference Governor, a Constraint Management Scheme for Batch Processes

Aidan Ridner Laracy
University of Vermont

Follow this and additional works at: https://scholarworks.uvm.edu/graddis

 Part of the Engineering Commons

Recommended Citation

Laracy, Aidan Ridner, "The Iteration Domain Reference Governor, a Constraint Management Scheme for Batch Processes" (2020). *Graduate College Dissertations and Theses*. 1259.
https://scholarworks.uvm.edu/graddis/1259

This Thesis is brought to you for free and open access by the Dissertations and Theses at UVM ScholarWorks. It has been accepted for inclusion in Graduate College Dissertations and Theses by an authorized administrator of UVM ScholarWorks. For more information, please contact scholarworks@uvm.edu.

The Iteration Domain Reference Governor, a Constraint Management Scheme for Batch Processes

A Thesis Presented

by

Aidan Laracy

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Mechanical Engineering

August, 2020

Defense Date: June 5th, 2020
Dissertation Examination Committee:

Hamid Ossareh, Ph.D., Advisor
James Bagrow, Ph.D, Chairperson
Dryver Huston, Ph.D.
Mads Almassalkhi, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

Abstract

In this work, 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 is proposed. 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 Admissible Set (MAS) is proposed that endows the ILC algorithm with constraint management capabilities. The proposed method is applied to the Scalar Reference Governor (SRG), the Vector Reference Governor (VRG) and the Command Governor (CG). An update law on the MAS is proposed to further improve performance.

For my family

Acknowledgements

I'd like to thank my advisor, Dr. Ossareh, for his guidance, assistance, and teachings during this past year.

I'd like to thank my committee for their time, questions, and interest in my work.

I'd like to thank Collin, Yudan, and Joycer for helping me to learn the ropes of RG, and for fielding all of my questions.

I'd like to thank my parents for their love and support throughout this degree. I couldn't have done it without y'all.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Initially proposed in [1], Iterative Learning Control (ILC) is a method of control used for systems that perform a task repeatedly. Similar to how humans learn from previous experiences, ILC controllers use information from previous iterations or batches to improve tracking performance. Its applications have been explored in high speed trains [2], hard disk drives [3], robotics [4], and numerous other systems performing a repetitive task [5–10].

To provide an illustrative example of a system controlled using ILC, consider the robotic arm in Figure 1.1, and imagine it is in a factory setting attempting to track some reference signal to perform a pickup and place task. The inputs to the system would be the motor torque at a given joint, the states would be the position, or velocity of a joint, and the output would be the position of the end effector of the robot. On the first attempt of the task, you would send an arbitrary input signal to the robot, and you would observe some response. On the next attempt (or iteration),

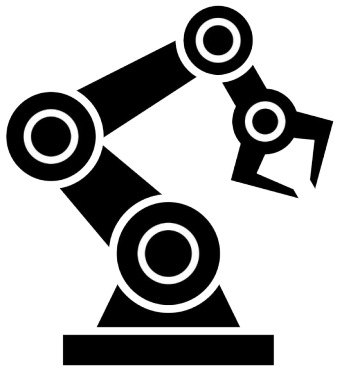

Figure 1.1: A robotic arm, for illustrative purposes.

you would send that same input signal, but also add a learning term based on the tracking error from the previous iteration of the task. Using this control method, the robot arm learns continually, and will eventually track the reference signal nearly perfectly. Currently, it is very difficult to tell what the system will do while it is learning, so constraints could be violated while the arm is learning.

One of the main challenges with ILC is enforcing constraints, be that input, output, or state constraints. To provide examples of constraints for an ILC controlled system, consider the example previously mentioned. Typical constraints in this system would be position and arm angle constraints, actuator saturation, and power consumption constraints. While this arm is learning, constraint violation could lead to damaged machine components, damaged machinery in the surrounding area, or injured factory workers.

Several schemes to solve the issue of constraint management in ILC have been proposed in the literature. These methods either are computationally expensive, do not consider output, state, or actuator constraints, or violate constraints as the system learns. This thesis provides a constraint management solution for ILC that

is computationally efficient, while being able to enforce input, output, and state constraints as the system learns.

1.2 Literature Review

In this section, we will be presenting a review of the current state of the literature on ILC, constraint management for systems controlled using ILC, and a brief overview of the current state of the art for reference governors.

1.2.1 Iterative Learning Control

The initial proposal of ILC in [1] was formulated in continuous time. This work proposed the derivative, or D-type ILC, where the time-derivative of the tracking error is used as the learning parameter. Shortly after this work, [11–13] proposed several ILC algorithms applied to robotic manipulators. These works were considered to have been the main elements that drove the popularity of ILC in the modern control community.

A PID-type ILC was proposed in [14], an improvement to the D-type ILC proposed in [1]. Later, model-based ILC was proposed in [15–17] to address more complex ILC problems, like MIMO systems. These model based methods were based on plant inversion.

More recent developments of ILC include the Quadratic criterion-based ILC (Q-ILC) algorithm. Q-ILC utilizes a quadratic program to solve for the control input at a given iteration. In [18], a quadratic criterion based ILC was proposed for batch processes subject to stochastic disturbances. Other variations of Q-ILC can be seen

in [19–21]

1.2.2 Constraint Management for Iterative Learning Control

Several schemes have been proposed in the literature to handle constraint management of systems controlled by ILC. In [22], 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. [23] 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 [24], 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. The work in [25] 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. In [26], a dual-loop ILC law was formed with a restrained learning law, and a saturated feedback law to deal with input constraint in a robotic arm experiment.

Optimization methods for constraint management in ILC are used in [27, 28]. This is done by formulating an ILC problem with a quadratic objective function and constraints, as a convex quadratic program. They also produce methods on reducing computational complexity, making the optimization easier to solve. Results are applied to a temperature control system for buildings in [28]. Barrier functions

are used in [29–33] to enforce output and input constraints.

The work in [34–37] uses constrained optimization techniques using super vector notation to enforce hard constraints.

Machine learning methods are explored in [38, 39] In [38], a genetic algorithm based optimization technique is used for constraint management in ILC. This method is able to deal with non-linearities and constraints. [39] uses a neural network based ILC algorithm to deal with speed constraints and actuator faults in a subway train.

The work proposed in [40] utilizes a feedback based PD type ILC controller with input constraints on a robotic manipulator.

One of the most compelling prior works that motivates the work presented in this thesis is seen in [41], which is another combination of the Reference Governor (RG) and ILC. The RG proposed in [41] reduces either the amplitude or the frequency of the reference signal, so that it can be realized within the saturation bounds of the system. The proposed solution in [41] does eventually reach an optimal input signal, but the imposed constraints are violated as the system learns. The work in this thesis differs from that in [41] in that it is able to enforce output, and state constraints in addition to input saturation constraints, and guarantees constraint satisfaction as the system learns. The above papers either do not consider output or state constraints, or use nonlinear or quadratic programming to update the control signal.

1.2.3 Reference Governor

The RG is an add-on scheme for pre-stabilized control systems. The main goal is to enforce constraints by modifying the reference signal (see Figure 1.2). The add-on functionality is ideal for black box or legacy type controllers. To give a brief

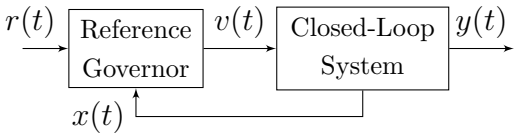

Figure 1.2: Reference governor block diagram.

background, the RG was initially proposed as a continuous-time control scheme in [42], and was later introduced in the discrete time framework in [43, 44], due to the fact that the continuous-time RG is not practically implementable. The static RG was proposed in [43] which had the control update law $v(t) = \lambda(t)r(t)$. Due to the chance of oscillations, it was succeeded by the dynamic RG, which has the ability of finite-time convergence for constant reference signals.

The dynamic RG includes methods such as the Scalar Reference Governor (SRG), as introduced in [43–46]; the Vector Reference Governor (VRG), as introduced in [47]; and the Command Governor (CG), as seen in [48–50], which was later extended to the Extended Command Governor (ECG). Each of these methods is based on the Maximal Admissible Set (MAS) [51], or the set of all initial conditions and constant control inputs that will satisfy constraints. The MAS will be explained in further detail later.

Applications of the RG have been explored in turbocharged automobile engines in [52–55] protecting against compressor surge, robots with actuator constraints [56], manufacturing equipment [57, 58], aerospace systems [59, 60], power distribution systems [61, 62], and many other areas.

Scalar Reference Governor

The SRG utilizes the maximal admissible set, or the set of all possible initial conditions and constant control inputs which satisfy constraints, to govern the reference signal to a closed loop system as a form of constraint management. The advantage of the SRG is that the optimization solved at each time step is a simple linear program that can be solved explicitly, making it very computationally efficient. The drawback is that there is only one optimization parameter, so for systems with more than one input, the SRG will produce an overly-conservative response.

Vector Reference Governor

Similar to the SRG, the VRG utilizes the maximal admissible set to form a control signal that is constraint admissible. The main difference between the SRG and the VRG is that instead of having only one optimization parameter, the VRG contains multiple. Specifically, for a multiple-input multiple-output system with multiple channels, the VRG has an optimization parameter for each channel. Instead of a simple linear program to solve the optimization problem, the VRG utilizes a quadratic program, making it more computationally expensive. A trade-off must be made between tracking capability and computational costs.

Command Governor

The CG is another RG method used for systems with more than one input. The CG also makes use of the maximal admissible set, but instead of using the optimization variable λ (which will be explained in Chapter 2), the control signal is optimized directly. In other words, $v(t)$, as seen in Figure 1.2 is treated as the optimization

variable. This leads to quicker convergence for MIMO systems, but as with the VRG, the control signal is optimized using a quadratic program so a trade off must be made.

1.3 Contributions

ILC, as stated above, is a method of control used for systems that perform a task repeatedly. There is plenty of existing theory describing convergence properties of ILC, or whether or not the output of the controlled system will converge to the reference signal. However, there is not established theory that described *how* an ILC algorithm will converge. ILC systems can converge monotonically: where tracking gets better with each batch, or asymptotically (i.e. non-monotonically): where tracking will worsen before it improves. Often times (especially in the case of non-monotonic convergence), system constraints can be violated as an ILC algorithm is learning. Thus, constraint management is a very hot topic in the ILC community.

The state of the literature shows that when dealing with constraint management for ILC systems, current methods either violate constraints as the ILC-controlled system is learning, cannot enforce input, output, or state constraints, or use nonlinear or quadratic programming to update the control signal.

The RG, by nature, is able to handle input, output, and state constraints in a very computationally efficient manner. The work presented in this thesis outlines a method using a lifted traditional ILC algorithm in conjunction with RG theory. The proposed method is able to enforce input, output, and state constraints for systems controlled by ILC, and is able to do so in a very computationally efficient manner thanks to the RG

To reiterate, the contributions of this work are:

- A constraint management scheme for systems controlled by ILC, that is able to enforce input, output, and state constraints, is robust to modeling uncertainties, and is computationally efficient.
- A new way of modeling uncertainties through radial scaling of the maximal admissible set.
- An algorithm to reduce the effects of an overly robust maximal admissible set.
- Various illustrative examples to demonstrate the efficacy of the proposed control solution.

1.4 Thesis Outline

Here, we will be giving a brief overview of the rest of this document. Chapter 2 will be a review on the theory of ILC, construction of the maximal admissible set, and theory of RG, VRG, and CG. Chapter 3 will give an overview of the contribution: the Iteration Domain Reference Governor (IDRG). Chapter 4 will show applications of the IDRG on different systems using SRG, VRG, and CG, and demonstrations of the MAS updating algorithm. Chapter 6 will present a summary of the work and a discussion of future works.

1.5 An Explanation of the Iteration Domain

In this work, we will often refer to the "iteration domain" or the "iteration dynamics" of a system. The iteration domain refers to the *batch* domain of the system. In other words, one iteration is equivalent to one batch of a repetitive system performing a task. When we refer to iteration dynamics, we are referring to how the dynamics of the batches are behaving as they evolve. In this thesis, we use the variable k to denote the iteration or batch number.

Chapter 2

Preliminaries

2.1 Iterative Learning Control

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:

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{2.1}$$

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 (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:

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{2.2}$$

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. Work from [63] addresses stability and convergence criteria of this algorithm. 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 (2.2) in this thesis.

We will now define the lifted version of the system in (2.1). To begin, the lifted versions of $r(t)$, $x_k(t)$, $y_k(t)$, and $u_k(t)$ are defined as r, x_k, y_k, and u_k respectively.

$$r = \begin{bmatrix} r(1) \\ \vdots \\ r(T) \end{bmatrix}, \; x_k = \begin{bmatrix} x_k(1) \\ \vdots \\ x_k(T) \end{bmatrix}, \; y_k = \begin{bmatrix} y_k(1) \\ \vdots \\ y_k(T) \end{bmatrix}, \; u_k = \begin{bmatrix} u_k(0) \\ \vdots \\ u_k(T-1) \end{bmatrix} \tag{2.3}$$

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 (2.1) 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:

$$H_y = \begin{bmatrix} CB & 0 & ... & 0 \\ CAB & CB & ... & 0 \\ \vdots & & & \\ CA^{T-1}B & CA^{T-2}B & ... & CB \end{bmatrix},\ H_x = \begin{bmatrix} B & 0 & ... & 0 \\ AB & B & ... & 0 \\ \vdots & & & \\ A^{T-1}B & A^{T-2}B & ... & B \end{bmatrix} \tag{2.4}$$

Now consider the ILC law in (2.2). 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:

$$\begin{aligned} u_{k+1} &= (I - \gamma H_y)u_k + \gamma r \\ x_k &= H_x u_k, \quad y_k = H_y u_k \end{aligned} \tag{2.5}$$

It is apparent that the ILC algorithm will converge if the system (2.5) is asymptotically stable. This condition is true if the eigenvalues of $I - \gamma H_y$ are inside the unit disk. Since H_y is a diagonal matrix, and $I - \gamma CB$ is on each diagonal element, then to put it more simply, the ILC algorithm will converge if the eigenvalues of $I - \gamma CB$ are in the unit disk. This is identical to the original ILC convergence criteria described by Arimoto. A block diagram of ILC described by (2.5) can be seen in Figure 2.1

The lifted system in (2.5) will be very important later on.

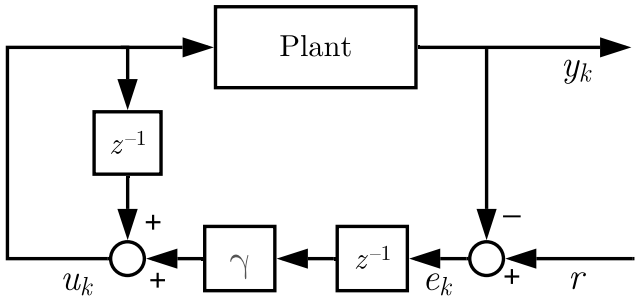

Figure 2.1: A block diagram of the ILC. All signals are lifted signals as defined in (5.5)*. The plant is given by* $y_k = H_y u_k$*, where* H_y *is given in* (5.4)*. Note:* z^{-1} *denotes a one-step delay in the iteration (i.e., k) domain.*

2.2 Maximal Admissable Set

The Maximal Admissible Set (MAS), referred to as O_∞ will be introduced in this section. The O_∞ is defined as the set of all possible initial conditions, and constant control inputs, for which a given system will satisfy constraints.

To construct O_∞, consider the multi-input multi-output (MIMO), discrete-time stable system below

$$x(t+1) = Ax(t) + Bv(t) \tag{2.6}$$

where x is subject to polytopic constraints

$$x(t) \in \mathbb{X} \triangleq \{x : Sx \leq s\} \tag{2.7}$$

In this thesis, vector inequalities are to be interpreted element-wise, and note that constraints on the state, output, and input can all be expressed using (2.7).

As stated previously, O_∞ is defined as the set of all possible initial conditions and constant control inputs that satisfy constraints for all future time:

$$O_\infty = \{(x_0, v_0) : x(0) = x_0,\ v(t) = v_0,\ x(t) \in \mathbb{X},\ \forall t \in \mathbb{Z}^+\} \tag{2.8}$$

As seen in (2.8), $v = v_0$ is held constant for all t. Using this assumption, $x(t)$ can be be expressed as a function of the initial condition x_0, and the constant control input as:

$$x(t) = A^t x_0 + (I - A^t)(I - A)^{-1} B v_0 \tag{2.9}$$

Making a substitution, the MAS in (2.8) can also be defined a polytope with an infinite number of inequalities:

$$O_\infty = \{(x_0, v_0) : SA^t x_0 + S(I - A^t)(I - A)^{-1} B v_0 \leq s,\ \forall t \in \mathbb{Z}^+\} \tag{2.10}$$

Practically, this set cannot be used, as it is infinitely large. To make this MAS finitely determined, the steady state value of $x(t)$ must be constrained to the interior of the constraint set as shown in [47,51]. The steady state value of $x(t)$ can be defined as $x(\infty) := (I - A)^{-1} B v_0$, and is constrained to the interior of the constraint set as seen below:

$$S(I - A)^{-1} B v_0 \leq (1 - \epsilon) s \tag{2.11}$$

where ϵ is a small number. When (2.11) is introduced into (2.10), it can be shown

that there exists a finite time t^* such that for all times $t > t^*$, the corresponding inequalities are redundant. Combining (2.10) and (2.11), the MAS can be represented with a polytope of the form below:

$$O_\infty = \{(x_0, v_0) : G_x x_0 + G_v v_0 \leq g\} \tag{2.12}$$

where the matrices $G_x,\ G_v$ and g are finite dimensional, and have the following form:

$$G_x = \begin{bmatrix} 0 \\ S \\ SA \\ \vdots \\ SA^{t^*} \end{bmatrix} G_v = \begin{bmatrix} S(I-A)^{-1}B \\ 0 \\ S(I-A)(I-A)^{-1}B \\ \vdots \\ S(I-A^{t^*})(I-A)^{-1}B \end{bmatrix} g = \begin{bmatrix} (1-\epsilon)s \\ s \\ s \\ \vdots \\ s \end{bmatrix} \tag{2.13}$$

This MAS is computed offline to be used in the Reference Governor, as will be described next.

2.3 Reference Governor

The Reference Governor (RG) for linear systems, as seen in [47, 64–68], modifies the *reference signal* to a closed-loop system, and is an add-on scheme to a traditional feedback control system. The inner dynamics of the system are not modified, so the RG is ideal for "black-box" systems, or systems with legacy controllers. A block diagram of the RG can be seen in Figure 1.2. In the block diagram, $r(t)$ is the reference signal, $v(t)$ is the governed reference signal, $x(t)$ is the state, and $y(t)$ is the constrained output.

The main goal of the RG is to select a control input that does not violate constraints, which can be done using various different methods. In this thesis, we investigate three different RG methods, the Scalar Reference Governor (SRG), the Vector Reference Governor (VRG), and the Command Governor (CG).

2.3.1 Scalar Reference Governor

As mentioned in the previous section, the main goal of the RG is to select a control input such that system constraints will not be violated. When using the SRG, the optimal control input that achieves this is as follows:

$$v(t) = v(t-1) + \lambda(r(t) - v(t-1)) \tag{2.14}$$

where $\lambda \in [0, 1]$. To find λ the following linear program is solved at each discrete-time step.

$$\begin{aligned} &\underset{\lambda \in [0,1]}{\text{maximize}} \quad \lambda \\ &\text{s.t.} \qquad \Big(x(t),\ v(t-1) + \lambda(r(t) - v(t-1)\Big) \in O_\infty \end{aligned} \tag{2.15}$$

In this case, $x(t)$, $v(t-1)$ and $r(t)$ are known parameters. If the reference is feasible, then $\lambda = 1$, and if the reference is not feasible, then $\lambda < 1$. One of the benefits of the SRG is that this optimization problem can be solved explicitly, and therefore is extremely quick to solve.

To briefly explain some of the properties of RG, if the initial condition $(x_0, v_0) \in O_\infty$, then $\lambda = 0$ will always be a solution to the optimization problem in (2.15), so constraints will always be satisfied, and the RG formulation is recursively feasible.

Second, if the reference $r(t)$ is bounded, then $v(t)$ is also bounded, as it lies on the straight line between $r(t)$ and $v(t-1)$. Last, if $r(t)$ is constant, then $v(t)$ will converge in finite time.

2.3.2 Vector Reference Governor

The VRG is an extension of the SRG, but differs in that there are multiple λ for each channel of the system, and instead of an easy to solve linear program to calculate $v(t)$, a more complex quadratic program must be solved. In the VRG, the control signal is selected as follows:

$$v(t) = v(t-1) + \mathbf{\Lambda}(r(t) - v(t-1)) \tag{2.16}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_i)$. To solve for $\mathbf{\Lambda}$, the following quadratic program is solved at each discrete-time step.

$$\begin{aligned} \underset{\lambda_i \in [0,1]}{\text{minimize}} \quad & \|v(t) - r(t)\| \\ \text{s.t.} \quad & v(t) = v(t-1) + \mathbf{\Lambda}(r(t) - v(t-1) \\ & \Big(x(t),\ v(t)\Big) \in O_\infty \end{aligned} \tag{2.17}$$

For MIMO systems, the SRG will select a λ that satisfies constraints, but this may be overly conservative for some channels. Using the VRG, each channel has its own λ_i which leads to a less conservative response.

2.3.3 Command Governor

The Command Governor (CG) is a generalization of the traditional RG, and while it still utilizes O_∞, there is no optimization of λ. Instead, the control signal $v(t)$ is directly used as an optimization variable. This is beneficial for systems with multiple inputs because, similar to the VRG, there can be more than one optimization parameter. The control signal is updated at each discrete time step by solving the quadratic program seen below.

$$\begin{aligned} v(t) = \underset{v}{\text{minimize}} \quad & \|v(t) - r(t)\| \\ \text{s.t.} \quad & \Big(x(t), v(t)\Big) \in O_\infty \end{aligned} \tag{2.18}$$

For MIMO systems, the CG is able to provide quicker convergence compared to the SRG and VRG, but since a quadratic program is needed for the optimization, it is more computationally expensive than the SRG.

Chapter 3

Main Results

In this chapter, we present the main results of this thesis, the Iteration Domain Reference Governor (IDRG), and a method for relaxing a robust MAS in the case of "over-governing".

3.1 The Iteration Domain Reference Governor

As mentioned in the Introduction, this thesis investigates a method of control that combines ILC and RG to enforce the constraints during the ILC learning process. Recall from Section 2.3 that the traditional RG algorithm governs the reference signal to a closed-loop system to enforce the constraints (see Figure 1.2). In this thesis, 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.

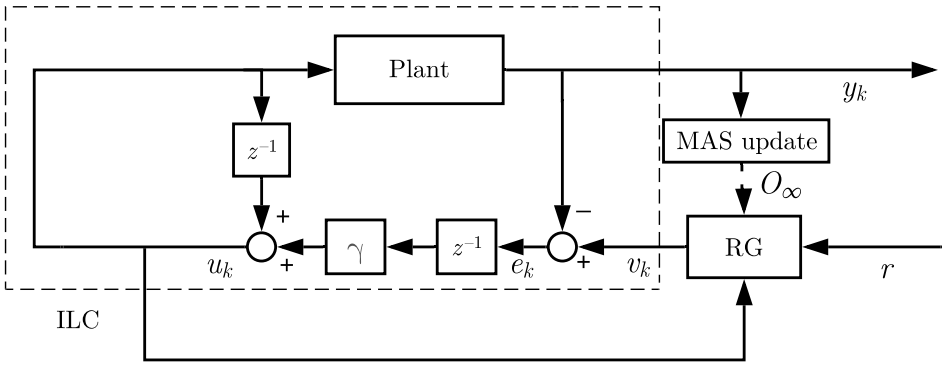

Figure 3.1: A block diagram of the proposed strategy. All signals are lifted signals as defined in (3.3)*. The plant is given by $y_k = H_y u_k$, where H_y is given in* (5.4)*. Note: z^{-1} denotes a one-step delay in the iteration (i.e., k) domain.*

To elaborate, a high-level block diagram of the proposed control strategy in shown in Figure 3.1, where the "Plant" is described by the following system:

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{3.1}$$

and is to be controlled by the Arimoto ILC law:

$$u_{k+1}(t) = u_k(t) + \gamma e_k(t+1) \tag{3.2}$$

The signals r, x_k, y_k, and u_k in the Figure 3.1 represent the lifted versions of $r(t)$, $x_k(t)$, $y_k(t)$, and $u_k(t)$ as defined below:

$$r = \begin{bmatrix} r(1) \\ \vdots \\ r(T) \end{bmatrix},\ x_k = \begin{bmatrix} x_k(1) \\ \vdots \\ x_k(T) \end{bmatrix},\ y_k = \begin{bmatrix} y_k(1) \\ \vdots \\ y_k(T) \end{bmatrix},\ u_k = \begin{bmatrix} u_k(0) \\ \vdots \\ u_k(T-1) \end{bmatrix} \tag{3.3}$$

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 (3.1) 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:

$$H_y = \begin{bmatrix} CB & 0 & ... & 0 \\ CAB & CB & ... & 0 \\ \vdots & & & \\ CA^{T-1}B & CA^{T-2}B & ... & CB \end{bmatrix},\ H_x = \begin{bmatrix} B & 0 & ... & 0 \\ AB & B & ... & 0 \\ \vdots & & & \\ A^{T-1}B & A^{T-2}B & ... & B \end{bmatrix} \tag{3.4}$$

Now consider the ILC law in (3.2), with the reference $r(t)$ replaced by the governed reference $v_k(t)$. After lifting this update law, and substituting $v_k(t)$ for $r(t)$, the lifted version of (3.2) can be expressed as:

$$u_{k+1} = u_k + \gamma e_k \tag{3.5}$$

Now, considering $e_k = v_k - y_k$, and $y_k = H_y u_k$, (3.5) becomes:

$$u_{k+1} = u_k + \gamma(v_k - H_y u_k) \tag{3.6}$$

and then through a slight algebraic manipulation, a state-space model for the closed-loop iteration-domain dynamics of the ILC algorithm can be formulated as:

$$\begin{aligned} u_{k+1} &= (I - \gamma H_y)u_k + \gamma v_k \\ x_k &= H_x u_k \\ y_k &= H_y u_k \end{aligned} \tag{3.7}$$

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 (2.1). Using the relation $x_k = H_x u_k$, we recast this constraint in terms of the lifted system:

$$H_x u_k \in \underbrace{\mathbb{X} \times \mathbb{X} \times \cdots \times \mathbb{X}}_{T \text{ terms}} \tag{3.8}$$

where $\times$ denotes the Cartesian product.

In other words, the matrices that describe the constraint set are defined as:

$$\begin{bmatrix} S & 0 & ... & 0 \\ 0 & S & ... & 0 \\ \vdots & & & \\ 0 & 0 & ... & S \end{bmatrix} \begin{bmatrix} x_k(1) \\ x_k(2) \\ \vdots \\ x_k(T) \end{bmatrix} \leq \begin{bmatrix} s \\ s \\ \vdots \\ s \end{bmatrix} \tag{3.9}$$

Remark 1 *The constraints defined here do not have to constant in time, they may be time-varying. Each row in the above inequality denotes a constraint on a certain instant in time in a given batch. Each entry s_i in the right hand side of the inequality may have a different value, where i denotes the i-th row.*

The iteration-domain RG proposed in this thesis is designed based on the lifted system (3.7) (treating u_k as the state) with constraint (3.9). This requires the computation of the MAS, $O_\infty \subset \mathbb{R}^{2Tm}$, for (3.7), (3.9). Note that, as explained in Section 2.3, 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 (3.7) 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 (3.3) can be slightly modified to overcome this issue (see [17] 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:

$$\begin{aligned} &\underset{\lambda \in [0,1]}{\text{maximize}} && \lambda \\ &\text{s.t.} && \Big(u_k,\ v_{k-1} + \lambda(r - v_{k-1})\Big) \in O_\infty \end{aligned} \tag{3.10}$$

It is important to note that this formulation does not modify the dynamics of the ILC, as the RG is outside of the ILC loop, and only v_k is modified. Thus, ILC convergence conditions still hold. To select a learning coefficient such that the output signal converges to the reference signal, γ must lead to all eigenvalues of $I - \gamma CB$ being inside the unit disk.

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* (3.10) *enjoys the following properties:*

1. *it is recursively feasible;*

2. *it guarantees constraint satisfaction for all* t *and* k*;*

3. *it guarantees convergence of* v_k *and, hence,* y_k *as* k *tends to infinity.*

Here, we provide a sketch of the proof. Selecting a $\lambda = 0$ will lead to $v_{k+1} = v_k$. Therefore, a $\lambda = 0$ is always a feasible solution to the optimization problem seen in (3.10). Constraint satisfaction qualities of the RG are preserved thanks to the definition of O_∞. To show that the convergence criteria of ILC is not modified, note that since only the reference signal is modified, the convergence condition discussed above still holds, as this is for an arbitrary reference signal.

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 Chapter 4.

3.2 Robust IDRG Formulation

ILC has traditionally been a model-free control technique, in that, similar to PID control, a model of the plant is not needed inside the controller for implementation. 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 [69] and [44] may be used. Specifically, [69] 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. [44] 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 this thesis 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 Chapter 2. This leads to a characterization of a "non-robust" O_∞ with the form shown in (2.12). To robustify this set, we radially shrink it as follows:

$$O_\infty^{robust} = \{(x, v) : G_x x + G_v v \leq \beta g\} \tag{3.11}$$

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. In this thesis, we refer to this overly conservative response as "over-governing", meaning that the RG believes that the system is going to violate constraints, when in fact it is safe.

To overcome the issue of over-governing, we reverse the tightening 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 (3.11) 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,

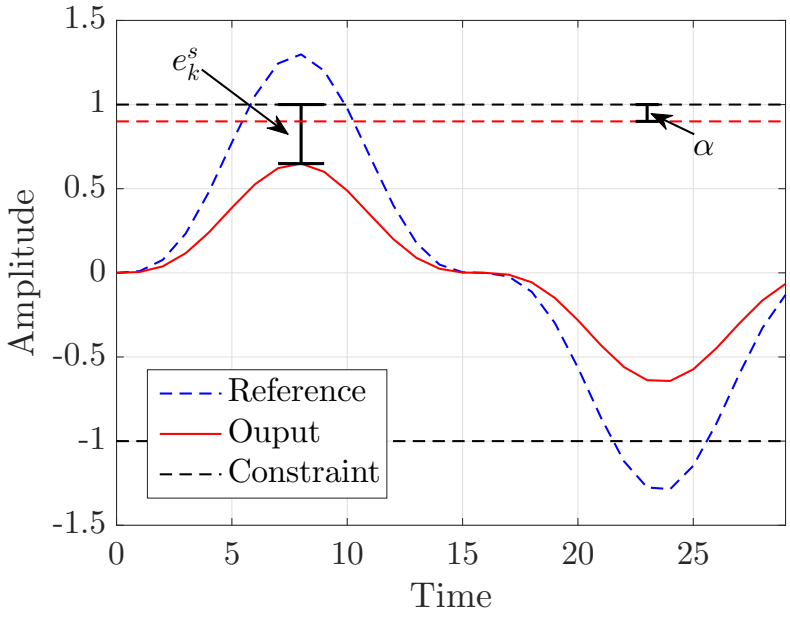

Figure 3.2: An illustration to help visualize α and e_k^s.

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. 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**
    $\beta \leftarrow \beta + \rho$

where ξ and α are user-defined threshold values, and ρ is the user-defined incremental update of O_∞^{robust}. See Figure 3.2 to help viusualize α and e_k^s.

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.

The parameter α can be thought of as the allowable distance of the output from the constraint. Normally this would be selected to be sufficiently small, so that the MAS is not overly relaxed during the MAS updating algorithm.

ρ is the amount β is increased for each update. This parameter is also selected to be sufficiently small, and we recommend to select ρ as follows: $\rho \leq \frac{\alpha}{\|s\|_\infty}$. A ρ larger than this would lead to quicker convergence to the optimal β, but could lead to an overshoot of the actual constraint, leading to violation.

The final parameter, ξ can be thought of as the maximum allowable tracking error. This parameter is checked so that β is not continually updated if the system is tracking well, but is far from the constraint. This way, the MAS is not continually updated which would make the RG algorithm become unstable.

Remark 2 *For this uncertainty method, matrix uncertainty may be time varying as long as the uncertainty is constant on the iteration domain. To provide an example of this with some system matrix $A_k(t)$, this matrix may vary with t. That is $A_k(1)$, $A_k(2)$ ... $A_k(T)$ may be different, but $A_1(1)$ must equal $A_k(1)$ for all k, and $A_1(2)$ must equal $A_k(2)$ for all k, and so on.*

Chapter 4

Illustrative Examples

In this chapter, we will be exploring a couple of applications of the IDRG. We will be looking at a SISO example with output constraints implemented with the IDRG, exploring state constraints for the same system, and then a MIMO system.

4.1 Output Constraints

To illustrate the results from Chapter 3, consider a system with known B and C matrices, and an uncertain A matrix of the following form

$$\begin{aligned} x_k(t+1) &= Ax_k(t) + Bu_k(t) \\ y_k(t) &= Cx_k(t) \end{aligned} \tag{4.1}$$

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 Chapter 3 is then implemented with this robust 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 $r(t) = 1.3\sin^3(0.2t)$. Figures 4.1 and 4.2 shows the output and control input of the simulated system respectively. 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 (3.10) was implemented using an explicit algorithm (similar to [66]). 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).

Now to mitigate the effects of over-governing, the MAS relaxing algorithm explained in Chapter 3 is implemented. To help visualize, e_k^s is the smallest distance of the output from the constraint. In this simulation, the parameters α, ρ, ξ are all set

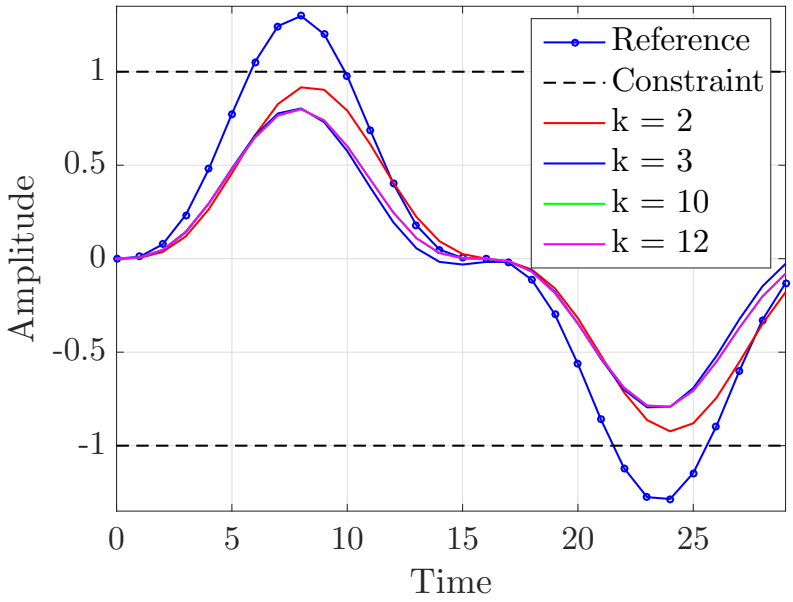

Figure 4.1: The system output y_k for various iterations. The dashed lines show the imposed constraint.

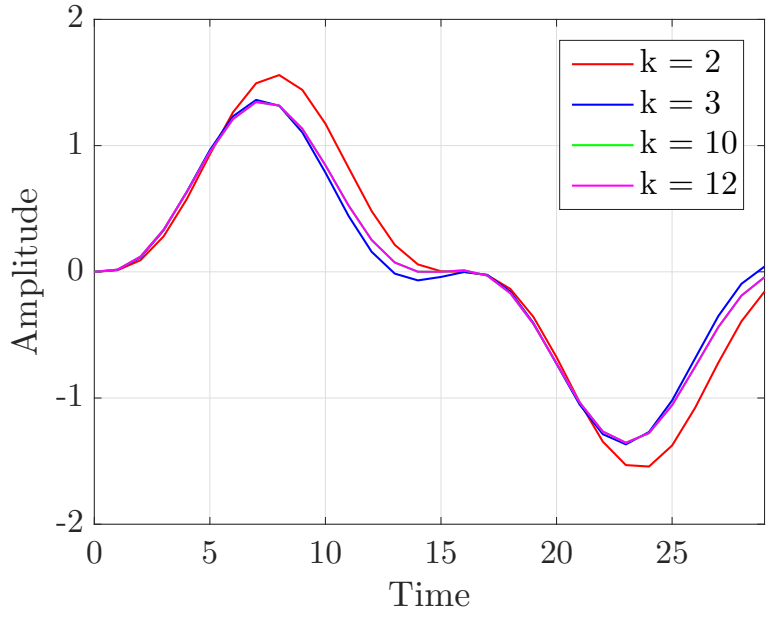

Figure 4.2: The control input u_k for various iterations.

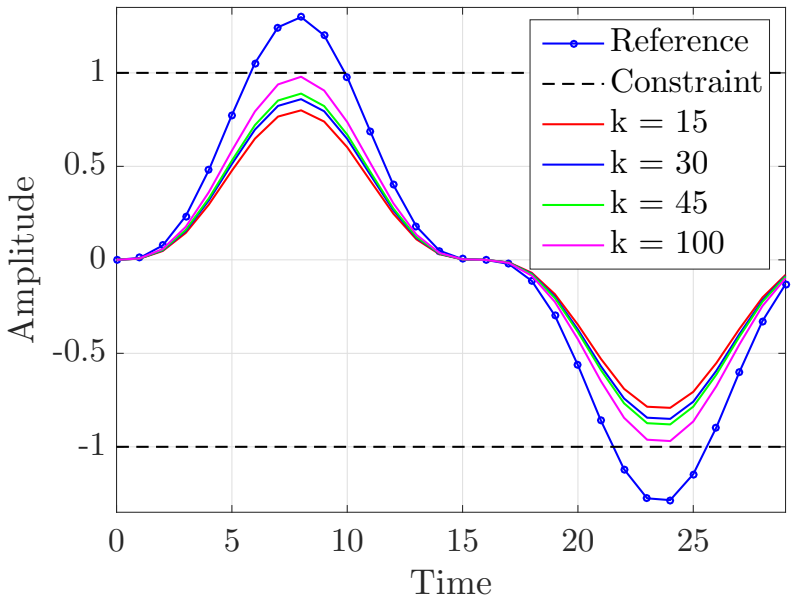

Figure 4.3: The system output y_k with the MAS updating law implemented.

to 0.03. The output and input of the system above implemented with the updating algorithm can be seen in Figures 4.3 and 4.4 respectively.

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.

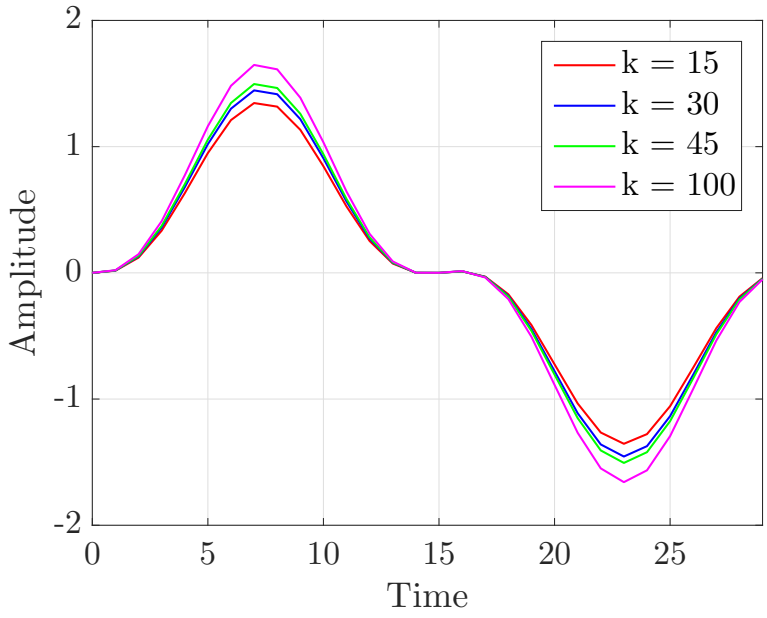

Figure 4.4: The system input u_k with the MAS updating law implemented.

4.2 State Constraints

As is stated in Chapter 3, the proposed control method can handle all sorts of constraints. To demonstrate other variations of constraints, the same example from the previous subsection will be recast with state constraints. Specifically, the constraints imposed on the state are $-.7 \leq x_{1,k}(t) \leq .7$, and $-1.5 \leq x_{2,k}(t) \leq 1.5$. All other parameters of the simulation are the same, including the A, B, and C matrices used to form the lifted model, and the matrix A_{actual}. Results of the simulation can be seen in Figure 4.5.

From this simulation, it is evident that the system is experiencing over-governing. To mitigate this, the updating algorithm from Chapter 3 is used, with $\alpha = \rho = \xi = 0.03$, as in the previous example.

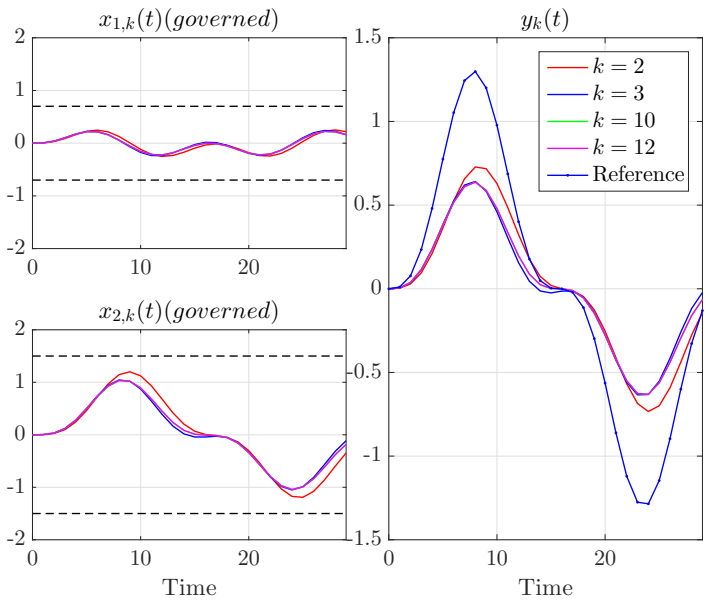

Figure 4.5: The same simulation as previous, but with constraints imposed on the state $x_k(t)$. The dashed lines show the imposed constraints.

The result of the updating algorithm implemented on the example with state constraints can be seen in 4.6. Notice that the parameter e_k^s accounts for the smallest distance of each state, from each constraint. This means that the MAS will not be continually updated if one output is at a constraint, and another is not. After 200 iterations, the state $x_{2,k}(t)$, is within α of it's constraint, and the updating algorithm is complete.

4.3 Multiple-Input Multiple-Output System

To further demonstrate the efficacy of the proposed control solution, an example of a MIMO system, with two inputs and two outputs was considered. The system had

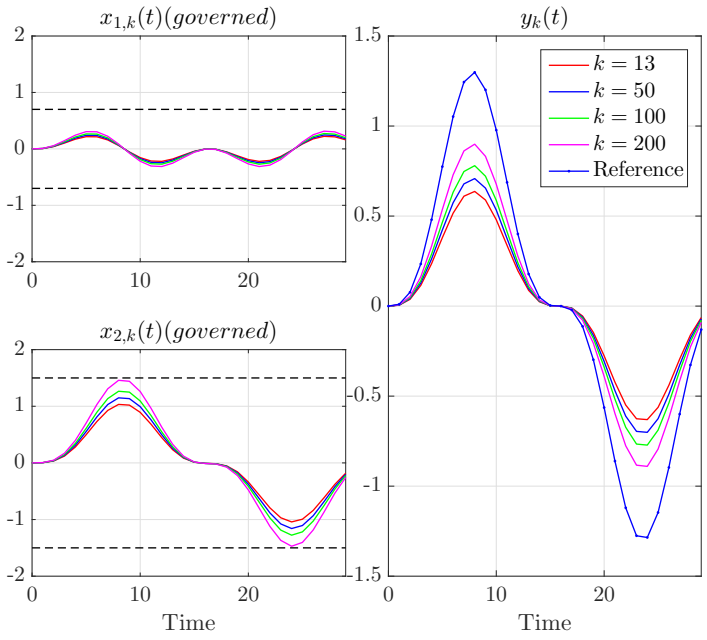

Figure 4.6: The simulation with state constraints, and the proposed MAS updating algorithm. The dashed lines show the imposed constraints.

the form of (4.1), with system matrices seen below.

$$A = \begin{bmatrix} 0.9 & 0 & 0 & 0 \\ 1 & 0.1 & 0.2 & 1 \\ 0 & 0 & 0.5 & 0.1 \\ 0.3 & 0 & 0 & 0.2 \end{bmatrix}, A_{actual} = \begin{bmatrix} 0.9 & 0 & 0 & 0 \\ 1 & 0.1 & 0.2 & 1 \\ 0 & 0 & 0.4 & 0.1 \\ 0.25 & 0 & 0 & 0.15 \end{bmatrix}, B = \begin{bmatrix} .5 & 0 \\ 0 & 0 \\ 0 & .0426 \\ 0.04285 & 0.1349 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The robust MAS is formed with the matrices A, B, and C, output constraints $-0.06 \leq y_1(t), y_2(t) \leq .06$, and $\beta = 0.8$. For this O_∞^{robust}, the G_v, G_x, g matrices were 40804×202, 40804×202 and 40804×1 respectively, and a learning coefficient of $\gamma = 0.5$ was used. The mean computation time for λ was 0.3059 seconds using the

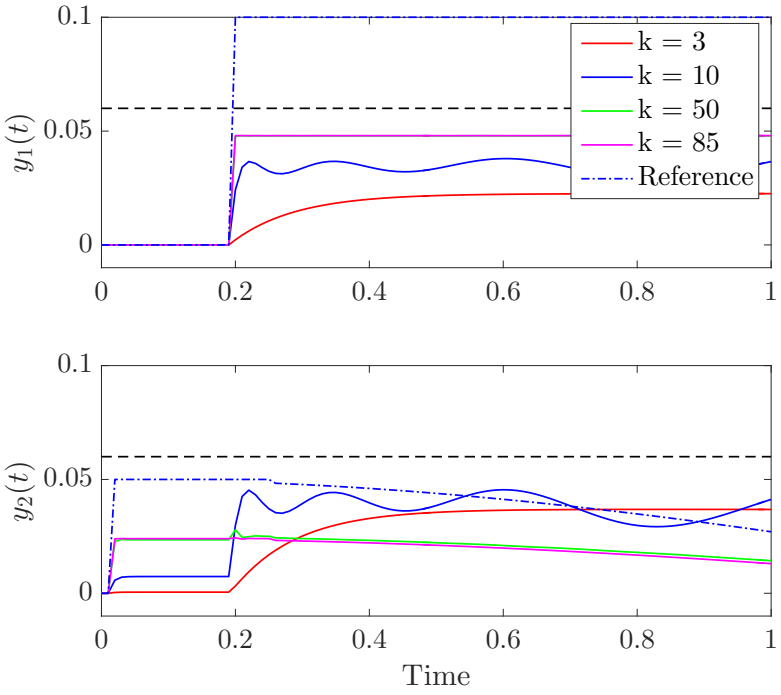

Figure 4.7: The outputs $y_1(t)$ and $y_2(t)$ of the system described in Section 4.3.

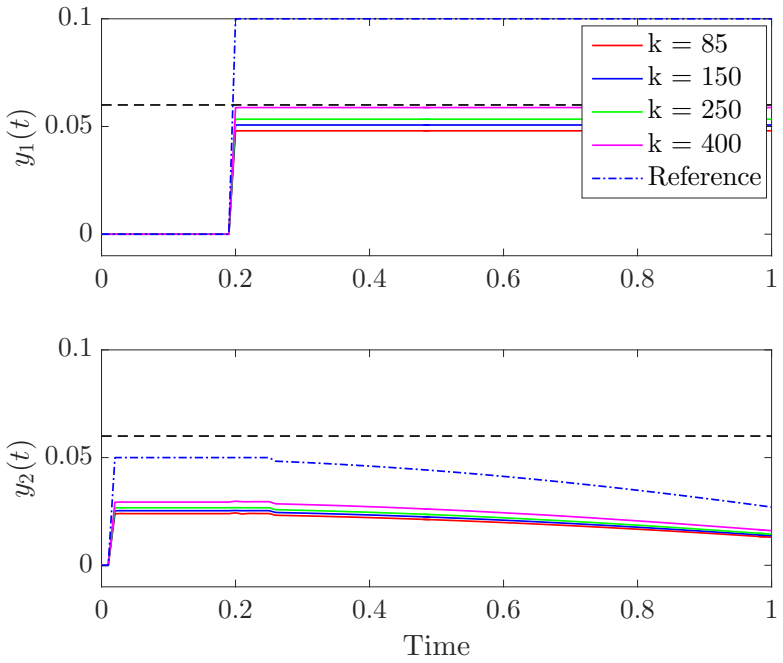

Figure 4.8: The outputs $y_1(t)$ and $y_2(t)$ of the system described in Section 4.3 implemented with the MAS updating law.

same computer described above. The outputs of the system can be seen Figure 4.7, where the reference signals for each output are shown by the blue dashed lines.

This MIMO system was also implemented with the MAS update law proposed in Chapter 3. As can be seen in Figure 4.7, this system has a settling-time of about 85 iterations, so the parameter N was chosen to be 90 iterations. To allow the output to come sufficiently close to the constraint, an α of 0.003 is used. To select the value of ρ the criteria $\rho \leq \frac{\alpha}{\|s\|_\infty}$ is used. This led to $\rho = 0.045$. The outputs $y_1(t)$ and $y_2(t)$ implemented with the IDRG, and the MAS updating law can be seen in Figure 4.8.

Chapter 5

Extensions

In this chapter, we will be exploring a couple of extensions of the IDRG. Specifically, we will be taking a look at the Vector Reference Governor (VRG) and the Command Governor (CG). These RG methods are better suited for systems with multiple inputs and outputs.

5.1 Vector Reference Governor

The VRG is commonly used for multiple-input multiple-output (MIMO) systems, as it has the ability to optimize multiple different λ for multiple different channels. The control signal for the VRG differs from the SRG in that the control signal is selected as

$$v(t) = v(t-1) + \mathbf{\Lambda}(r(t) - v(t-1)) \tag{5.1}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_i)$ and to reiterate, the quadratic program solved at each discrete-time step is

$$\begin{aligned} \underset{\lambda_i \in [0,1]}{\text{minimize}} \quad & \|v(t) - r(t)\| \\ \text{s.t.} \quad & v(t) = v(t-1) + \mathbf{\Lambda}(r(t) - v(t-1) \\ & \Big(x(t),\ v(t)\Big) \in O_\infty \end{aligned} \tag{5.2}$$

To motivate the use of the VRG in the IDRG control method, consider again the iteration domain model of a system controlled with the Arimoto-type ILC controller:

$$\begin{aligned} u_{k+1} &= (I - \gamma H_y)u_k + \gamma r \\ x_k &= H_x u_k, \quad y_k = H_y u_k \end{aligned} \tag{5.3}$$

where $H_x,\ H_y$ are

$$H_y = \begin{bmatrix} CB & 0 & ... & 0 \\ CAB & CB & ... & 0 \\ \vdots & & & \\ CA^{T-1}B & CA^{T-2}B & ... & CB \end{bmatrix},\ H_x = \begin{bmatrix} B & 0 & ... & 0 \\ AB & B & ... & 0 \\ \vdots & & & \\ A^{T-1}B & A^{T-2}B & ... & B \end{bmatrix} \tag{5.4}$$

and $r,\ x_k,\ y_k,\ u_k$ are

$$r = \begin{bmatrix} r(1) \\ \vdots \\ r(T) \end{bmatrix},\ x_k = \begin{bmatrix} x_k(1) \\ \vdots \\ x_k(T) \end{bmatrix},\ y_k = \begin{bmatrix} y_k(1) \\ \vdots \\ y_k(T) \end{bmatrix},\ u_k = \begin{bmatrix} u_k(0) \\ \vdots \\ u_k(T-1) \end{bmatrix} \tag{5.5}$$

The iteration-domain model in (5.3) is essentially a MIMO system, as $y_k,\ u_k$, and x_k are $mT \times 1$, $mT \times 1$, and $nT \times 1$ vectors respectively. Each discrete-time instant in the batch is treated as an input in the IDRG. This way, each input (or discrete-time

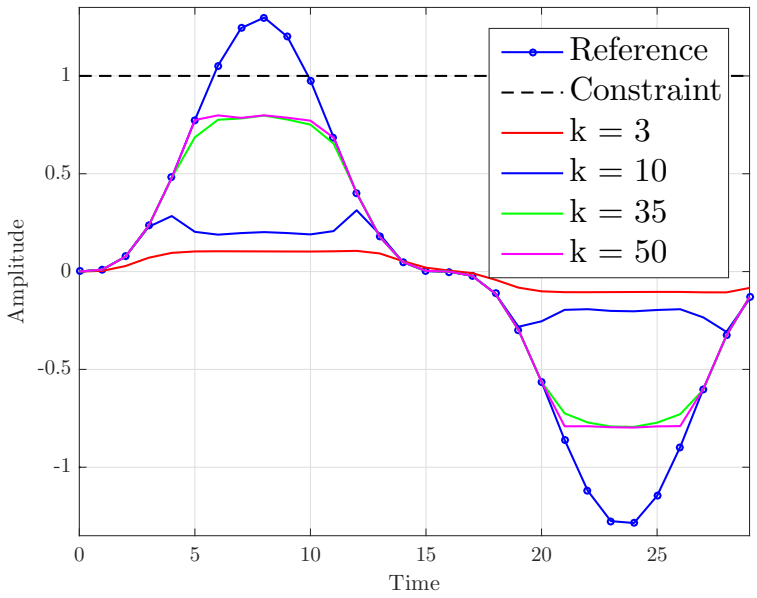

Figure 5.1: The output y_k for various iterations implemented with the VRG.

instant) would have it's own λ_i potentially improving tracking performance.

5.1.1 Revisiting the Original Example

To demonstrate this idea, we will take another look at the example used in Section 4.1. This example is almost exactly the same as above, the only difference being as opposed to being controlled by a SRG, a VRG is used. The simulated output and input of the IDRG controlled system can be seen in Figures 5.1 and 5.2.

Notice, that at each discrete-time instant where the reference signal is inside constraint boundaries, perfect tracking can be achieved, but in places where the reference exceeds constraint boundaries the reference is governed. This is seen around $t = 5$ seconds to $t = 10$ seconds, and around $t = 20$ seconds to $t = 27$ seconds. The same $\beta = 0.8$ is used, so the system is still experiencing over-governing. Tracking

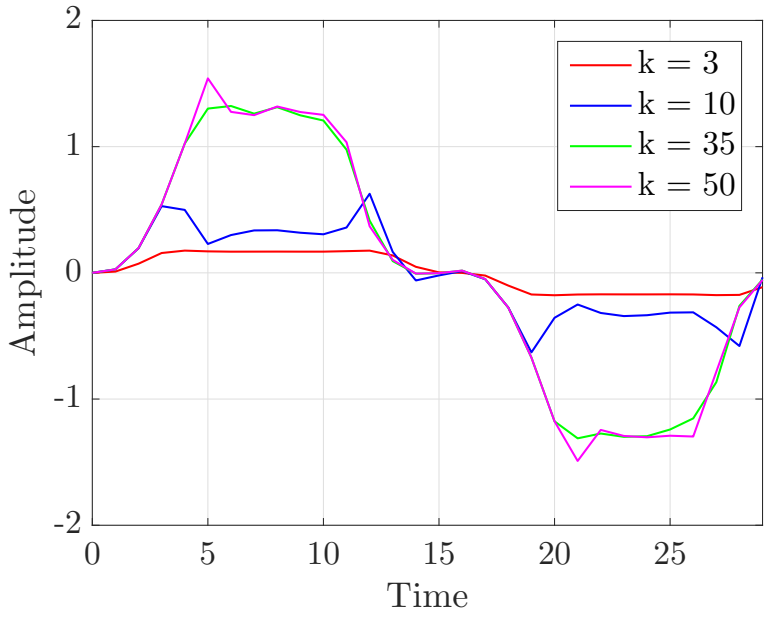

Figure 5.2: The input u_k for various iterations implemented with the VRG.

performance was obviously greatly improved with the use of the VRG, but the mean computation time for $\boldsymbol{\lambda}$ was 0.3076 seconds.

Next, we apply the MAS updating algorithm to the system controlled by the VRG. Similar to previous sections, it was implemented with $\alpha = \rho = \xi = 0.03$, but a larger N is needed. The settling time of the system implemented with the VRG was considerably larger than that of the system implemented with the SRG. As can be seen in Figure 5.1, the transient of the learning response dies out by around 50 iterations, so for this update algorithm to work, $N = 50$ is used. Simulated outputs and inputs with the update law of this system are presented in Figures 5.3 and 5.4. It can be seen that after 275 iterations, the output is within α of the constraint, and MAS updating terminates.

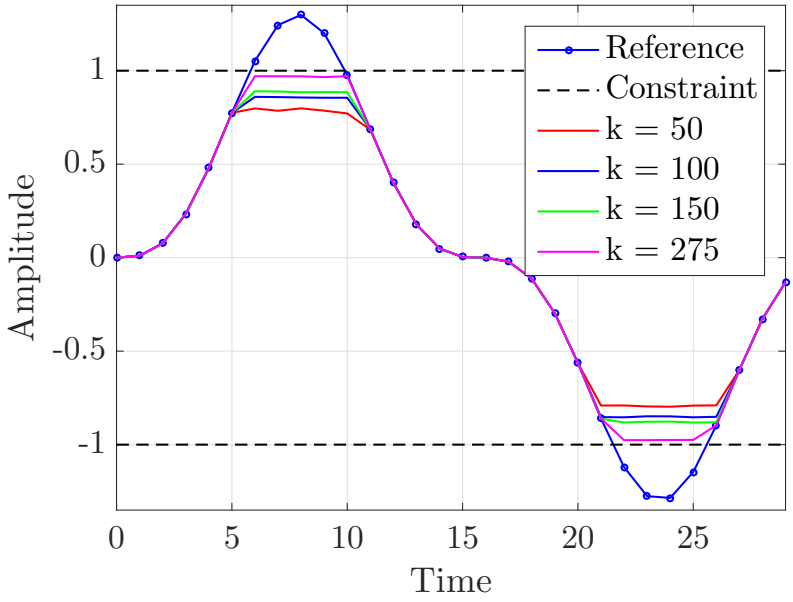

Figure 5.3: The output y_k for various iterations implemented with the VRG and the MAS updating algorithm.

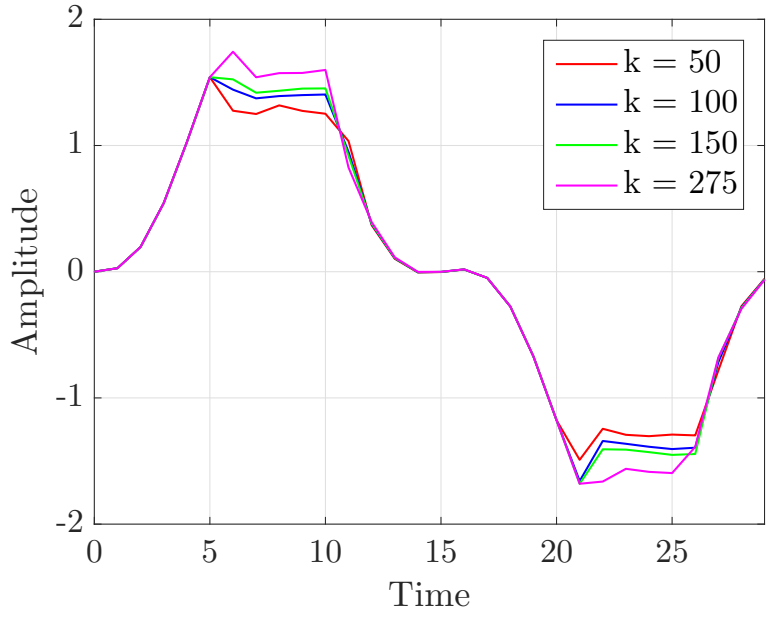

Figure 5.4: The input u_k for various iterations implemented with the VRG and the MAS updating algorithm.

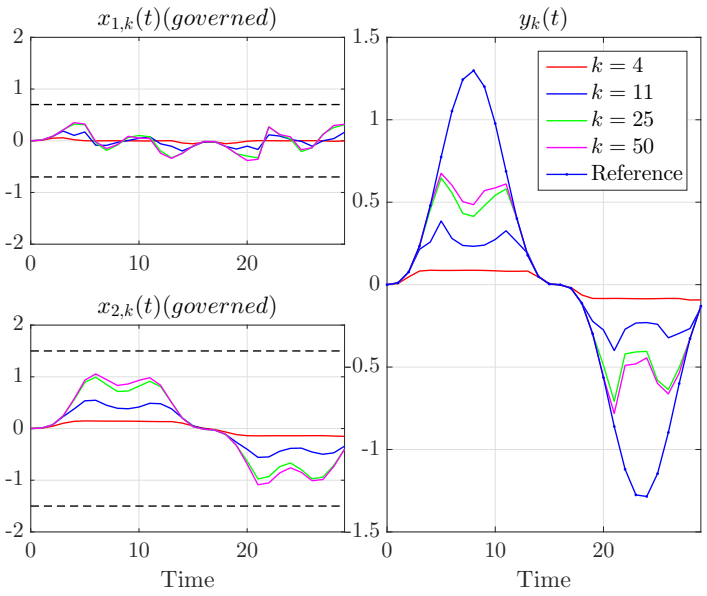

Figure 5.5: The system implemented with a VRG, and constraints on the states.

5.1.2 State Constraints

The VRG controlled IDRG was also implemented with state constraints. Similar to Section 4.2, the same system matrices and learning coefficient are used, and state constraints $-.7 \leq x_{1,k}(t) \leq .7$, and $-1.5 \leq x_{2,k}(t) \leq 1.5$ are imposed. Similar to the previous section, the A, B, and C matrices are used to form the lifted system (and O_∞^{robust}), but the actual matrix A_{actual} is used for simulations, and the learning coefficient is $\gamma = 2$. As can be seen in Figure 5.5, the learning response is slightly different, but the time for the learning dynamics to die out is the same, and overgoverning is experienced.

When the update law was implemented, parameters of the update law, similar to the previous section are $\alpha = \rho = \xi = 0.03$, and $N = 50$. The VRG with state constraints, and the MAS update law are seen in 5.6.

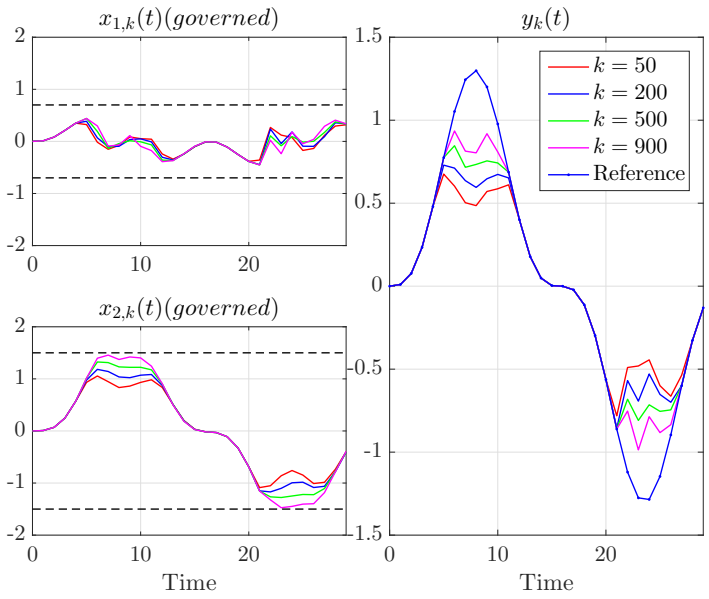

Figure 5.6: The system implemented with a VRG, constraints on the states, and the MAS updating algorithm.

5.2 Command Governor

Similar to the VRG, the Command Governor (CG) provides better, less conservative control response for MIMO systems. The most apparent difference between the VRG and the CG is that instead of using the optimization parameter λ_i, the control signal is optimized directly.

$$\begin{aligned} v(t) = \arg \underset{v}{\text{minimize}} \quad & \|v(t) - r(t)\| \\ \text{s.t.} \quad & \Big(x(t), v(t)\Big) \in O_\infty \end{aligned} \tag{5.6}$$

In other words, each control input at each time instant is treated as an input, that is $v(1), \ldots, v(T)$ are all inputs to the CG.

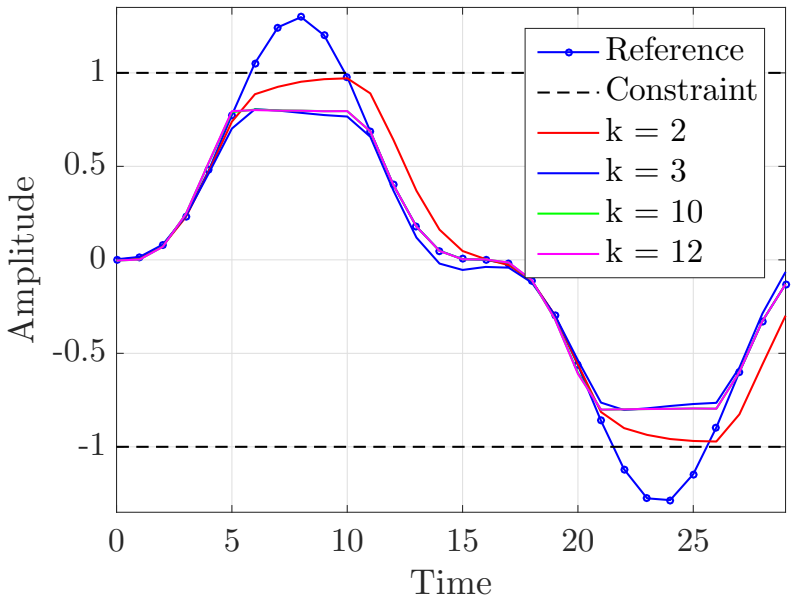

Figure 5.7: The output y_k of the original example implemented with the command governor.

5.2.1 Revisiting the Original Example

Again, we will be looking at the original example, with the state space model seen in 4.1, and system matrices A_{actual}, B, C, and where the O_∞ is formed with the matrices A, B, and C. Results of this example implemented with the CG can seen in Figures 5.7 and 5.8.

As can be seen by these results, the CG provides a convergence time very similar to the SRG results with a convergence time of around 12 iterations. Also, similar to the VRG, tracking within constraint boundaries is drastically improved compared to the SRG. In this simulation, the average computation time for each iteration was 0.2932 seconds, which makes the trade off between tracking ability and computation time very apparent. This is quicker than the VRG, but still drastically slower than

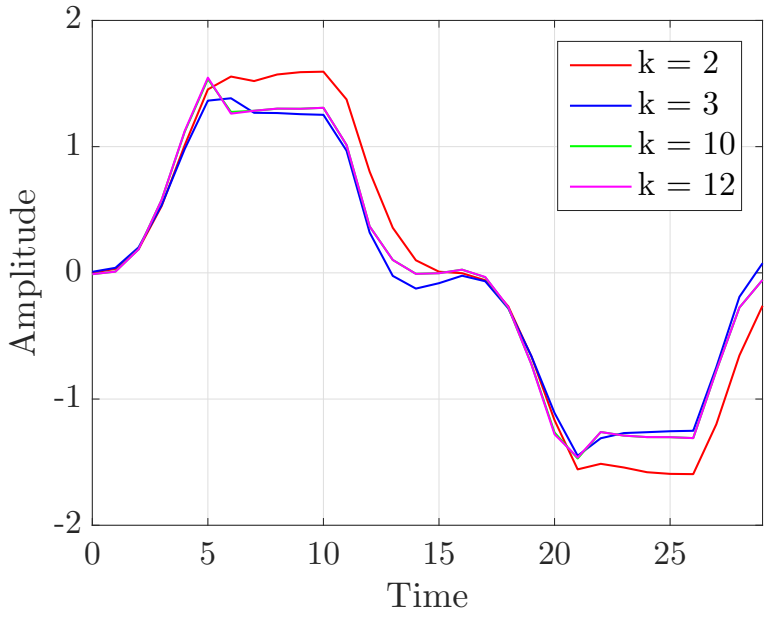

Figure 5.8: The input u_k of the original example implemented with the command governor.

the SRG.

For the sake of continuity, the CG simulation was also implemented with the MAS updating algorithm. Results of this simulation can be seen in Figures 5.9 and 5.10.

As with all of the previous examples with this system, the parameters α, ρ, and ξ are all set to 0.03. Since the convergence time of the CG is considerably quicker than that of the VRG, an N of 15 iterations is used. The total iterations needed to reach the optimal MAS was very similar to that of the SRG. Comparatively, the CG provided better tracking within constraint boundaries, but took longer to compute the control signal.

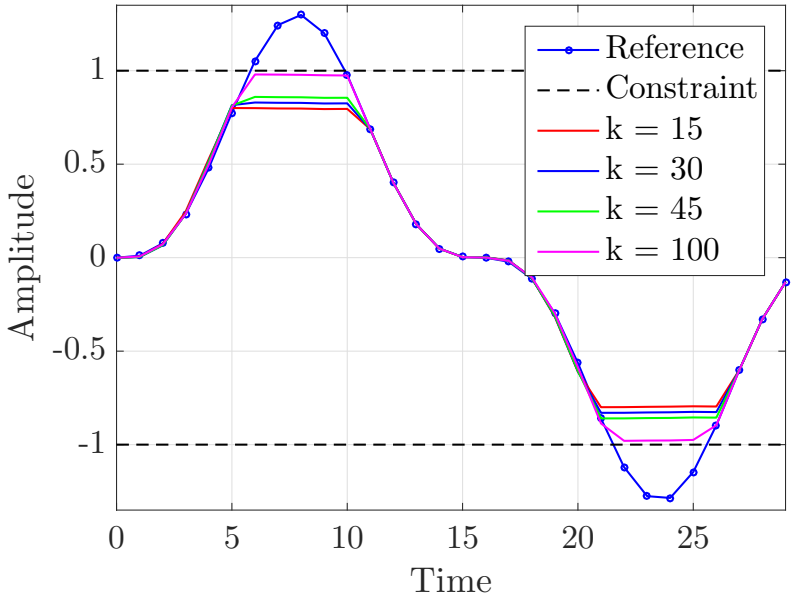

Figure 5.9: The output y_k of the system controlled with the CG, implemented with the MAS updating algorithm.

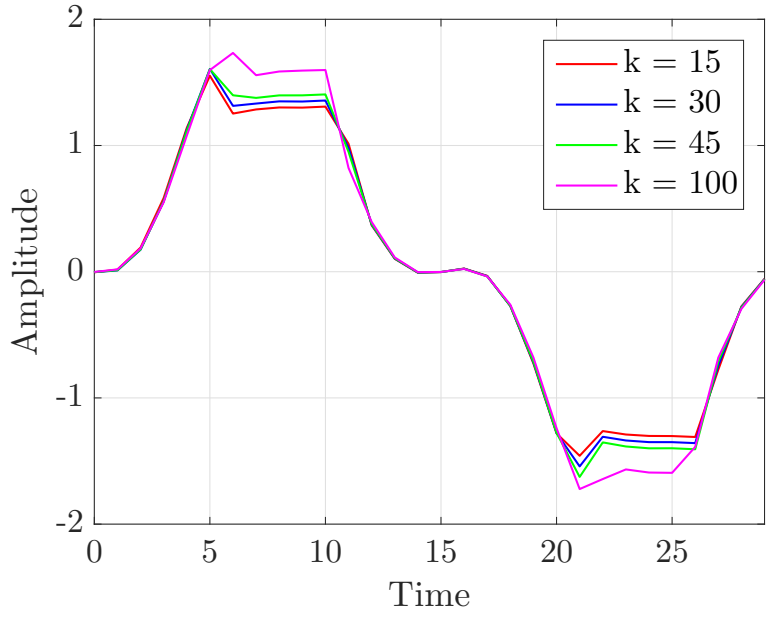

Figure 5.10: The input u_k of the system controlled with the CG, implemented with the MAS updating algorithm.

Chapter 6

Conclusions

6.1 Summary

Today, many modern processes are considered "batch processes", where they are performed repeatedly. Manufacturing processes with robotic arms, hard disk drives, and high speed trains can all be considered batch processes. A popular method of control for these batch processes is Iterative Learning Control (ILC), which, similar to how humans learn, uses information from previous runs or batches of the process to improve tracking performance. Several schemes have been proposed in the literature to tackle the problem of constraint management for ILC, but many of these schemes either do not consider input, output or state constraints, or utilize quadratic or non-linear programming to optimize the control signal.

This thesis provides a novel solution to the issue of constraint management for ILC, by using the Reference Governor (RG). Specifically the Iteration Domain Reference Governor (IDRG) is a modification of a traditional RG, in that it governs the iteration domain dynamics of a system controlled by ILC, as opposed to the time

domain dynamics. The proposed solution can handle constraints imposed on the input, output, and the state, and thanks to the RG, provides a very computationally efficient solution.

This IDRG formulation is endowed with robustness properties through a robust Maximal Admissible Set (MAS). As the algorithm learns, the set is updated to allow better tracking performance within constraint boundaries. Results were applied to different systems, utilizing different forms of the RG: the SRG, the VRG, and the CG.

6.2 Future Work

In this work, there are many different routes to go down for future research. One of the most motivating would be to investigate other ILC learning laws. Similar to classical control (PID), as the ILC is updated, the learning term of the ILC can contain a proportion, derivative, or integral of the error. This thesis considered the D-type, or Arimoto ILC algorithm, but the error term did not contain proportional or integral terms. Extending this theory to contain proportional and integral action in the ILC law would be a great addition to the work, and generalize its applications greatly.

Traditionally, ILC is a "model-free" control technique, meaning that a model of the plant is not needed inside the controller for ILC to be implemented, but a model can be used to design the controller to guarantee convergence properties and such (similar to PID). The RG on the other hand, is model-based. In this thesis, we deal with this discrepancy by implementing a maximal admissible set robust to modeling

uncertainties. Another method to deal with this discrepancy would be to implement a data-driven IDRG, where a MAS would be formed from input and output data alone. This would greatly improve the variety of applications for the IDRG, and put more emphasis on the model-free aspect of ILC.

There is also another common type of learning control called Repetitive Control (RC) [70, 71]. This type of control is also commonly used in batch processes, but is mostly seen used in rotational type systems (like hard disc drives). The main difference between ILC and RC is that ILC systems start from the same initial conditions for each batch, and RC initial conditions are the final conditions of the previous batch. An interesting extension of this work would be to extend it to RC, as it is another type of learning control.

Bibliography

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

[2] 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.

[3] 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.

[4] P. C. Marchal, O. SĂśrnmo, B. Olofsson, A. Robertsson, J. GĂłmez Ortega, and R. Johansson. Iterative learning control for machining with industrial robots. *IFAC Proceedings Volumes*, 47(3):9327 – 9333, 2014. 19th IFAC World Congress.

[5] Jiang Xiaoming, Yu Zhiliang, and Chen Xinglin. Iterative learning control for the synchronization control system of the scanner. In *Proceeding of the 11th World Congress on Intelligent Control and Automation*, pages 2519–2524, 2014.

[6] Hu Yunan and Bin Qu. Application of iterative learning genetic algorithms for pid parameters auto-optimization of missile controller. In *2006 6th World Congress on Intelligent Control and Automation*, volume 1, pages 3435–3439, 2006.

[7] M. K. Cobb, K. Barton, H. Fathy, and C. Vermillion. Iterative learning-based path optimization for repetitive path planning, with application to 3-d crosswind flight of airborne wind energy systems. *IEEE Transactions on Control Systems Technology*, pages 1–13, 2019.

[8] T. Van Pham, D. H. Nguyen, and D. Banjerdpongchai. Design of iterative learning control via alternating direction method of multipliers for building temperature control system. In *2017 14th International Conference on Electrical*

Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), pages 814–817, 2017.

[9] B. S. Zhang and J. R. Leigh. Predictive time-sequence iterative learning control with application to a fermentation process. In *Proceedings of IEEE International Conference on Control and Applications*, pages 439–442 vol.1, 1993.

[10] F. You and J. An. Iterative learning control for batch weighing and feeding process*. In *2018 37th Chinese Control Conference (CCC)*, pages 2904–2908, 2018.

[11] John J Craig. Adaptive control of manipulators through repeated trials. In *American Control Conference*, number 21, pages 1566–1573, 1984.

[12] Giuseppe Casalino. A learning procedure for the control of movements of robotic manipulators. *IASTED Sympo. on Robotics and Automation, 1984*, 1984.

[13] Sadao Kawamura. Iterative learning control for robotic systems. *Proc. of IECON'84, Tokyo*, pages 393–398, 1984.

[14] Paola Bondi, Giuseppe Casalino, and Lucia Gambardella. On the iterative learning control theory for robotic manipulators. *IEEE Journal on Robotics and Automation*, 4(1):14–22, 1988.

[15] KS Lee, SH Bang, and KS Chang. Feedback-assisted iterative learning control based on an inverse process model. *Journal of Process Control*, 4(2):77–89, 1994.

[16] Pasquale Lucibello. Learning control of linear systems. In *1992 American Control Conference*, pages 1888–1892. IEEE, 1992.

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

[18] Won Cheol Kim, In Sik Chin, Kwang Soon Lee, and Jinhoon Choi. Analysis and reduced-order design of quadratic criterion-based iterative learning control using singular value decomposition. *Computers & Chemical Engineering*, 24(8):1815–1819, 2000.

[19] Notker Amann, David H Owens, and Eric Rogers. Iterative learning control for discrete-time systems with exponential rate of convergence. *IEE Proceedings-Control Theory and Applications*, 143(2):217–224, 1996.

[20] Insik Chin, S Joe Qin, Kwang S Lee, and Moonki Cho. A two-stage iterative learning control technique combined with real-time feedback for independent disturbance rejection. *Automatica*, 40(11):1913–1922, 2004.

[21] Kwang S Lee and Jay H Lee. Convergence of constrained model-based predictive control for batch processes. *IEEE Transactions on Automatic Control*, 45(10):1928–1932, 2000.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] T. Meng and W. He. Iterative learning control of a robotic arm experiment platform with input constraint. *IEEE Transactions on Industrial Electronics*, 65(1):664–672, 2018.

[27] S. Mishra, U. Topcu, and M. Tomizuka. Optimization-based constrained iterative learning control. *IEEE Transactions on Control Systems Technology*, 19(6):1613–1621, 2011.

[28] C. Peng, L. Sun, W. Zhang, and M. Tomizuka. Optimization-based constrained iterative learning control with application to building temperature control systems. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 709–715, 2016.

[29] Jian-Xin Xu, Ying Tan, and Tong-Heng Lee. Iterative learning control design based on composite energy function with input saturation. *Automatica*, 40(8):1371–1377, 2004.

[30] Ying Tan, Hao-Hui Dai, Deqing Huang, and Jian-Xin Xu. Unified iterative learning control schemes for nonlinear dynamic systems with nonlinear input uncertainties. *Automatica*, 48(12):3173–3182, 2012.

[31] J. Xu and X. Jin. State-constrained iterative learning control for a class of mimo systems. *IEEE Transactions on Automatic Control*, 58(5):1322–1327, 2013.

[32] Xu Jin and Jian-Xin Xu. A barrier composite energy function approach for robot manipulators under alignment condition with position constraints. *International Journal of Robust and Nonlinear Control*, 24(17):2840–2851, 2014.

[33] G. Sebastian, Y. Tan, D. Oetomo, and I. Mareels. Feedback-based iterative learning design and synthesis with output constraints for robotic manipulators. *IEEE Control Systems Letters*, 2(3):513–518, 2018.

[34] Y. Chen, B. Chu, and C. T. Freeman. Point-to-point iterative learning control with optimal tracking time allocation. *IEEE Transactions on Control Systems Technology*, 26(5):1685–1698, 2018.

[35] C. T. Freeman and Y. Tan. Iterative learning control with mixed constraints for point-to-point tracking. *IEEE Transactions on Control Systems Technology*, 21(3):604–616, 2013.

[36] S. Mishra, U. Topcu, and M. Tomizuka. Iterative learning control with saturation constraints. In *2009 American Control Conference*, pages 943–948, 2009.

[37] Marnix Volckaert, Moritz Diehl, and Jan Swevers. Generalization of norm optimal ilc for nonlinear systems with constraints. *Mechanical Systems and Signal Processing*, 39(1-2):280–296, 2013.

[38] V. E. Hatzikos, J. Hatonen, T. Harte, and D. H. Owens. Robust analysis of a genetic algorithm based optimization method for real-time iterative learning control applications. In *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696)*, volume 2, pages 396–401 vol.2, 2003.

[39] G. Liu and Z. Hou. Rbfnn-based adaptive iterative learning fault-tolerant control for subway trains with actuator faults and speed constraint. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–15, 2019.

[40] G. Sebastian, Z. Li, Y. Tan, and D. Oetomo. On implementation of feedback-based pd-type iterative learning control for robotic manipulators with hard input constraints. In *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pages 43–48, 2019.

[41] Y. Tan, J.X. Xu, M. NorrlĂśf, and C. Freeman. On reference governor in iterative learning control for dynamic systems with input saturation. *Automatica*, 47(11):2412 – 2419, 2011.

[42] Petros Kapasouris, Michael Athans, Günter Stein, et al. Design of feedback control systems for stable plants with saturating actuators. 1988.

[43] Elmer G Gilbert, Ilya Kolmanovsky, and Kok Tin Tan. Nonlinear control of discrete-time linear systems with state and control constraints: A reference governor with global convergence properties. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 1, pages 144–149. IEEE, 1994.

[44] 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 1995.

[45] Alberto Bemporad and Edoardo Mosca. Nonlinear predictive reference filtering for constrained tracking. In *Proc. European Control Conf.*, pages 1720–1725, 1995.

[46] Elmer G Gilbert and Ilya Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 9(15):1117–1141, 1999.

[47] 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 1995.

[48] Alberto Bemporad, Alessandro Casavola, and Edoardo Mosca. Nonlinear control of constrained linear systems via predictive reference management. *IEEE transactions on Automatic Control*, 42(3):340–349, 1997.

[49] Alessandro Casavola, Edoardo Mosca, and David Angeli. Robust command governors for constrained linear systems. *IEEE transactions on Automatic Control*, 45(11):2071–2077, 2000.

[50] David Angeli and Edoardo Mosca. Command governors for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 44(4):816–820, 1999.

[51] 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.

[52] Adam Nathan Banker, Julia Helen Buckland, Joseph Norman Ulrey, Uros Vojko Kalabic, Matthew John Gerhart, Tobias John Pallett, Ilya Kolmanovsky, and Suzanne Kay Wait. Methods and systems for torque control, November 3 2015. US Patent 9,174,637.

[53] Uros Kalabic, Ilya Kolmanovsky, Julia Buckland, and Elmer Gilbert. Reference and extended command governors for control of turbocharged gasoline engines based on linear models. In *2011 IEEE International Conference on Control Applications (CCA)*, pages 319–325. IEEE, 2011.

[54] Uros Kalabic. *Reference governors: Theoretical Extensions and Practical Applications.* PhD thesis, 2015.

[55] Uroš V Kalabić, Julia H Buckland, Stephen L Cooper, Suzanne K Wait, and Ilya V Kolmanovsky. Reference governors for enforcing compressor surge constraints. *IEEE Transactions on Control Systems Technology*, 24(5):1729–1739, 2016.

[56] S-R Oh and Sunil Kumar Agrawal. A reference governor-based controller for a cable robot under input constraints. *IEEE transactions on control systems technology*, 13(4):639–645, 2005.

[57] S Di Cairano, A Goldsmith, and S Bortoff. Model predictive control and spatial governor for multistage processing machines in precision manufacturing. In *Proc. 5th IFAC Nonlinear Model Predictive Control Conference*, pages 3800–3805. Citeseer, 2013.

[58] Sohrab Haghighat, Stefano Di Cairano, Dmytro Konobrytskyi, and Scott Bortoff. Coordinated control of a dual-stage positioning system using constrained model predictive control. In *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2014.

[59] James Wilborn and John Foster. Defining commercial transport loss-of-control: A quantitative approach. In *AIAA atmospheric flight mechanics conference and exhibit*, page 4811, 2004.

[60] Meir Pachter and Russel B Miller. Manual flight control with saturating actuators. *IEEE Control Systems Magazine*, 18(1):10–20, 1998.

[61] Francesco Tedesco and Alessandro Casavola. Fault-tolerant distributed load-/frequency coordination strategies for multi-area power microgrids. *IFAC-PapersOnLine*, 48(21):54–59, 2015.

[62] Alessandro Casavola, Giuseppe Franze, Francesco Tedesco, and Emanuele Garone. Distributed coordination-by-constraint strategies in networked multi-area power systems. In *2011 IEEE International Symposium on Industrial Electronics*, pages 1697–1702. IEEE, 2011.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

[68] 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.

[69] 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.

[70] Shinji Hara, Yutaka Yamamoto, Tohru Omata, and Michio Nakano. Repetitive control system: A new type servo system for periodic exogenous signals. *IEEE Transactions on automatic control*, 33(7):659–668, 1988.

[71] Youqing Wang, Furong Gao, and Francis J Doyle III. Survey on iterative learning control, repetitive control, and run-to-run control. *Journal of Process Control*, 19(10):1589–1600, 2009.