2018

Decoupled Reference Governors for Multi-Input Multi-Output Systems

Yudan Liu
University of Vermont

Decoupled Reference Governors for Multi-Input Multi-Output Systems

A Thesis Presented

by

Yudan Liu

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 Electrical Engineering

October, 2018

Defense Date: July 25th, 2018
Dissertation Examination Committee:

Hamid R. Ossareh, Ph.D., Advisor
Safwan Wshah, Ph.D., Chairperson
Jeff Frolik, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

Abstract

In this work, a computationally efficient solution for constraint management of *square* multi-input multi-output (MIMO) systems is presented. The solution, referred to as the Decoupled Reference Governor (DRG), maintains the highly-attractive computational features of scalar reference governors (SRG) compared to Vector Reference Governor (VRG) and Command Governor (CG). This work focuses on square MIMO systems that already achieve the desired tracking performance. The goal of DRG is to enforce output constraints and simultaneously ensure that the degradation to tracking performance is minimal. DRG is based on decoupling the input-output dynamics of the system so that every channel of the system can be viewed as an independent input-output relationship, followed by the deployment of a bank of scalar reference governors for each decoupled channel. We present a detailed set-theoretic analysis of DRG, which highlights its main characteristics. A quantitative comparison between DRG, SRG, and the VRG is also presented in order to illustrate the computational advantages of DRG. Finally, a distillation process is introduced as an example to illustrate the applicability of DRG.

This thesis is dedicated to my family and friends, for their love and support.

Acknowledgements

First, I would like to thank my advisor, Professor Hamid R. Ossareh, for all his support, guidance, and insight throughout this work.

Next, I wish to thank Professor Safwan Wshah and Professor Jeff Frolik for their interest in my work and serving as members of my thesis committee.

Moreover, I want to thank Joycer Osorio and Weiping Huang for several discussions about this work and their support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, there are two major divisions of control theory, namely classical and modern control. Classical control theory is based on Laplace transforms method, a block diagram of which is shown in Figure 1.1. In this figure, $P(s)$ represents the plant that is mathematically modeled after the physical process, and $C(s)$ is the controller used to manage the dynamic behavior of the plant.

Modern control theory is based on linear algebra and carried out in state space. A block diagram of modern control is shown in Figure 1.2, where A, B, C, D are matrices that represent the behavior of the system, and K is state feedback that is used to regulate the behavior of the system. Over the past hundred years, many effective technologies have been explored to design a system with desired performance in both classical and modern control.

In real systems, there always exist constraints, such as actuator saturation or bounds imposed on process variables (i.e., states or outputs). Moreover, constraints

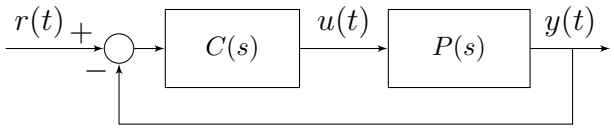

Figure 1.1: Classical control system

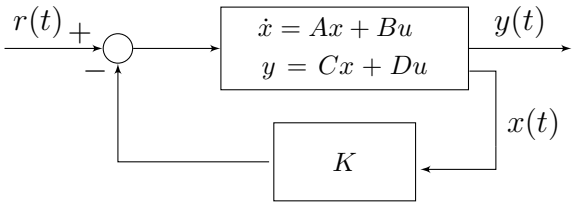

Figure 1.2: Modern control system

may be set due to the limitation of the physical properties of the equipment or to guarantee of a safe system operation. For example, consider a distillation process. The temperature inside the distillation column must be limited to ensure the safety of the process, and the impurity of final products should be bounded to have an efficient system [2]. More details on the distillation process will be presented in Chapter 5. Another example is the turbocharged gasoline engine, where the inputs are throttle and wastegate, and the outputs are turbocharger speed and compressor outlet temperature. The temperature limitation is set to increase the efficiency of the process [3], and the constraint for speed is for hardware protection [4].

Systems can be classified as Single-Input Single-Output (SISO), where the system has one input and one output, or Multi-Input Multi-Output (MIMO), where the number of inputs and the number of outputs are larger than 1. MIMO system are usually coupled in the sense that each output depends dynamically on several inputs. Thus, it is more complex to achieve constraint management for MIMO systems than it is for SISO systems. In this thesis, our focus is on constraint management of *square* MIMO systems, namely, the number of inputs and the number of outputs are equal

and larger than 1. This is to ensure that the input and output behavior of the system can be effectively decoupled, as shown in Chapter 2. Thus, motivated by the need for a safe system, the goal of this thesis is to develop a method to achieve constraint management for square MIMO systems, while maintaining the tracking performance of the system as much as possible.

1.2 Literature Review

In this section, we first review the current literature on constraint management for square MIMO systems. Then, we will introduce literature on the main technique we use in this thesis: the Reference Governor.

1.2.1 Constraint Management for MIMO Systems

There are many choices for engineers to design a desired behavior MIMO system with constraint satisfaction. One route is to redesign the controller and include a Model Predictive Controller (MPC) [5–10] in it. MPC addresses both tracking and constraint management simultaneously. This approach for constraint management in MIMO systems is explored in works like [11–13], where decentralized MPC strategies are proposed. Other MPC solutions are centralized [14], distributed [15], robust [16], and cascade or hierarchical strategies [17]. To briefly summarize, the idea behind MPC is that at time t, the current plant state is estimated and a cost minimizing control strategy is computed for a relatively short time horizon in the future: $[t, t+T]$,

where T is the so-called prediction horizon. However, MPC tends to be computationally demanding, which has limited its applicability, especially for systems with fast dynamics and/or high order. Theoretical guarantees such as stability are also difficult to be obtained in practice.

Another route is to augment a well-designed controller, which already achieves the desired small-signal tracking performance, with constraint handling capability. The tracking part for MIMO system can be achieved by various methods such as the Linear Quadratic Regulator (LQR) [18, 19], Linear Quadratic Gaussian (LQG) [18, 20] sliding mode control [21, 22], *SVD* control [23, 24], $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control [23, 25], and decentralized control methods [26, 27]. Then, the constraint management part is handled by nonlinear functions (e.g., saturation functions) that maintain the constrained signal within the desired bounds. Anti-windup schemes [28, 29] and state feedback-based control methods [30–33] are examples of the second route. However, these approaches either do not consider the coupling with the tracking problem or are difficult to use for design. Currently, a relatively new constraint management technique which alleviates the above technologies' shortcomings, is the Reference Governor (RG), reviewed below.

1.2.2 Reference Governor

Before diving into the details of RG, we will introduce its history. RG was first proposed in the continuous time framework [26]. After that, a natural extension to the discrete time domain has been widely adopted because of the mathematical simplicity [34,35]. A simple block diagram of RG is shown in Figure 1.3. Basically, RG is an add-on scheme for enforcing pointwise-in-time state and control constraints by modifying,

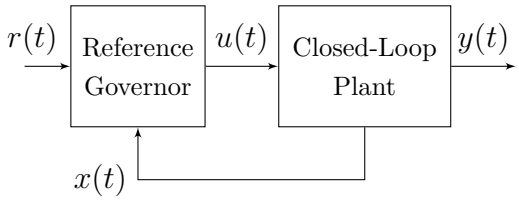

Figure 1.3: Scalar reference governor block diagram. In this figure, $r(t)$, $u(t)$, $y(t)$, and $x(t)$ are the reference, input, constrained output, and state, respectively.

whenever required, the reference to a well-designed stable closed-loop system. In the framework of RG, the *static reference governor* was first introduced by Gilbert in 1994 [35]. Because the static reference governor has the disadvantage of oscillation on the command signal $u(t)$ when constraint violations are detected, it was replaced by *dynamic reference governor* [35]. The formulation of dynamic reference governor includes Scalar Reference Governor (SRG) [36,37], Vector Reference Governor (VRG) [35], and Command Governor (CG) [38]. Later on, CG was extended to Extended Command Governor (ECG) to accelerate the response of the system [39]. In this work, we mainly focus on SRG, because of its lower computational requirements.

Recently, RG was extended to systems with disturbance inputs [40, 41], and stochastic systems with chance constraints. Applications of RG are fuel cells [42, 43], automotive [44, 45], robotics [46], and aerospace [47].

Scalar Reference Governor (SRG)

A block diagram of SRG is shown in Figure 1.3, where $y(t)$ is the constrained output, $r(t)$ is the reference, $u(t)$ is the governed reference, and $x(t)$ is the system state (measured or estimated). To compute $u(t)$, SRG employs the so-called maximal admissible set (MAS) [48], which is defined as the set of all inputs and states that are constraint-admissible and will be explained in detail in Section 2.1. By solving

a simple linear program, SRG selects a $u(t)$ that is as close as possible to $r(t)$ such that the constraints are satisfied for all time. Note that because SRG only provides one scalar adjustable parameter, it works well on single input systems. However, for MIMO systems, SRG may lead to overly conservative response.

Vector Reference Governor (VRG)

Similar to SRG, VRG is also an add-on control scheme to a closed-loop system to achieve constraint management by modifying the reference to the closed-loop system. However, VRG extends the ability of SRG and offers multiple adjustable parameters to allow the governing of different channels. A block diagram of VRG is depicted in Figure 1.4, where $y_1(t), \ldots, y_m(t)$ are the constrained outputs, $r_1(t), \ldots, r_m(t)$ are the references, and $u_1(t), \ldots, u_m(t)$ are the governed references. Similar to SRG, VRG employs MAS to compute $u_i(t), \forall t, \forall i$. However, instead of solving a linear program as is the case for SRG, VRG solves a quadratic program (QP) to find $u_i(t)$ that is as close as possible to $r_i(t)$. Even though VRG shares some properties with SRG, its implementation demands a higher computational load compared to SRG. This is because of the QP that must be solved at each time step, either by implicit methods or multi-parametric explicit methods.

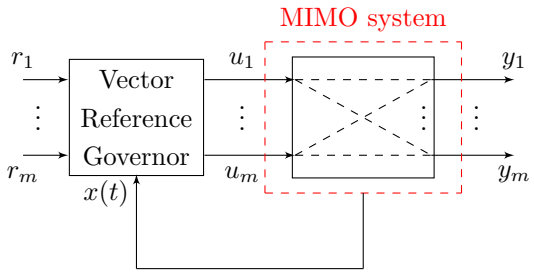

Figure 1.4: Vector reference governor block diagram. The possible couplings between the input-output channels of the plant are shown by dashed lines.

1.2.3 Maximal Output Admissible Set (MAS)

Above, we introduced MAS as a set of all inputs and states that ensures the outputs satisfy the constraints. In this section, we provide a literature review of MAS.

MAS relies on the idea of positive invariant sets. A set $Z \subset \mathcal{R}^n$ is positive invariant if for every initial state $x(0) \in Z$, the subsequent motion, $x(t), t > 0$, belongs to Z. For a complete treatment of invariant sets, see reference [49]. This idea has been used to produce invariant sets that are also output admissible [50–52]. However, these sets may be too conservative in the sense that there may exist much larger output admissible sets. The idea of *maximal* output admissible set was first proposed in 1991 by Gilbert and Tan [48]. After that, more exploration on MAS has been proposed, such as MAS with disturbance inputs [53], MAS for nonlinear systems with constraints [54, 55], MAS with time delay in states and inputs [56, 57], computation of polytopic MAS [58], and MAS for periodic systems [59]. We will explain the computation of MAS in detail in Chapter 2.

1.3 Research Objectives and Contributions

As previous mentioned, SRG is easy to compute, but performs poorly in MIMO systems. While VRG can be used effectively in MIMO systems, it requires more computational effort. Thus, in this work, we present an alternative approach to SRG and VRG for MIMO systems, based on system decoupling, that retains the simplicity of SRG and the performance benefits of VRG.

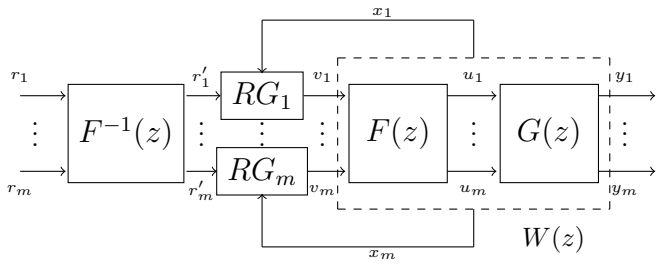

Figure 1.5: Decoupled reference governor block diagram

Consider the discrete-time closed-loop MIMO system $G(z)$ depicted in Figure 1.5, where u_i are the inputs to $G(z)$, and y_i are the constrained outputs. Over the latter, the constraints are imposed: $y_i(t) \in \mathbb{Y}_i, \forall t$, where $\mathbb{Y}_i$ are specified sets. Note that similar to the literature of RG, we consider the system in discrete-time, hence we use *Z-transform* instead of *Laplace transform*. Given desired set-points r_i, the goals are: first to select each u_i as close as possible to r_i to ensure that the degradation to tracking performance is minimal. The second goal is to ensure that the output constraints are satisfied, i.e., $y_i \in \mathbb{Y}_i$. We present two candidate solutions:

First method is as follows: System $G(z)$ is first decoupled by finding a suitable filter, $F(z)$, that will eliminate the coupling dynamics of $G(z)$. The resulting decoupled system is $W(z) := G(z)F(z)$ (as seen in Figure 1.5), which is diagonal; that is, each output y_i depends only on the new input v_i. In this method, all the decoupling operations are performed in the transfer function domain, in order to take an advantage of the simple algebra that transfer functions offer. Next, we introduce a bank of m decoupled scalar RGs, where the goal of the i-th RG is to select v_i as close as possible to r'_i while ensuring $y_i \in \mathbb{Y}_i$. Each scalar RG, RG_i, uses only the states of the i-th decoupled subsystem. Finally, since we would like to ensure that $u_i = r_i$ when constraint violation is not detected, we introduce the inverse of the filter, $F^{-1}(z)$, to

couple the dynamics back. Note that $F^{-1}(z)$ also ensures that u_i and r_i are close if r_i is not constraint admissible. The implementation details, such as observer design, can be found in Chapter 3. Invertibility of $F(z)$ and applicability of DRG are investigated thoroughly in this thesis.

The second method is as follows: similar to the first method, we decouple $G(z)$. However, instead of performing the decoupling process in the transfer function domain, it is handled in the state-space domain by using state-feedback [60]. Then, we introduce a number of m decoupled scalar RGs, where the goal of the i^{th} RG is to select v_i as close as possible to r_i' while ensuring $y_i \in \mathbb{Y}_i$. Each scalar RG, RG_i, uses only the states of the i-th decoupled subsystem. Finally, we need to solve a quadratic program to minimize the gap between $r(t)$ and $u(t)$ because an inverse filter like the one in the first method is not readily available. However, this goes against our goal of maintaining the computational advantages of SRG. Thus, in this work, we will study only the first method, which we refer to as the Decoupled Reference Governor (DRG).

The main contributions of this work are as follows:

- A computationally efficient constraint management technique for square MIMO systems, i.e., the DRG, which is a novel extension of the SRG.
- Steady-state analysis of admissible inputs for DRG in comparison with VRG.
- Analysis of DRG performance with respect to the system singular values, the condition number, and the relative gain array. We show that the proposed approach is appropriate for a specific class of systems and illustrate this by examples.

- Analysis of advantages and disadvantages of DRG with respect to two decoupling approaches.
- Quantitative comparison of explicit and implicit optimization techniques for VRG, SRG, and DRG, where we show that DRG can run at a similar speed compared to SRG, but can be as much as 2500 times faster than VRG.

1.4 Thesis Outline

This thesis is structured as follows: Chapter 2 reviews two decoupling methods, the maximal admissible set, and the SRG and VRG theory. Chapter 3 introduces DRG for square MIMO systems and addresses its applicability. Chapter 4 illustrates the relationship between the MAS of coupled and decoupled systems, and compares DRG, SRG, and VRG in terms of performance and execution time. Chapter 5 presents the simulation results of DRG applied to a practical MIMO system. Chapter 6 concludes this work and discusses topics for future research.

1.5 Notation

The following notations are used. $\mathbb{Z}_+$ denotes the set of all non-negative integers. The identity matrix is denoted by I. Given a vector x, x_i denotes the i-th component of x. If $A \in \mathbb{R}^{n\times m}$, $x \in \mathbb{R}^{m\times 1}$ and $B \in \mathbb{R}^{n\times 1}$ are matrices, $Ax \leq B$ means that the i-th row of the product of A and x is less than or equal to the i-th row of B, for all i. $F(z)$ denotes the $\mathcal{Z}$-transform of function $f(t)$. $F^{-1}(z)$ denotes the inverse of $F(z)$. The norm function is denoted by $\|.\|$. Given a set $\mathbb{M}$, $x \in \mathbb{M}$ means x is an

element of the set $\mathbb{M}$. G^T means the transpose of matrix G. $\sigma_{max}(G)$ denotes the largest singular value of G and $\sigma_{min}(G)$ denotes the smallest singular value of G.

Chapter 2

Review of Preliminaries

In this chapter, we will review the theory behind the maximal output admissible set to have a better understanding of RG. Then, we will explain SRG and VRG. Finally, two decoupling methods and metrics to quantify the applicability of DRG will be introduced.

2.1 Maximal Output Admissible Set (MAS)

In this section, the definition of MAS, denoted by O_∞, will be introduced. Consider a discrete-time system, given in state-space form by:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{2.1}$$

where $t \in \mathbb{Z}^+$ is the discrete time variable, $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input, and $y(t) \in \mathbb{R}^m$ is the constrained output vector. Over the latter, the following constraints are imposed: $y(t) \in \mathbb{Y}$, where $\mathbb{Y}$ is a polytopic set defined by $Sy \leq s$, for given $S \in \mathbb{R}^{q \times m}$ and $s \in \mathbb{R}^q$.

MAS is the set of all safe initial conditions and inputs, defined as:

$$O_\infty := \bigcap_{t=0}^{\infty} P_t \tag{2.2}$$

where the polytopes P_t are defined by:

$$P_t := \{(x_0, u_0) \in \mathbb{R}^{m+m} : x(0) = x_0, u(j) = u_0,\ j = 0, \ldots, t, y(t) \in \mathbb{Y}\} \tag{2.3}$$

Below, we explain how to generate MAS in detail.

Starting from time 0, the constraint for system (2.1) can be expressed as $Sy(j) \leq s$ for all $j \geq 0$. Assuming $u(j)$ is held constant all the time (i.e., $u(j) = u(0) = u, j \geq 0$), then $x(j)$ and $y(j)$ can be written as:

$$x(1) = Ax(0) + Bu, y(0) = Cx(0) + Du,$$

$$x(2) = A^2x(0) + (AB + B)u, y(1) = CAx(0) + (CB + D)u,$$

$$x(3) = A^3x(0) + (A^2B + AB + B)u, y(2) = CA^2x(0) + (CAB + CB + D)u,$$

$$\vdots$$

$y(j)$ is given by:

$$y(j) = A^j x(0) + (C(A^{j-1} + A^{j-2} + \cdots + I)B + D)u,$$

which is equivalent to:

$$y(j) = A^j x(0) + (C(I - A^j)(I - A)^{-1}B + D)u. \tag{2.4}$$

To satisfy $Sy(j) \leq s$, we must have:

$$Sy(j) = SA^j x(0) + S(C(I - A^j)(I - A)^{-1}B + D)u \leq s. \tag{2.5}$$

Clearly, the challenge of the above process is that we need to check infinite inequalities to create O_∞. To overcome this challenge, a finitely determined inner approximation of O_∞ can be obtained by tightening the steady-state constraint and introducing it as a new half-space [35, 61]:

$$P_{ss} := \{(x, u) : G_0 u \in \mathbb{Y}_{ss}\} \tag{2.6}$$

where $G_0 = C(I - A)^{-1}B + D$ is the DC gain of system (2.1), $\mathbb{Y}_{ss} := (1 - \epsilon)\mathbb{Y}$ and $\epsilon \in (0, 1)$. This can also be written as:

$$S(C(I - A)^{-1}B + D)u \leq (1 - \epsilon)s \tag{2.7}$$

With (2.5) and (2.7), there exists a finite time, j^*, such that if (2.5) holds for $0 \leq j \leq j^*$, then it will hold for $j \geq j^*$.

Based on this result, we define $O_\infty^\epsilon \subset O_\infty$ as the set of all initial-state and control pairs that ensure (2.5) holds for all time:

$$O_\infty^\epsilon = \{(x,u) : Sy(\infty) \leq (1-\epsilon)s, Sy(j) \leq s, \forall j\} \tag{2.8}$$

Substituting (2.5) and (2.7) into (2.8), O_∞^ϵ can also be written as:

$$O_\infty^\epsilon = \{(x,u) : H_x x + H_u u \leq h\} \tag{2.9}$$

where

$$H_x = \begin{bmatrix} 0 \\ SC \\ SCA \\ \vdots \\ SCA^{j^*} \end{bmatrix}, H_u = \begin{bmatrix} S(C(I-A)^{-1}B + D) \\ SD \\ S(C(I-A)(I-A)^{-1}B + D) \\ \vdots \\ S(C(I-A^{j^*})(I-A)^{-1}B + D) \end{bmatrix}, h = \begin{bmatrix} (1-\epsilon)s \\ s \\ s \\ \vdots \\ s \end{bmatrix}$$

To summarize, the set O_∞^ϵ can be viewed as a polytope, which characterizes the set $(x(t), u)$ so that constraint management is achieved for all future time.

2.2 Reference Governor

In this section, first, we will declare the assumptions that we made to develop DRG, and then introduce the details of two reference governors, namely Scalar Reference Governor (SRG) and Vector Reference Governor (VRG). Finally, these two reference governors will be compared.

Consider the discrete-time linear system $G(z)$ in Figure 1.5, given in state-space form as (2.1). Recall from the Introduction that we require the outputs to satisfy the constraints $y_i \in \mathbb{Y}_i$, for specified sets $\mathbb{Y}_i$. Note that this requirement can also be expressed as $y \in \mathbb{Y}$, where $\mathbb{Y} = \mathbb{Y}_1 \times \mathbb{Y}_2 \times \cdots \mathbb{Y}_m$ and $\times$ denotes the Cartesian product. The following assumptions are made for the development of the theory presented in this thesis:

Assumption 1. *System* (2.1)*, represented by* $G(z)$ *in Figure 1.5, reflects the combined closed-loop dynamics of the plant with a stabilizing controller. Consequently, it is asymptotically stable. Furthermore, we assume all diagonal subsystems of* $G(z)$ *are also asymptotically stable.*

Assumption 2. *System* (2.1) *is invertible and has a stable inverse, which will be used later in this thesis.*

Note that a necessary condition for Assumption 2 is that system $G(z)$ does not have any non-minimum phase zeros.

Assumption 3. *The constraint sets* $\mathbb{Y}_i$ *are closed intervals of the real line containing the origin in their interiors. This is in agreement with the assumptions commonly made in the literature of reference governors.*

2.2.1 Scalar Reference Governor (SRG)

Since we consider square systems, that is, the number of inputs and the number of outputs are equal, we have that $m = n$. From Section 2.1, it is possible to see that O_∞ contains the predictions of the outputs based on the current states and the inputs.

Based on the predictions, the controller can anticipate if a constraint may be violated and then take corrective actions over the reference. The SRG computes $u(t)$ as:

$$u(t) = u(t-1) + \kappa(r(t) - u(t-1)) \tag{2.10}$$

where κ is a scalar and solved by the following linear program:

$$\begin{aligned} \underset{\kappa \in [0,1]}{\text{maximize}} \quad & \kappa \\ \text{s.t.} \quad & u(t) = u(t-1) + \kappa(r(t) - u(t-1)) \\ & (x(t), u(t)) \in O_\infty \end{aligned}$$

where O_∞ is the MAS discussed before. In the above, $x(t)$, $r(t)$, and $u(t-1)$ are known parameters. Note that (2.10) implies that $u(t)$ lies on the straight line between $u(t-1)$ and $r(t)$ as shown in Figure 2.1. Note that $\kappa = 0$ means that, in order to keep the system safe, $u(t) = u(t-1)$, where $u(t-1)$ is already admissible. Furthermore, $\kappa = 1$ means that no violation is detected and, therefore, $u(t) = r(t)$. This RG formulation ensures system stability and recursive feasibility. For more details, see [35].

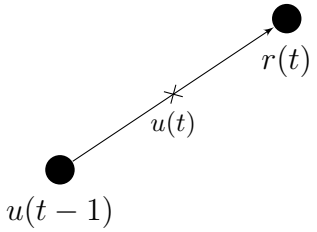

Figure 2.1: The relationship between $u(t)$, $u(t-1)$, and $r(t)$

2.2.2 Vector Reference Governor (VRG)

VRG extends the capabilities of SRG and uses diagonal matrix $\mathbf{K}$ instead of scalar κ. For our proposes, we will assume a square system with $u(t) \in \mathbb{R}^m$ and $y(t) \in \mathbb{R}^m$. Equation (2.10) is reformulated as:

$$u(t) = u(t-1) + \mathbf{K}(r(t) - u(t-1))$$

where $\mathbf{K} = diag(\kappa_i)$. The values of $\kappa_i, i = 1, ..., m$, are chosen by solving the following Quadratic Program (QP):

$$\begin{aligned} \underset{\kappa_i \in [0,1]}{\text{minimize}} \quad & \|u(t) - r(t)\|_Q \\ \text{s.t.} \quad & u(t) = u(t-1) + \mathbf{K}(r(t) - u(t-1)) \\ & (x(t), u(t)) \in O_\infty \end{aligned}$$

where $Q = Q^\top > 0$. Note that for VRG, $O_\infty \subset \mathbb{R}^{n+m}$ can be computed in the same way as explained in Section 2.1. Because of the increased number of optimization variables and the QP formulation, VRG is more computationally demanding than SRG. In this work, we compare SRG and VRG with the proposed DRG method in terms of computational efficiency (i.e., execution time) and performance (i.e., closeness of u and r, as well as constraint satisfaction).

2.2.3 Comparison between SRG and VRG

In this section, we will illustrate with an example that SRG performs poorly in MIMO systems because of the use of a single decision variable (i.e., κ).

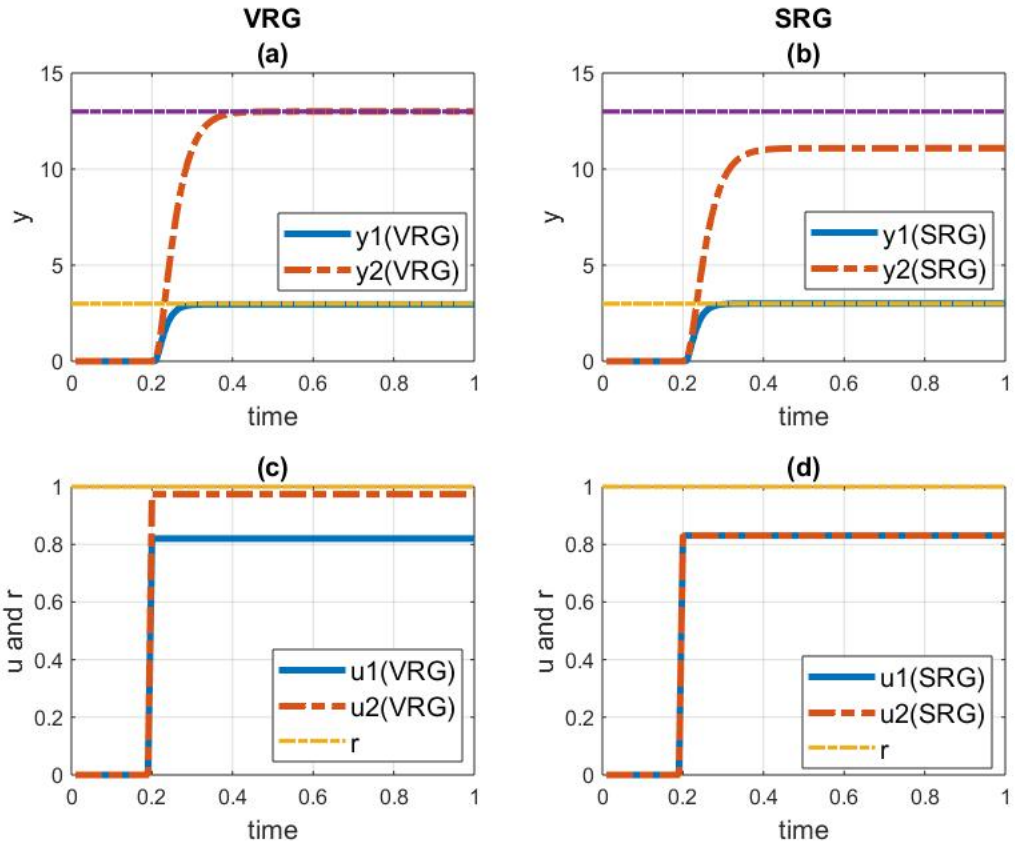

Figure 2.2: Comparison between SRG and VRG. The left-hand plots (i.e., (a) and (c)) represent simulation results for VRG. The right-hand plots (i.e., (b) and (d)) represent simulation results for SRG. In the top two plot (i.e., (a) and (b)), the dashed purple and yellow lines are the output constraints

Consider a two-input two-output system given by:

$$G(z) = \begin{bmatrix} \frac{0.9}{(z-0.5)^2} & \frac{0.05}{3z+1} \\ \frac{0.06}{2z+1} & \frac{1.2}{(z-0.7)^2} \end{bmatrix}$$

The constraint for the first output is $y_1 \leq 3$, and the constraint for the second output is $y_2 \leq 13$. The reference signals are unit step inputs.

As shown in Figure 2.2 (a) and (b), the outputs for VRG and SRG all satisfy the

constraints. However, from Figure 2.2 (c) and (d), it is clear that the gap between u and r for VRG is smaller than that of SRG, which means the degradation to tracking performance for VRG is smaller than that of SRG. Note that, for SRG, because r_1 and r_2 are equal, the single decision variable, κ, leads to u_1 equals to u_2.

From this example, we can see that for both VRG and SRG, the constraint are enforced as desired. However, VRG minimizes the gap between v and r, while the SRG can not because of the single decision variable.

2.3 Review of Decoupling Methods

Decoupling methods can be used to produce partially decoupled or completely decoupled systems [62]. In this section, diagonal decoupling method, identity decoupling method and metrics to quantify the applicability of DRG are reviewed.

2.3.1 Decoupling methods

Consider the square coupled system $G(z)$ shown in Figure 1.5, where

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \underbrace{\begin{bmatrix} G_{11}(z) & \dots & G_{1m}(z) \\ \vdots & \ddots & \vdots \\ G_{m1}(z) & \dots & G_{mm}(z) \end{bmatrix}}_{G(z)} \begin{bmatrix} u_1 \\ \vdots \\ u_m \end{bmatrix} \tag{2.11}$$

The system $G(z)$ consists of diagonal subsystems with dynamics $G_{ii}(z)$ and off-diagonal (interaction) subsystems with dynamics $G_{ij}(z), i \neq j$. Clearly, each output may be influenced by multiple inputs through different channels, which leads to the

coupled behavior. A decoupled system is perfectly diagonal, which means that each system output can be independently controlled by a corresponding system input.

As shown in Figure 1.5, we decouple the system by adding a filter, $F(z)$, before $G(z)$, so that the product $G(z) \times F(z)$ yields a diagonal transfer function matrix $W(z) := G(z) \times F(z)$ [62]. By doing so, each output y_i depends only on the new input v_i through: $y_i = W_{ii}(z)v_i$, where $W_{ii}(z)$ is the i^{th} diagonal element of $W(z)$. We make the assumption that $W_{ii}(z)$ are chosen to be stable and invertible with stable inverses.

We present the following two decoupling methods [62]:

Diagonal method

We find $F(z)$ such that:

$$W(z) = \begin{bmatrix} G_{11}(z) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & G_{mm}(z) \end{bmatrix}$$

The filter is defined as:

$$F(z) = G(z)^{-1} \times W(z) \tag{2.12}$$

Identity method

We find $F(z)$ such that $W(z)$ equals the identity matrix. The filter is defined as:

$$F(z) = G(z)^{-1} \tag{2.13}$$

Notice that in both methods, either $F(z)$ or $F^{-1}(z)$ or both of them may be improper transfer functions because the inversion of $G(z)$ might cause the order of the denominator be smaller than the order of the numerator. If this is the case, they cannot be implemented. In order to make them proper, we multiply them by time-delays of the form $\frac{1}{z^{\beta}}$, where $\beta \in \mathbb{R}$ refers to how much time delay should be added to make the transfer functions proper. Note that if delays are added to either $F(z)$ or $F^{-1}(z)$, the system response will be delayed under the DRG scheme, even if no constraint violation is detected. This is a caveat of the DRG approach; however, if the sample time is small, the introduced delay would be negligible. Also note that $G(z)^{-1}$ might introduce unstable poles to $F(z)$, which will cause the system to become unstable. Assumption (2) is introduced in this work to avoid such situations.

2.3.2 Relative Gain Array

In this section, we will introduce Relative Gain Array (RGA) [63] to quantify the level of interaction of the internal dynamics of a MIMO system. RGA can be defined in two ways, either as a function of frequency or at steady state. We adopt the latter [1,64]:

$$\mathrm{RGA}(G) = G_0 \circ (G_0^{-1})^T \tag{2.14}$$

where $\circ$ denotes element-by-element multiplication (i.e., Hadamard product), and G_0 is the DC gain of system G. In general, large off-diagonal RGA elements (i.e., larger than 10) indicate that the system is highly coupled. To be more specific, if the ij^{th} element of the RGA is large, then y_j is highly influenced by u_i through subsystem G_{ij}. On the other hand, a system with small off-diagonal RGA elements (i.e., smaller

than 10) is weakly coupled.

2.3.3 Condition Number

In this section, a measurement called the condition number that is used to quantify the sensitivity of systems is reviewed [65]. In this work, we use condition number to quantify the applicability of DRG.

Condition number shows how much the output would change due to small changes in input. Mathematically, the condition number for a matrix can be defined as the ratio between the maximum and minimum singular values:

$$\gamma(G) = \frac{\sigma_{max}(G)}{\sigma_{min}(G)}$$

where σ_{max} refers to the largest singular value and σ_{min} refers to the smallest singular value.

A matrix with large condition number is said to be ill-conditioned, which means it is sensitive to small changes in inputs and its inverse "almost" does not exist. On the other hand, a matrix with small condition number is said to be well-conditioned.

The relationship between RGA and condition number is that large RGA elements indicate a system with large condition number. However, the inverse may not hold [66]. The DRG is suitable for systems with small condition numbers, which will be analyzed in later chapters.

In this chapter, we reviewed the concept of MAS. Then we explained the detail of SRG and VRG. Later on, we introduced diagonal and identity decoupling methods to decouple a system. Finally, RGA, as a measure to quantify the interaction of

subsystems, and condition number, which is used to quantify DRG, were reviewed.

Chapter 3

Decoupled Reference Governor

As mentioned in the Introduction, the Decoupled Reference Governor (DRG) is based on decoupling the closed-loop system using the filter F, and then implementing m independent scalar reference governors for the resulting decoupled subsystems, and coupling the system back together using the inverse filter, F^{-1} (see Figure 1.5). Below, we elaborate on these ideas. We consider the two decoupling methods (diagonal decoupling method and identity decoupling method) studied in the previous chapter separately. Without loss of generality, we assume that the initial conditions on G, F, and F^{-1} are all zero, i.e., the entire system starts from rest. Note that the development is presented in the discrete transfer function domain. Once the system is decoupled, the decoupled system is transformed into state space form for DRG implementation, as discussed below.

Consider the system $G(z)$ in (2.11). For this system, we apply the decoupling

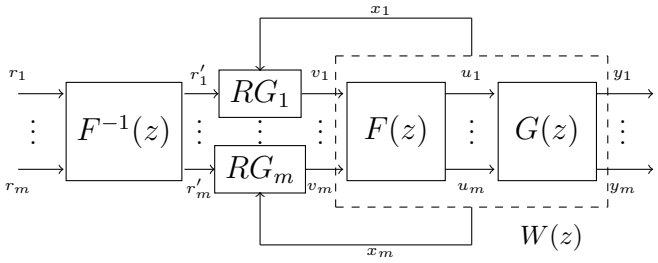

Figure 3.1: Decoupled reference governor block diagram. This figure has been explained in Chapter 2. We reintroduced it again for clarification.

techniques to obtain a completely diagonal system $W(z)$, where $W(z)$ is defined as:

$$W(z) = \begin{bmatrix} W_{11}(z) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & W_{mm}(z) \end{bmatrix}$$

Next, for each decoupled subsystem, W_{ii}, we compute the maximal admissible set (MAS) separately, denoted by $O^W_{i,i}$. To obtain these sets, we convert each subsystem W_{ii} to state space form and, for each, compute the MAS as:

$$\begin{aligned} O^W_{i,i} :=& \{(x_{i_0}, v_{i_0}) \in \mathbb{R}^{n_i+1} : x_i(0) = x_{i_0}, v_i(t) = v_{i_0}, \\ & y_i(t) \in \mathbb{Y}_i, \forall t \in \mathbb{Z}_+\} \end{aligned} \tag{3.1}$$

where x_i and n_i are the state and the order of the i^{th} subsystem, respectively. In this work, it is assumed that the states of G are known. If this is not the case, an observer can be designed if measurements are available. We will introduce how to design the observer later. The issue of observer error dynamics or plant/model mismatch are not investigated here and are subject of study in future work.

The DRG formulation is based on $O^W_{i,i}$. Specifically, the inputs to the diagonal

decoupled system (see Figure 3.1) are defined by:

$$v_i(t) = v_i(t-1) + \kappa_i(r_i'(t) - v_i(t-1)) \tag{3.2}$$

where κ_i are computed by m independent linear programs:

$$\begin{aligned} \underset{\kappa_i \in [0,1]}{\text{maximize}} \quad & \kappa_{\text{i}} \\ \text{s.t.} \quad & v_i(t) = v_i(t-1) + \kappa_i(r_i'(t) - v_i(t-1)) \\ & (x_i(t), v_i(t)) \in O_{i,i}^W \end{aligned} \tag{3.3}$$

Note that, since $F(z)$ and $F^{-1}(z)$ are both assumed to be stable, the DRG formulation above inherits the stability and recursive feasibility properties of scalar RGs.

Below, we specialize the DRG formulation to the two decoupling methods presented in Chapter 2. We show that DRG is suitable for a system with relatively small condition number. On the contrary, when G has large condition number, then plant inputs $u_i(t)$ (see Figure 3.1) may be far from the references $r_i(t)$, and therefore tracking performance for the closed-loop system may deteriorate.

3.1 Diagonal Method

For this case, we implement the diagonal method to obtain a completely decoupled system:

$$W(z) = \begin{bmatrix} W_{11}(z) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & W_{mm}(z) \end{bmatrix}$$

One challenge of this method is that if the states of $G(z)$ are not known, then how can the states of $W(z)$ that feed back to the RGs be found? To answer this, we can design an observer to estimate the states of $G(z)$. Note that the state-space form of $G(z)$ in this work is a special one and is not a minimal realization. To elaborate, assume that the state-space forms of subsystems $G_{11}(z)$, $G_{12}(z)$, $\cdots$, $G_{mm}(z)$ are $(A_{11},B_{11},C_{11},D_{11})$, $(A_{12},B_{12},C_{12},D_{12})$, $\ldots$, $(A_{mm},B_{mm},C_{mm},D_{mm})$.

Then, system $G(z)$ can be written as:

$$\begin{bmatrix} x_{11}(t+1) \\ x_{12}(t+1) \\ \vdots \\ x_{mm}(t+1) \end{bmatrix} = \underbrace{\begin{bmatrix} A_{11} & 0 & \ldots & 0 \\ 0 & A_{12} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & A_{mm} \end{bmatrix}}_{A} \begin{bmatrix} x_{11}(t) \\ x_{12}(t) \\ \vdots \\ x_{mm}(t) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{11} & 0 & \ldots & 0 \\ 0 & B_{12} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & B_{mm} \end{bmatrix}}_{B} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_m(t) \end{bmatrix} = \underbrace{\left[\begin{array}{cccc:c:ccc} C_{11} & C_{12} & \ldots & C_{1m} & 0 & \multicolumn{3}{c}{0} \\ \hdashline \multicolumn{4}{c:}{0} & \ddots & \multicolumn{3}{c}{0} \\ \hdashline \multicolumn{4}{c:}{0} & 0 & C_{m1} & C_{m2} & \ldots \quad C_{mm} \end{array}\right]}_{C} \begin{bmatrix} x_{11}(t) \\ x_{12}(t) \\ \vdots \\ x_{mm}(t) \end{bmatrix} + \underbrace{\begin{bmatrix} D_{11} & D_{12} & \ldots & D_{1m} \\ D_{21} & D_{22} & \ldots & D_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ D_{m1} & D_{m2} & \ldots & D_{mm} \end{bmatrix}}_{D} \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix} \tag{3.5}$$

Note that the dimension of the state vector is $\sum_i \sum_j n_{ij}$, where n_{ij} is the state dimension of G_{ij}. The reason we use this realization is that it is easy to find the states that feed back to RGs after decoupling, which will be clarified after the following example: suppose $G(z)$ is a two-input two-output system. For G_{11}: $x_{11}(t+1) = 0.6x_{11}(t) + u_1(t), y_{11}(t) = 0.4x_{11}(t)$. For G_{12}: $x_{12}(t+1) = 0.6x_{12}(t) + u_2(t), y_{12}(t) = 0.3x_{12}(t)$. For G_{21}: $x_{21}(t+1) = 0.6x_{21}(t) + u_1(t), y_{21}(t) = 0.5x_{21}(t)$. For G_{22}: $x_{22}(t+1) = 0.6x_{22}(t) + u_2(t), y_{22}(t) = 0.1x_{22}(t)$. Then, based on (3.4) and (3.5), the state-space form of $G(z)$ is:

$$A = \begin{bmatrix} 0.6 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 0.6 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0.4 & 0.3 & 0 & 0 \\ 0 & 0 & 0.5 & 0.1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

After using A, B, C, D to design an observer for system $G(z)$, we can get the estimated states: $[\hat{x}_{11}, \hat{x}_{12}, \ldots, \hat{x}_{mm}]^T$ as shown in Figure 3.2. Note that because the state space realization is detectable (because of Assumption 2), it is always possible to design an observer for $G(z)$. The state that feeds back to RG_i is $\hat{x}_{ii}$. The caveat of this approach is if the responses of unobservable (detectable) eigenvalues of $G(z)$ are slow, the observer might converge slowly. However, if the initial condition of the observer is close to the initial condition of $G(z)$, then the effect of the slow eigenvalues can be ignored. In this work, we assume all the states are known, so we do not pursue observer design further. Note that DRG can be viewed as "semi closed-loop" because decoupling is done in open loop but RGs need feedback of the states. In the following,

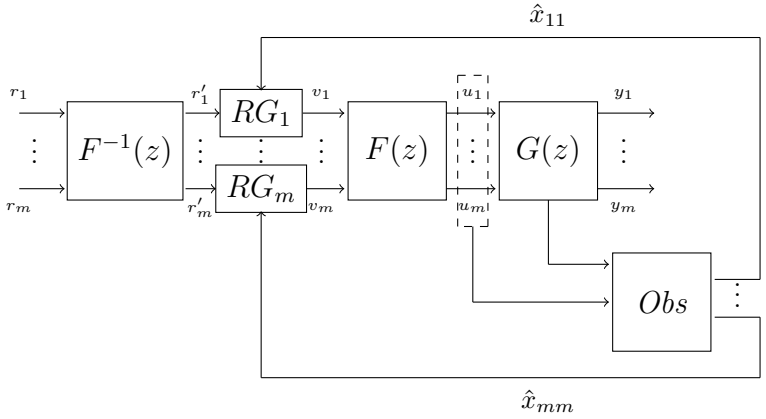

Figure 3.2: Decoupled reference governor with observer (obs)

we will introduce an example to illustrate the results of the diagonal method and show that DRG works well for systems with small condition number.

Consider system $G(z)$ in (2.11) given by:

$$G(z) = \begin{bmatrix} \frac{0.9}{(z-0.2)^2} & \frac{q}{3z+1} \\ \frac{3}{(2z-1)^2} & \frac{0.4}{z-0.6} \end{bmatrix}$$

The decoupled system is given by:

$$W(z) = \frac{1}{z} \begin{bmatrix} \frac{0.9}{(z-0.2)^2} & 0 \\ 0 & \frac{0.4}{z-0.6} \end{bmatrix}$$

In order to show that this method works well for systems with small condition number, we select two different q's: $q = 0.5$ and $q = 0.05$. If $q = 0.5$, the condition number for the system is 11.54. If $q = 0.05$, the condition number for the system is 8.6. The second case has smaller condition number than the first.

Next, we use (2.12) to find $F(z)$. Noticing that in this example, we encounter

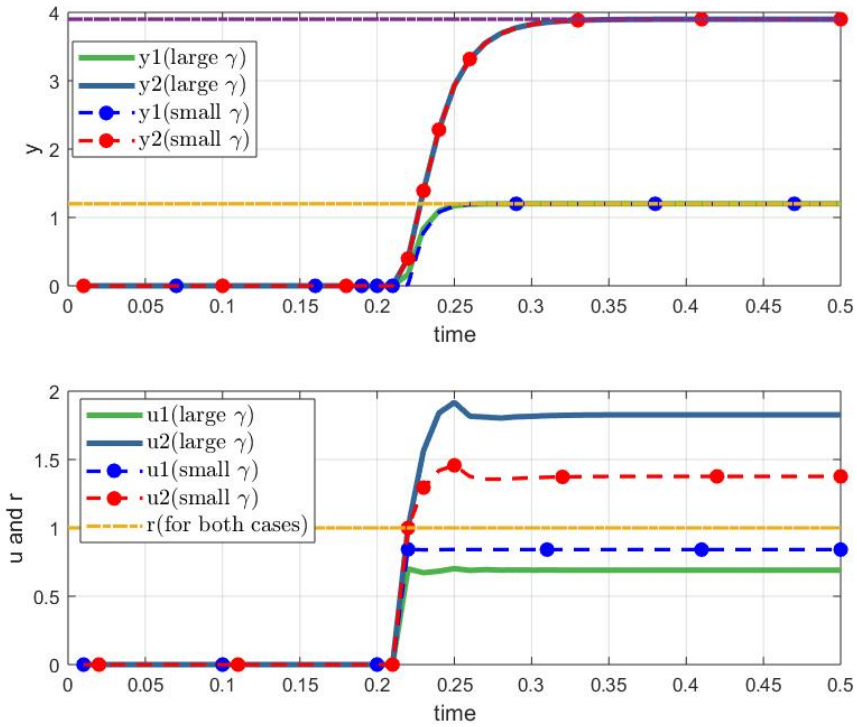

Figure 3.3: Comparison of DRG with small and large condition number system (γ). Top plot is the output (constraints shown by dashed lines) and the bottom plot is the reference $r(t)$ and the plant input $u(t)$

the situation that $F(z)$ and $F^{-1}(z)$ are not proper, we multiply them by z^{-1}. Finally, we obtain the decoupled system.

We proceed to design the DRG based on $W(z)$. In this example, we obtain $O^W_{1,1}$ and $O^W_{2,2}$ based on (3.1). The DC-gain of $W(z)$ is:

$$\begin{bmatrix} 1.40 & 0 \\ 0 & 1 \end{bmatrix}$$

So, if there is no RGs, the outputs for unit step inputs are 1.4 and 4.0, respec-

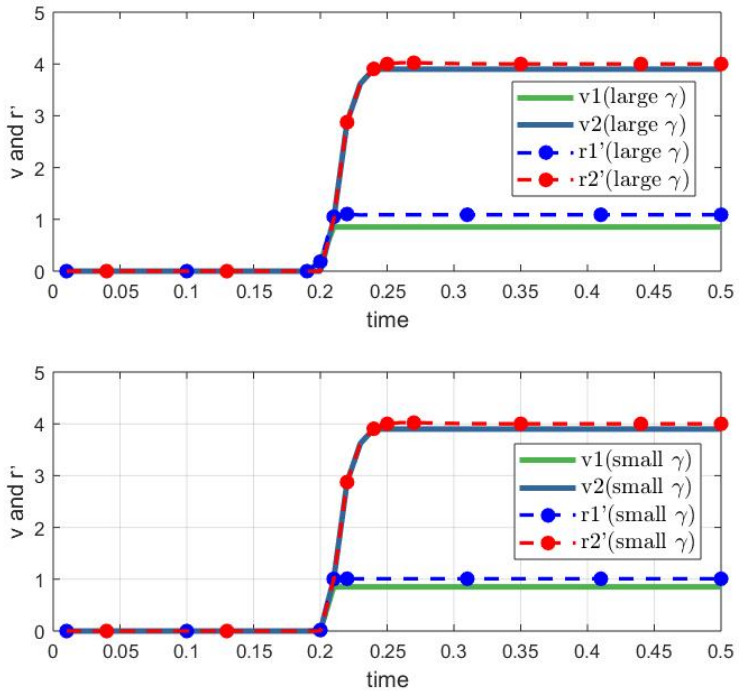

Figure 3.4: Comparison of $r'(t)$ and $v(t)$ in DRG with large (top plot) and small (bottom plot) condition number systems (γ).

tively. Thus, we define the constraint set as $\mathbb{Y} := \{(y_1, y_2) : y_1 \leq 1.2, y_2 \leq 3.9\}$ to ensure both RGs are active at steady state. We simulate the response of this system to a step of size 1 in both r_1 and r_2. The simulation results for both $q = 0.5$ and $q = 0.05$ are depicted in Figure 3.3 and 3.4.

From the results of Figure 3.3, the outputs in both cases satisfy the constraints (dashed purple line and dashed yellow line), as required. However, the system input $u(t)$ is much closer to $r(t)$ for the system with smaller condition number (i.e., for $q = 0.05$), which is desirable for tracking.

Furthermore, Figure 3.4 shows the comparison between the inputs and the outputs of RGs. It can be seen from Figure 3.4 that $v(t)$ is always below $r'(t)$, which is

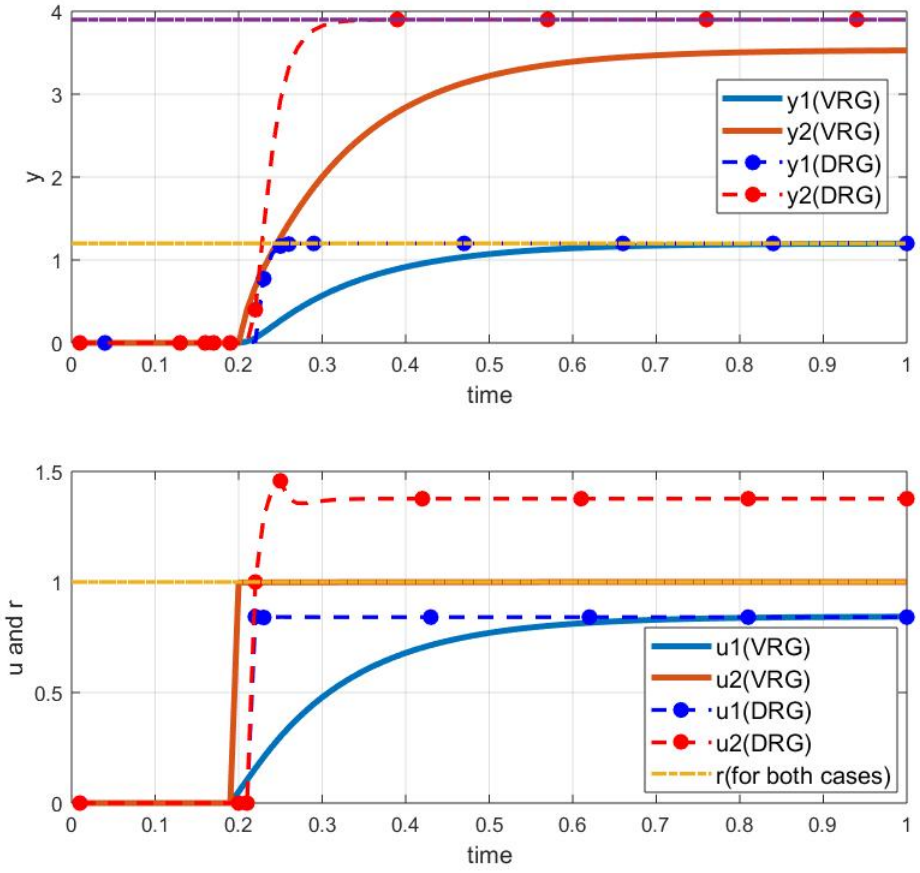

Figure 3.5: Comparison of VRG and DRG for the small condition number system. Top plot is the output. Bottom plot is u compared with r.

a feature of scalar RGs. However, note from Figure 3.3, that $u(t)$ may be above or below $r(t)$. The reason can be explained by the relation $u = F(z)v$ (see Figure 3.1): at steady state, u converges to $F_0 v$, where F_0 is the DC gain of F. Therefore, u may converge to a larger or smaller value than r depending on F_0. This discussion helps to understand the results that will be presented in the sequel.

Figure 3.5 shows a comparison between VRG and DRG for $q = 0.05$. There is a time delay between the responses of VRG and DRG that is caused by the delay

added to F and F^{-1} to make them proper. Note that the rise time for DRG is much faster than that of VRG in this example. This is because the interacting dynamics are slow and dominant, which causes the VRG to generate slow inputs. The DRG, on the other hand, operates on the decoupled system where these slow dynamics have been canceled. This shows that, in addition to computational advantages, the DRG may also have performance advantages compared to VRG.

3.2 Identity Method

As previously mentioned, for the identity method, $W(z)$ is either the identity matrix (if $G^{-1}(z)$ is proper) or the identity matrix with time delay (if $G^{-1}(z)$ is not proper). In other words, the input-output behavior of the i^{th} channel is given by $y_i(t) = v_i(t-\beta)$, where $\beta \in \mathbb{Z}_+$ is the delay added to make $G^{-1}(z)$ proper. An interesting observation can be made: the MAS for a pure delay system is independent of the state, which means that the m independent RGs do no need feedback of the states:

$$O_{i,i}^{W} = \{(x_0, v_0) : v_0 \in \mathbb{Y}_i\}.$$

The above follows directly from the definition of O_∞ in (2.2), (2.3) and by noting that the initial states (i.e., outputs) of the time-delay can be chosen as 0, which is automatically admissible. Note also that, MAS for this case is finitely determined, without the need to tighten the steady-state constraint.

The DRG formulation for the case of identity method is the same as (3.2), (3.3). However, the implementation is greatly simplified due to the structure of $O_{i,i}^{W}$. To see this, note that the formulation of (3.2), (3.3) together with the above structure

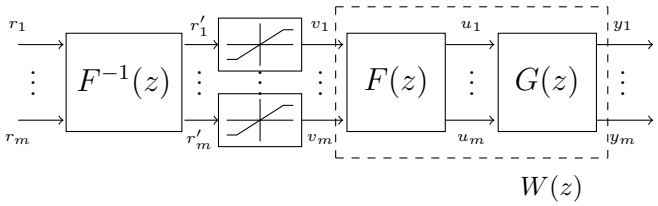

Figure 3.6: Decoupled reference governor with identity method block diagram

of $O^W_{i,i}$ imply that κ_i in (3.3) is chosen so that $v_i(t) \in \mathbb{Y}_i$. Since $\mathbb{Y}_i$ is an interval (per Assumption 3), this implies that κ_i is selected so that $v_i(t)$ is simply clipped (i.e., saturated) at the constraint. Thus, the overall DRG can be implemented as a bank of m decoupled saturation functions as shown in Figure 3.6, which greatly simplifies real-time implementation. In this case, DRG can be viewed as a purely open loop constraint management strategy.

Similar to the diagonal method, if $G(z)$ has large condition number, the inputs to $G(z)$ would be far away from the references and, hence, the tracking performance may suffer. To illustrate, consider the same system $G(z)$ presented in Section 3.1. We use (2.13) to find $F(z)$. Noticing that in this example, we also encounter the situation that $F(z)$ and $F^{-1}(z)$ are not proper, we multiply them by z^{-1}. Finally, we obtain the decoupled system:

$$W(z) = \frac{1}{z}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Similar to the first case, the constraint set is defined as $\mathbb{Y} := \{(y_1, y_2) : y_1 \leq 1.2, y_2 \leq 3.9\}$. We simulate the response of this system to a step of size 1 in both r_1 and r_2. The simulation results for both $q = 0.5$ and $q = 0.05$ are shown in Figure 3.7.

From the results of Figure 3.7, the outputs in both cases satisfy the constraints,

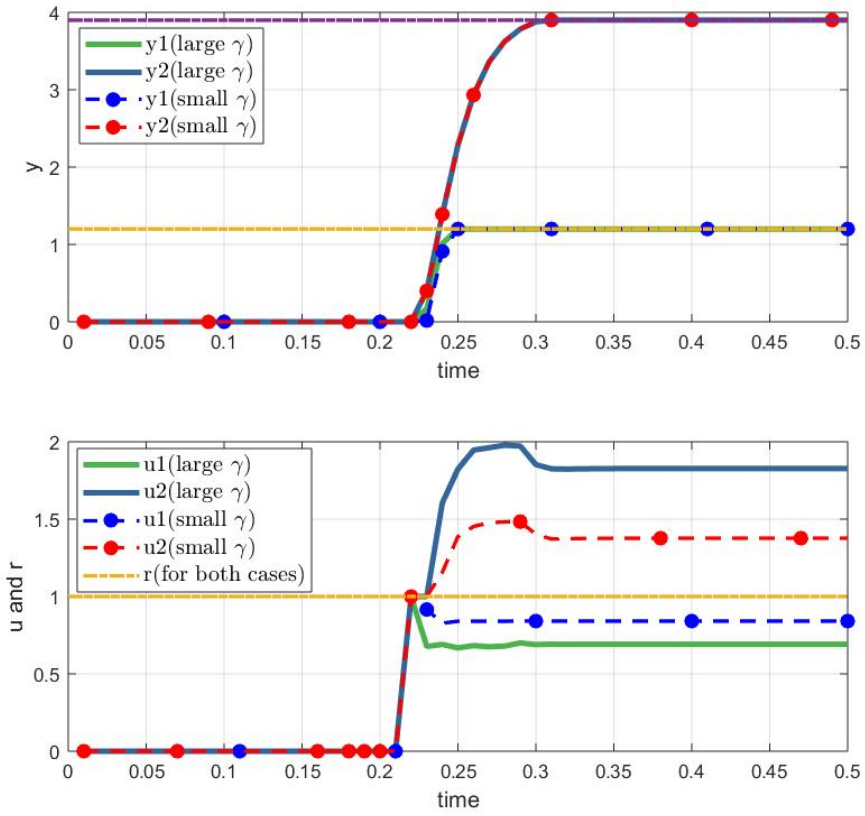

Figure 3.7: Comparison of DRG with small and large condition number system (γ) for the identity method. Top plot is the output (constraints shown by dashed lines) and the bottom plot is reference $r(t)$ and plant input $u(t)$.

as required. However, the system input $u(t)$ is much closer to $r(t)$ for the system with smaller condition number (i.e., for $q = 0.05$), which is desirable.

Furthermore, it can be seen from Figure 3.8 that $v(t)$ is always below $r'(t)$, which is a feature of scalar RGs. However, note from Figure 3.7, that $u(t)$ may be above or below $r(t)$. The reason is similar to the one presented for the diagonal method.

While the identity method is simpler and computationally superior to the diagonal method, it has a drawback. If system $G(z)$ has under-damped dynamics, then

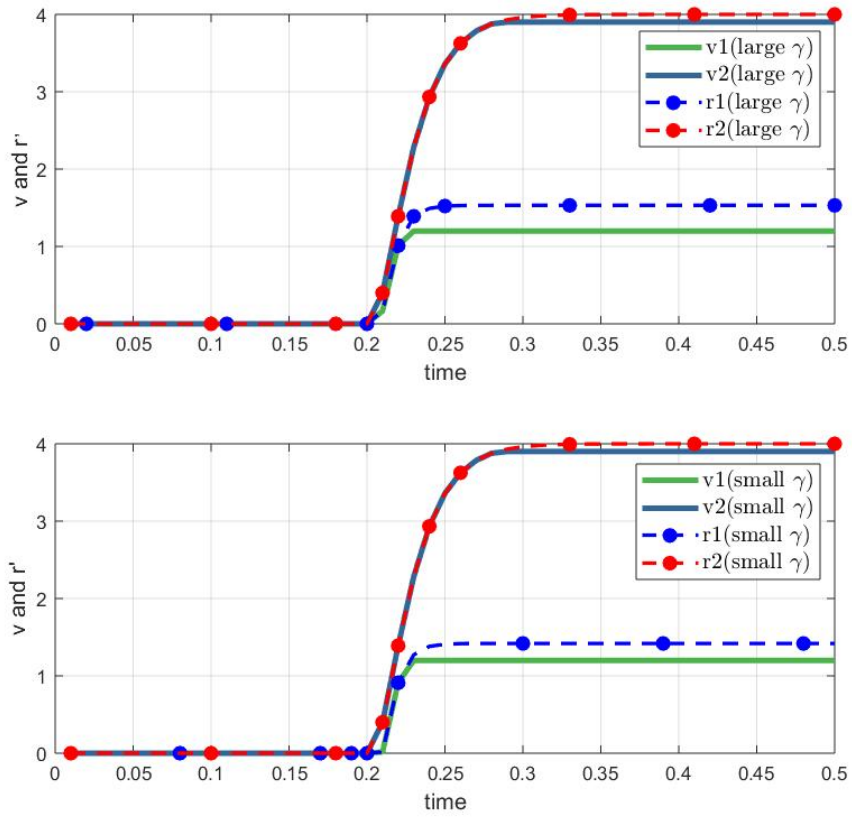

Figure 3.8: Comparison of $r'(t)$ and $v(t)$ in DRG with large (top plot) and small (bottom plot) condition number system (γ).

this method would cause large oscillation in the output.

To illustrate, we select $q = 0.05$ in the example of Section 3.1 and change $G_{11}(z)$ in $G(z)$ to:

$$G_{11}(z) = \frac{0.54z - 0.49}{z^2 - 1.85z + 0.9}$$

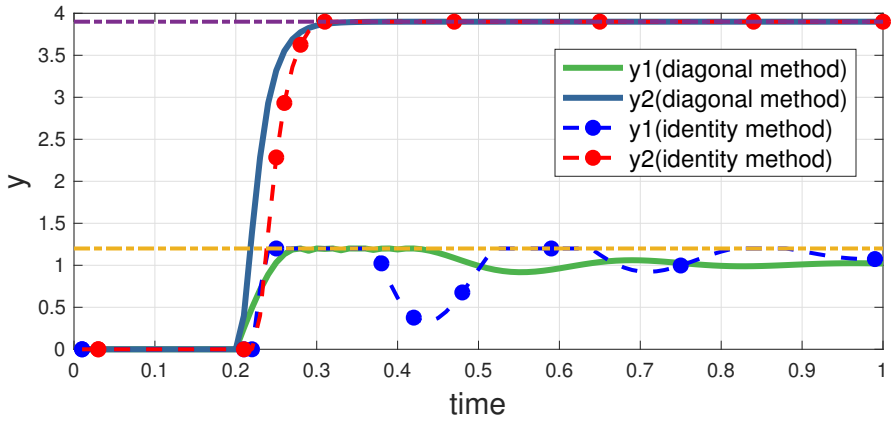

Figure 3.9: Comparison of outputs between diagonal method and identity method

Then, the closed-loop system $G(z)$ becomes:

$$G(z) = \begin{bmatrix} \frac{0.54z-0.49}{z^2-1.85z+0.9} & \frac{0.05}{3z+1} \\ \frac{3}{(2z-1)^2} & \frac{0.4}{z-0.6} \end{bmatrix}$$

A comparison between the outputs of this system after applying DRG with the diagonal and identity methods is shown in Figure 3.9. It can be seen that the constraints are satisfied for both outputs. However, unlike the diagonal method, the output using the identity method has large oscillations. The reason for this behavior can be explained as follows. Because $G(z)$ has slow under-damped dynamics, and since $F^{-1}(z) = G(z)$ for the identity method, applying a step to $r(t)$ causes oscillatory response in $r'(t)$. Viewing DRG as saturations in this case, $v(t)$ is computed as $r'(t)$ clipped at the constraints. Finally, since $W(z)$ is an identity matrix or identity matrix with some time delay, these oscillations will directly show up at the output $y(t)$.

In this chapter, we introduced diagonal DRG and identity DRG with two exam-

ples to show their performance. The results show that they both work well on small condition number systems. Moreover, through one example, we illustrated that the identity DRG may cause oscillations for systems that have under-damped behavior.

Chapter 4

Analysis of Decoupled Reference Governors

In this section, we analyze the structure of O_∞ and the behavior of DRG at steady-state. This analysis sheds light on some of the important features of the proposed method. Then, we provide a comparative analysis of the computation time of DRG compared to SRG and VRG.

4.1 Steady-State Analysis

Recall that DRG requires $O^W_{i,i}$ to be computed separately for each channel. The steady-state halfspace in $O^W_{i,i}$ can be defined similarly to (2.6). In order to study the steady-state admissible inputs, we consider the projection of the steady-state halfspace onto the v_i coordinate, which results in:

$$V^W_{i,i} := \{v_i \in \mathbb{R} : W_{ii_0} v_i \in \mathbb{Y}_{i,ss}\} \tag{4.1}$$

where $W_{ii_0} \in \mathbb{R}$ is the DC gain of subsystem W_{ii} and $\mathbb{Y}_{i,ss} = (1-\epsilon)\mathbb{Y}_i$ (recall that $\mathbb{Y}_i$ is the constraint set for y_i). Since W is diagonal, it follows that the steady-state constraint-admissible input set for W is:

$$V_{ss}^W := V_{1,1}^W \times V_{2,2}^W \times \cdots \times V_{m,m}^W$$

We now compare the above set with the steady-state constraint-admissible input set of system G, which arises in VRG applications. This set, noted by U_{ss}, is defined by:

$$U_{ss} := \{u \in \mathbb{R}^m : G_0 u \in \mathbb{Y}_{ss}\}. \tag{4.2}$$

where G_0 is the DC gain of system G (see Fig. 1.5).

From the above, the following theorem emerges.

Theorem 1. *For the system of Figure 1.5, and U_{ss} and V_{ss}^W defined in* (4.1) *and* (4.2)*, the following relation holds*

$$V_{ss}^W = F_0^{-1} \times U_{ss}, \tag{4.3}$$

where F_0 is the DC-gain of $F(z)$ and the operation $F_0^{-1} \times U_{ss}$ is the point-by-point mapping of the set U_{ss} through matrix F_0^{-1}.

Proof. From (2.12) the following relationship follows $W_0 = G_0 \times F_0$. By using definitions (4.2) and (4.1), the proof follows. □

An important implication of this theorem is as follows. If r is not steady-state admissible with respect to system G (i.e., $r \notin U_{ss}$), then, after feeding through F_0^{-1}, r' must also not be steady-state admissible with respect to the system W (i.e., $r' \notin V_{ss}^W$).

The sets (4.1) and (4.2) describe the steady-state operations of DRG and VRG, respectively. Note that VRG solves a QP whereas DRG solves an LP. This implies that, at steady-state, DRG finds a solution on a vertex of V_{ss}^W, or from Theorem 1, a vertex of U_{ss}. On the other hand, VRG finds a solution that may or may not be at a vertex of U_{ss}. Therefore, DRG leads to a suboptimal solution with respect to the objective function of VRG. This will be illustrated in the next section.

As previously mentioned, two requirements for successful implementation of DRG are that the plant input, u_i, and the setpoint, r_i, should be equal if no constraint violation is detected, and that they should be as close as possible if constraint violation is detected. This is to ensure that the degradation of tracking performance is minimal. We note that each scalar RG in Figure 1.5 ensures that v_i and r_i' are equal if no constraint violation is detected and close if constraint violation is detected; however, u and r may be far. In the following theorem, we show that, at steady state, the closeness of u and r and, hence, the performance of DRG, depends on the decoupling filter, $F(z)$.

Theorem 2. *Given the system of Figure 1.5, at steady-state, we have that:*

$$\|F_0^{-1}\|^{-1}\|v - r'\| \leq \|u - r\| \leq \|F_0\|\|v - r'\|$$

where $\|\cdot\|$ refers to the induced matrix norm.

Proof. Since, at steady state, $u = F_0 v$ and $r = F_0 r'$, we have that: $\|u - r\| = \|F_0 v - F_0 r'\| = \|F_0(v - r')\| \leq \|F_0\|\|v - r'\|$. This proves the right hand inequality. To show the left hand inequality, write $\|v - r'\| = \|F_0^{-1}u - F_0^{-1}r\| = \|F_0^{-1}(u - r)\| \leq \|F_0^{-1}\|\|u - r\|$. This can be re-written as $\|F_0^{-1}\|^{-1}\|v - r'\| \leq \|u - r\|$, which concludes

the proof. □

This theorem shows that the difference between r and u is upper bounded by the difference between v and r' scaled by the induced norm of F_0 and lower bounded by the difference between v and r' scaled by the norm of F_0^{-1}, which are known a-priori. As previously mentioned, RG would guarantee that the gap between v and r' is as small as possible; therefore, if $\|F_0\|$ is small, then small $\|v-r'\|$ implies small $\|u-r\|$, which is desirable. Also, if $\|F_0^{-1}\|^{-1}$ is large, then small $\|v-r'\|$ implies large $\|u-r\|$, which is undesirable. However, in the case of large $\|F_0\|$ or small $\|F_0^{-1}\|^{-1}$, no definite conclusion can be made.

Note that if the 2-norm (i.e., $\|.\|_2$) is chosen, then $\|F_0\| = \sigma_{\max}$, and $\sigma_{\max}$ is the largest singular value of F_0. Similarly, $\|F_0^{-1}\|^{-1} = \sigma_{\min}$. Therefore,

$$\sigma_{\min}\|v - r'\|_2 \leq \|u - r\|_2 \leq \sigma_{\max}\|v - r'\|_2.$$

Since $\|u - r\|_2$ is exactly the objective function in VRG optimization, the above shows that the performance of DRG and VRG will be close if F_0 has small singular values.

Finally, note that if the identity decoupling method is implemented, then $F = G^{-1}$. Hence, using Theorem 2, the following relation follows:

$$\|G_0\|^{-1}\|v - r'\| \leq \|u - r\| \leq \|G_0^{-1}\|\|v - r'\|.$$

Similar to the above, if the 2-norm is used, then $\|G_0\| = \sigma_{\max}(G_0)$, and

$\|G_0^{-1}\|^{-1} = \sigma_{\min}(G_0)$. Therefore:

$$\frac{1}{\sigma_{\max}(G_0)}\|v - r'\|_2 \leq \|u - r\|_2 \leq \frac{1}{\sigma_{\min}(G_0)}\|v - r'\|_2.$$

This relationship shows that if the largest singular value of G_0 is small, then small $\|v - r'\|$ would lead to large $\|u - r\|$, which is undesirable. Meanwhile, if the smallest singular value of G_0 is large, then small $\|v - r'\|$ implies small $\|u - r\|$, which is desirable. In this case, if we want the performance of DRG and VRG to be close, large singular value of G_0 is needed.

4.2 Computation Time of DRG, SRG, and VRG

In this section, we present the computation of DRG and compare it with SRG and VRG. All simulations were performed in Matlab. The simulation device is a Macbook with 1.1 GHz Intel Core m3 processor and 8 GB memory.

The formulation of SRG requires one single solution of one linear program (LP), which can be solved implicitly via online LP solvers, or explicitly as explained below. At the same time, DRG formulation requires the solution to m LPs. VRG, on the other hand, requires the solution to a Quadratic Program (QP), which can be solved implicitly via online optimization or explicitly via multi-parametric programming. Explicit QP is an off-line optimization which divides the state space into several regions and finds the optimal solutions explicitly as a function of the states [67]. Implicit QP or LP is an on-line optimization that solves the problem iteratively at

each time step. In this work, we employed several toolboxes to implement explicit QP, implicit QP and implicit LP: MPT toolbox from Matlab [68], Quadprog function from Matlab, and Gurobi. Finally, we chose MPT because it resulted in the smallest run time. As for explicit DRG, we implemented Algorithm 1 below.

To introduce this algorithm, let us assume $O^W_{i,i}$ is finitely determined and the t^{th} row of H_x and H_v is defined as: $H_x := CA^t x$, $H_v := (C(I - A^t)(I - A)^{-1}B + D)v$. Let j^* be the number of rows of H_x, H_v, h.

Algorithm 1 Custom Explicit DRG Algorithm

let $a = H_v(r(t) - v(t-1))$
let $b = h - H_x x(t) - H_v v(t-1)$
set $\kappa = 1$
for $i = 1$ to j^* **do**
    if $a(i) > 0$ **then**
        $\kappa = \min(\kappa, b(i)/a(i))$
    end if
end for
$\kappa = \max(\kappa, 0)$

Table 4.1: Computation time for SRG in the example of Chapter 5

	Implicit LP	**Algorithm 1**
average	0.21×10^{-2}s	0.16×10^{-7}s
maximum	0.79×10^{-2}s	0.11×10^{-5}s

Table 4.2: Computation time for DRG in the example of Chapter 5

	Implicit LP	**Algorithm 1**
average	0.49×10^{-2}s	5.2×10^{-7}s
maximum	2.2×10^{-2}s	1.42×10^{-5}s

Table 4.3: Computation time for VRG in the example of Chapter 5

	Implicit QP	**Explicit QP**
average	0.45×10^{-2}s	0.13×10^{-2}s
maximum	2.30×10^{-2}s	1.96×10^{-2}s

The explicit SRG is implemented using an algorithm similar to Algorithm 1. We simulate the system in the example of Chapter 5 using 6 different governor and solver combinations: explicit SRG (i.e., Algorithm 1), implicit SRG (i.e., implicit LP), explicit DRG (i.e., Algorithm 1), implicit DRG (i.e., implicit LP), explicit VRG (i.e., mutli-parametric QP), and implicit VRG (i.e., implicit QP). The simulation length is 10000 time steps in all cases with a sample time of 0.01s; for more details, see Chapter 5. Upon simulating the system, we compute the average and maximum computation times of the solvers. In order to eliminate the effects of background processes running on the computer, each of the above experiments is run eight times and we compute the averages value of last five times. The results are shown in Table 4.1, Table 4.2 and Table 4.3. As can be seen, both the average time and maximum time indicate that the Explicit DRG runs almost 2500 times faster than VRG. Furthermore, SRG computation terminates faster than DRG, and DRG computation terminates faster than VRG.

In this chapter, we introduced the algorithm and toolbox we used to run DRG, SRG, and VRG. Then, we compared the computation time to run SRG, DRG and VRG. The results show that SRG requires less computational effort than that of DRG, and DRG requires less computational effort than that of VRG. However, in general, DRG performs better than SRG and is comparable to VRG.

Chapter 5

Practical Example

In this chapter, firstly, we will introduce a practical example: distillation process. Then, we will illustrate the simulation results after applying diagonal DRG and identity DRG to it. Finally, we will compare the DRG and VRG in steady-state.

5.1 Distillation Process

Distillation is a process of separating components from mixed liquid by selective boiling point and condensation [1]. The overall control problem for the distillation process, as shown in Figure 5.1, has five inputs:

$$u = [L \;\; V \;\; D \;\; B \;\; V_T]^T$$

which are the reflux L, boilup V, distillation D, bottom flow B, overall vapor V_T. The five outputs are:

$$y = [y_D \;\; x_B \;\; M_D \;\; M_B \;\; p]^T$$

which represent top composition, bottom composition, condenser holdup, reboiler holdup, and pressure, respectively.

The mixed liquid are fed into the distillation column through F. Based on different boiling point of different components, the "lightest" products (those with the lowest boiling point) exit from the top of the columns and the "heaviest" products (those with the highest boiling point) exit from the bottom of the column. The products that come from the top of the columns will first be condensed and cooled down. Then part of them will return to the columns again and the rest of them will be collected as overhead product. Meanwhile, the products that come out from the bottom of the columns will first be condensed and reboiled, as the same, part of them would become bottom products and the rest of them will return back to the columns. These processes are used to increase the efficiency of the distillation columns.

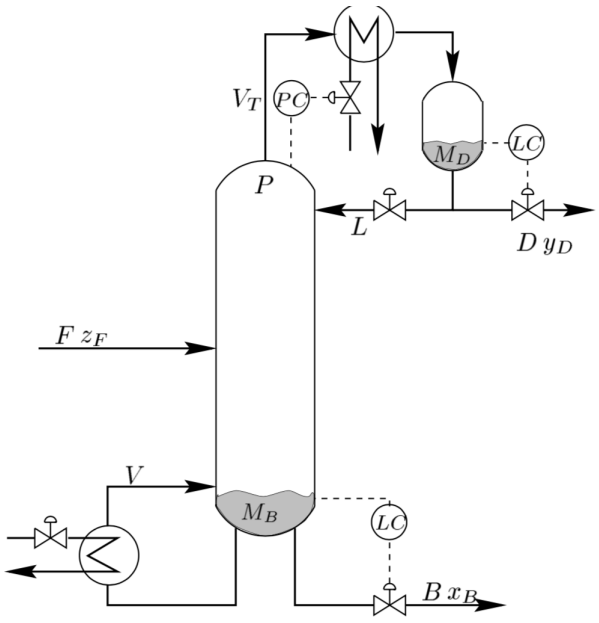

Figure 5.1: distillation process [1]

In this work, we only focus on partially controlled system:

$$u_1 = [L \;\; V]^T$$

to control top composition y_D and bottom composition x_B. The following model presents an ideal model of this subsystem [1]:

$$P(s) = \frac{1}{75s+1}\begin{bmatrix} 87.8 & -86.4 \\ 108.2 & -109.6 \end{bmatrix}$$

To get a high purity for each product, we impose the following constraints: $y_1(t) \leq 1.1$ and $y_2(t) \leq 0.5 \; \forall t$. To implement DRG, we need a controller to close the loop of the system. In this work, we use a SVD-controller [69]. SVD-controller is a special case of a pre- and post-compensator design. The block diagram for pre- and post-controller is as Figure 5.2 shows:

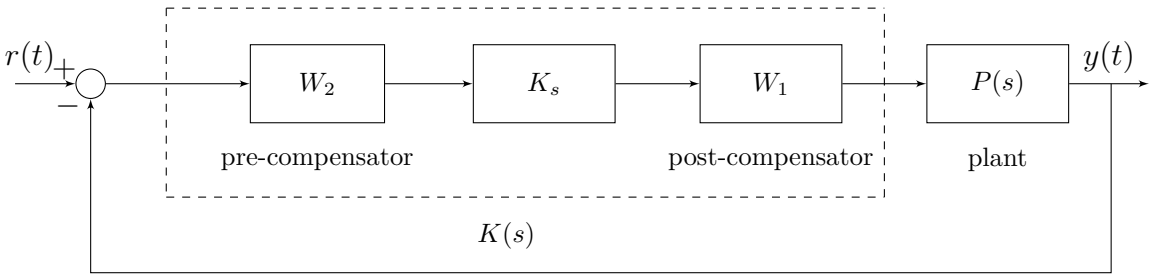

Figure 5.2: Pre- and Post-compensator

The overall controller is:

$$K = W_1 K_s W_2$$

where $W_1 = V_0$, $W_2 = U_0^T$ and V_0, U_0 are obtained from the singular value decomposition of P_0, where P_0 is an approximation of $P(jw_0)$ at a given frequency w_0 (note

that $P(s)$ is the plant in Figure. 5.2).

After designing a SVD controller for this system, the close-loop system becomes:

$$G(s) = P(s)K(s) \times (I + P(s)K(s))^{-1}$$

where

$$K(s) = \begin{bmatrix} 0.45 & 0.89 \\ 0.89 & -0.45 \end{bmatrix} \begin{bmatrix} c_1 \frac{75s+1}{s} & 0 \\ 0 & c_2 \frac{75s+1}{s} \end{bmatrix} \begin{bmatrix} 0.71 & -0.71 \\ 0.71 & 0.71 \end{bmatrix}$$

c_1 and c_2 are adjustable parameters that can be tunned to obtain systems with different interaction behavior. In this work, we change these parameters to find two systems with different condition numbers.

5.2 Implementation of DRG on the Distillation Process

To implement DRG, we first discretize $G(s)$ using sample and hold. Then, we apply the diagonal and identity decoupling methods, which we know from previous analysis work well for systems with small condition number. Below, we illustrate that with small condition number, the gap between $r(t)$ and $u(t)$ would be small, and also illustrate Theorem 1. To do so, we choose two pairs of c_1 and c_2. The constraint set is defined as $\mathbb{Y} := \{(y_1, y_2) : y_1 \leq 1.1, y_2 \leq 0.5\}$. We let $r_1(t)$ and $r_2(t)$ both be unit step functions.

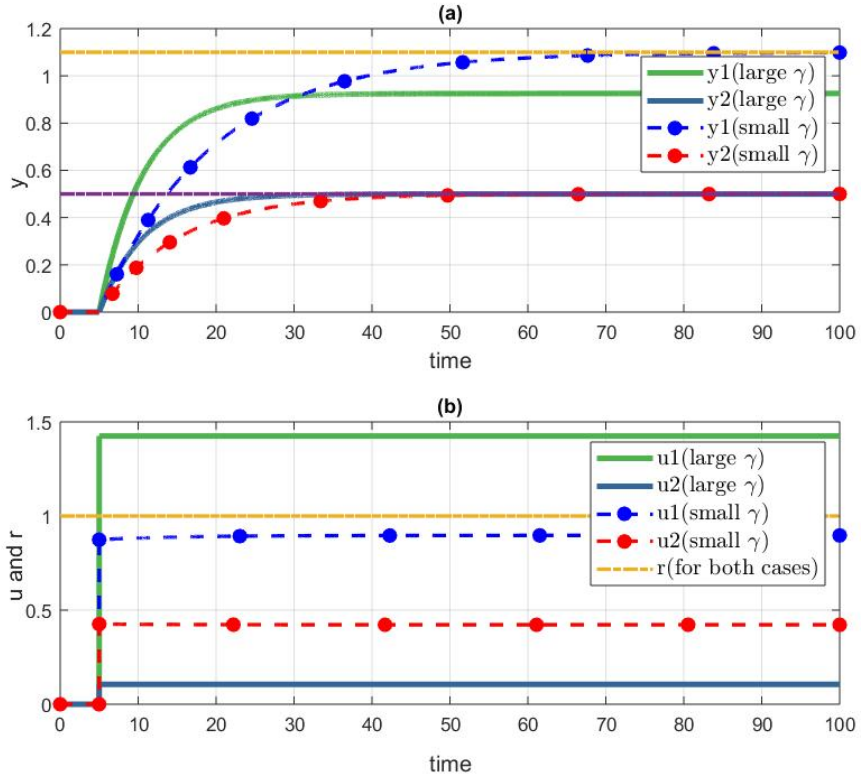

Figure 5.3: Simulation results of diagonal method. γ refers to condition number. Top plot (a) shows the outputs and the bottom plot (b) is r vs. u

Larger condition number case: Choosing $c_1 = 0.0009$ and $c_2 = 0.05$, the condition number (i.e., γ) is 2.55. The simulation results of applying DRG to this system after using diagonal and identity decoupling methods are shown in Figure 5.3, 5.4(a) and 5.5, 5.6(a), respectively.

Smaller condition number case: Choosing $c_1 = 0.0005$ and $c_2 = 0.05$, the condition number is 1.47. The simulation results of applying DRG to this case are shown in Figure 5.3, 5.4(b) and 5.5 and 5.6(b), respectively.

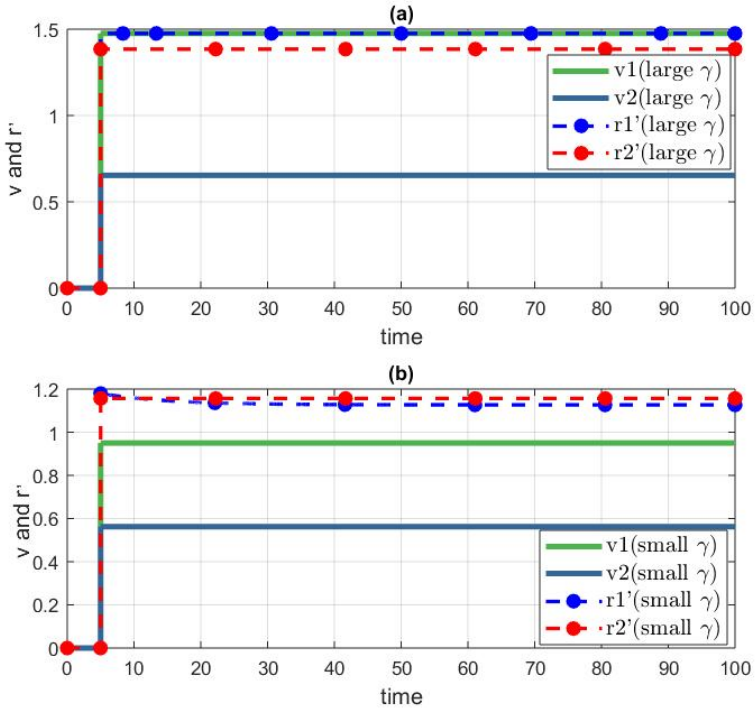

Figure 5.4: Comparison of r' and v in DRG after applying the diagonal method. γ refers to condition number. Top plot (a) is for large condition number and bottom plot (b) is for small condition number.

5.2.1 SIMULATION RESULTS OF DIAGONAL METHOD

From Figure 5.3(a), it can be seen that the outputs satisfy the constraints, which are shown by yellow and purple dashed lines, for both large and small condition number cases. The results in Figure 5.3(b) indicate that the difference between $r(t)$ and $u(t)$ are smaller in the system with smaller condition number. Figure 5.4 confirms that $v(t)$ is always below $r'(t)$, which is a feature of scalar RGs used in the DRG.

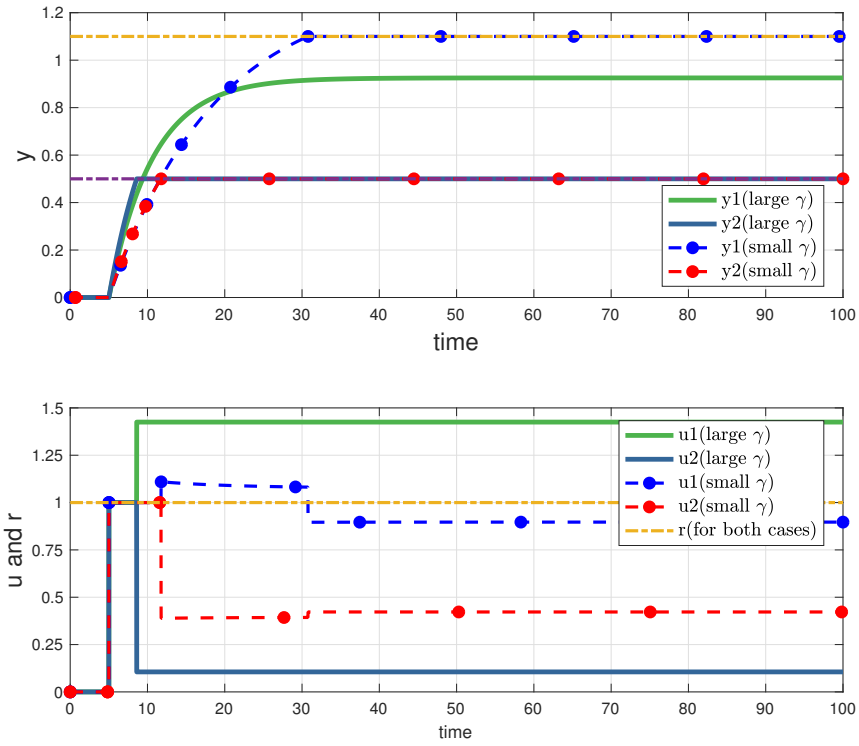

Figure 5.5: Simulation results of identity method. Top plot (a) shows the outputs and the bottom plot (b) is r vs u

5.2.2 SIMULATION RESULTS OF IDENTITY METHOD

In Figure 5.5(a), the constraints are presented by yellow and purple dashes lines. It is clear that the outputs are within the constraints. The results in Figure 5.5(b) show that the gap between $r(t)$ and $u(t)$ is smaller in the system with smaller condition number. Note that $u(t)$ is equal to $r(t)$ during the initial time steps for both small and large condition number cases. The reason is that, during these time steps, no violation is detected for both scalar RGs inside the DRG, and from Figure 5.6, we can see that $v(t)$ is equal to $r'(t)$, which leads to $u(t)$ equals to $r(t)$.

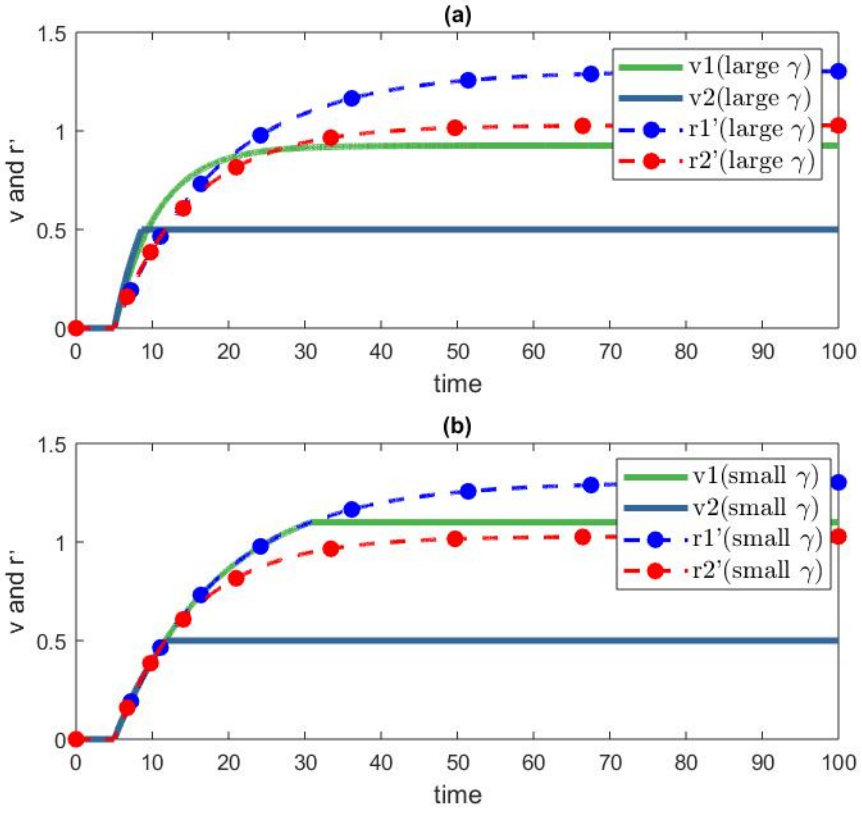

Figure 5.6: Comparison of r' and v in DRG after applying the identity method. Top plot (a) is for large condition number and bottom plot (b) is for small condition number.

5.2.3 Comparison Between DRG and VRG

In this section, we will compare DRG with VRG in steady-state, and this comparison can help us have a better understanding about Theorem 1.

Figure 5.7 compares u computed by DRG and u computed by VRG in steady state, where the left and right plots are for system with large and small condition number, respectively. $r = (1, 1)$ is the set-point. In both figures, the contour lines represent the level sets of the cost functions for VRG (distance from r). From Figure

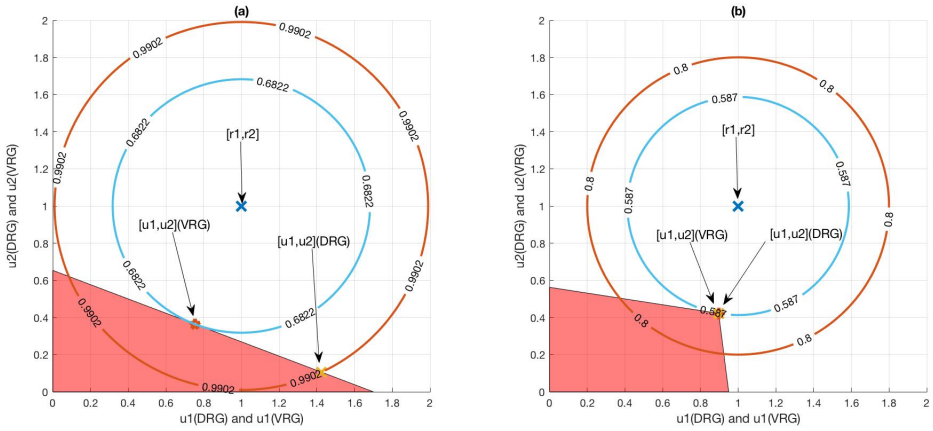

Figure 5.7: Comparison between v for DRG and v for VRG in steady state. Left (a) and right (b) plots corresponds to the system with large and small condition number, respectively.

5.7(a), it can be seen that for VRG, the optimal solution is the closest point to r that belongs to the steady-state constraint-admissible set (shaded region). As for DRG, it finds the solution at a different location, which is a sub-optimal solutions with respect to the VRG cost function. Figure 5.7(b) shows that DRG finds the solution at a vertex in the admissible set (as alluded to in Section 4.1), because both r_1 and r_2 are inadmissible. In this case, the vertex happens to be the closest point to r, so VRG and DRG find the same solution. In sum, the solutions may or may not be the same, depending on the specific situation.

In this chapter, we introduced the distillation process, and compared the simulation results for DRG in both small and large condition number cases. The results show that diagonal and identity DRG both perform better for the systems with smaller condition number in the sense of tracking performance. Finally, we compared VRG and DRG in steady-state. The results illustrate that in this specific example, VRG and DRG find the same optimal solution for smaller condition number system. However,

for large condition number case, DRG finds the sub-optimal solution compared to VRG.

Chapter 6

Conclusion and future research

6.1 Conclusion

Reference governor (RG) is an add-on control strategy for constraint management of closed-loop systems. There are several types of reference governors, such as scalar RG (SRG) and vector RG (VRG). SRG has computational benefits but has performance limitation on MIMO systems. Meanwhile, VRG performs better in MIMO systems but requires higher computational effort.

In this work, a method for constraint management of coupled square MIMO systems was studied. The method is referred to as the Decoupled Reference Governor (DRG), which maintains the computational advantages of SRG and, at the same time, performs better than SRG (comparable to VRG). DRG is based on decoupling the input-output dynamics, followed by application of SRGs to each decoupled channel.

In this thesis, we first presented the DRG formulation with two different decoupling techniques and demonstrated the applicability of the method as a function of the singular values and the condition number of the system. Secondly, we presented

steady state analyses of the DRG and compared the computation time of DRG, SRG, and VRG. It was shown that DRG can run faster than VRG by a factor of 2500 and is similar to SRG in terms of execution time. Finally, a distillation process was used as an illustrative example to compare DRG and VRG with small and large condition numbers.

6.2 Future Research

Future work will explore modifications to DRG to ensure that the inputs to the closed-loop system (i.e. u in Figure 1.5) remain below the references (i.e. r). Moreover, we will explore ways of relaxing the limitations of DRG, specifically, for cases where $G(z)$ is unstable or have non-minimum phase zeros.

To make DRG be more applicable for practical systems, we will also explore the inclusion of external disturbances, plant model mismatch, and observers in the DRG design.

Bibliography

[1] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[2] Sigurd Skogestad, Manfred Morari, and John C Doyle. Robust control of ill-conditioned plants: High-purity distillation. *IEEE Transactions on Automatic Control*, 33(12):1092–1105, 1988.

[3] Norihiko Sumi, Satoshi Hirano, Kosuke Fujimoto, Takeshi Nakajima, Yosuke Kudo, and Koki Ito. Influence of engine oil properties on soot containing deposit formation in turbocharger compressor. Technical report, SAE Technical Paper, 2013.

[4] Nicolas Arnault and Samuel Bonne. Engine lube-oil consumption stakes and benefits from significant blow-by oil mist reduction. In *SAE Technical Paper*. SAE International, 09 2012.

[5] G. Shah and S. Engell. Tuning mpc for desired closed-loop performance for mimo systems. In *Proceedings of the 2011 American Control Conference*, pages 4404–4409, June 2011.

[6] Alberto Bemporad, Francesco Borrelli, Manfred Morari, et al. Model predictive control based on linear programming - the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[7] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice-a survey. *Automatica*, 25(3):335–348, 1989.

[8] Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

[9] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[10] Wook Hyun Kwon and Soo Hee Han. *Receding horizon control: model predictive control for state models*. Springer Science & Business Media, 2006.

[11] Matthew S Elliott and Bryan P Rasmussen. Decentralized model predictive control of a multi-evaporator air conditioning system. *Control Engineering Practice*, 21(12):1665–1677, 2013.

[12] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[13] Graham Goodwin, María M Seron, and José A De Doná. *Constrained control and estimation: An Optimisation Aapproach*. Springer Science & Business Media, 2006.

[14] Wenlin Wang, D. E. Rivera, and K. G. Kempf. Centralized model predictive control strategies for inventory management in semiconductor manufacturing supply chains. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 1, pages 585–590 vol.1, June 2003.

[15] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar. Distributed model predictive control. *IEEE Control Systems*, 22(1):44–52, Feb 2002.

[16] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

[17] Riccardo Scattolini. Architectures for distributed and hierarchical model predictive control-a review. *Journal of Process Control*, 19(5):723–731, 2009.

[18] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

[19] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[20] N Lehtomaki, NJAM Sandell, and Michael Athans. Robustness results in linear-quadratic Gaussian based multivariable control designs. *IEEE Transactions on Automatic Control*, 26(1):75–93, 1981.

[21] F Garelli, RJ Mantz, and H De Battista. Limiting interactions in decentralized control of MIMO systems. *Journal of Process Control*, 16(5):473–483, 2006.

[22] Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications*. Crc Press, 1998.

[23] Morten Hovd, Richard D Braatz, and Sigurd Skogestad. SVD controllers for H_2, H_∞ and μ-optimal control. *Automatica*, 33(3):433–439, 1997.

[24] Virginia Klema and Alan Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[25] Jeff B Burl. *Linear optimal control: H_2 and H_∞ methods*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[26] N. Sandell, P. Varaiya, M. Athans, and M. Safonov. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control*, 23(2):108–128, Apr 1978.

[27] Dragoslav D Siljak. *Decentralized control of complex systems*. Courier Corporation, 2011.

[28] Karl Johan Astrom and Lars Rundqwist. Integrator windup and how to avoid it. In *American Control Conference, 1989*, pages 1693–1698. IEEE, 1989.

[29] PJ Campo, M Morari, and CN Nett. Multivariable anti-windup and bumpless transfer: A general theory. In *American Control Conference, 1989*, pages 1706–1711. IEEE, 1989.

[30] S. S. Ge and Z. Li. Robust adaptive control for a class of MIMO nonlinear systems by state and output feedback. *IEEE Transactions on Automatic Control*, 59(6):1624–1629, June 2014.

[31] P. Kapasouris, M. Athans, and G. Stein. Design of feedback control systems for stable plants with saturating actuators. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 469–479 vol.1, Dec 1988.

[32] Petros Kapasouris, Michael Athans, and Günter Stein. Design of feedback control systems for unstable plants with saturating actuators. In *Proc. IFAC Symposium on Nonlinear Control System Design, Pergamon Press, Oxford*, 1990.

[33] Richard C Dorf and Robert H Bishop. *Modern control systems*. Pearson, 2011.

[34] 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 *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 1, pages 144–149. IEEE, 1994.

[35] Elmer G Gilbert and Ilya Kolmanovsky. Discrete-time reference governors for systems with state and control constraints and disturbance inputs. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 2, pages 1189–1194. IEEE, 1995.

[36] Ilya Kolmanovsky, Emanuele Garone, and Stefano Di Cairano. Reference and command governors: A tutorial on their theory and automotive applications. In *American Control Conference (ACC), 2014*, pages 226–241. IEEE, 2014.

[37] Emanuele Garone, Stefano Di Cairano, and Ilya Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 1 2017.

[38] 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.

[39] Elmer G. Gilbert and Chong-Jin Ong. Constrained linear systems with hard constraints and disturbances: An extended command governor with large domain of attraction. *Automatica*, 47(2):334 – 340, 2011.

[40] Elmer Grant Gilbert and Ilya V Kolmanovsky. Fast reference governors for systems with state and control constraints and disturbance inputs. 1999.

[41] Ilya Kolmanovsky and Elmer G Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical problems in engineering*, 4(4):317–367, 1998.

[42] Jing Sun and Ilya V Kolmanovsky. Load governor for fuel cell oxygen starvation protection: A robust nonlinear reference governor approach. *IEEE Transactions on Control Systems Technology*, 13(6):911–920, 2005.

[43] Ardalan Vahidi, Ilya Kolmanovsky, and Anna Stefanopoulou. Constraint handling in a fuel cell system: A fast reference governor approach. *IEEE Transactions on Control Systems Technology*, 15(1):86–98, 2007.

[44] Ilya Kolmanovsky, Emanuele Garone, and Stefano Di Cairano. Reference and command governors: A tutorial on their theory and automotive applications. In *American Control Conference (ACC), 2014*, pages 226–241. IEEE, 2014.

[45] Hayato Nakada, Peter Martin, Gareth Milton, Akiyuki Iemura, and Akira Ohata. An application study of online reference governor to boost pressure control for automotive diesel engines. In *American Control Conference (ACC), 2014*, pages 3135–3140. IEEE, 2014.

[46] 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.

[47] Kenji Hirata and Hiroshi Minemura. Experimental evaluations of reference governor control schemes with applications to the constrained control of rc helicopters. In *Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on*, volume 2, pages 855–859. IEEE, 2004.

[48] Elmer 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.

[49] Franco Blanchini and Stefano Miani. *Set-theoretic methods in control*. Springer, 2008.

[50] Georges Bitsoris. Positively invariant polyhedral sets of discrete-time linear systems. *International Journal of Control*, 47(6):1713–1726, 1988.

[51] M Vassilaki, JC Hennet, and G Bitsoris. Feedback control of linear discrete-time systems under state and control constraints. *International Journal of Control*, 47(6):1727–1735, 1988.

[52] Abdellah Benzaouia and Christian Burgat. The regulator problem for a class of linear systems with constrained control. *Systems & control letters*, 10(5):357–363, 1988.

[53] Ilya Kolmanovsky and Elmer G Gilbert. Maximal output admissible sets for discrete-time systems with disturbance inputs. In *American Control Conference, Proceedings of the 1995*, volume 3. IEEE, 1995.

[54] Y. Ohta and H. Tanizawa. On approximation of maximal admissible sets for nonlinear continuous-time systems with constraints. In *2007 American Control Conference*, pages 5206–5211, July 2007.

[55] L Magni, Giuseppe De Nicolao, Lorenza Magnani, and Riccardo Scattolini. A stabilizing model-based predictive control algorithm for nonlinear systems. *Automatica*, 37(9):1351–1362, 2001.

[56] M Rachik, A Abdelhak, and J Karrakchou. Discrete systems with delays in state, control and observation: The maximal output sets with state and control constraints. *Optimization*, 42(2):169–183, 1997.

[57] Sorin Olaru and S-I Niculescu. Predictive control for linear systems with delayed input subject to constraints. *IFAC Proceedings Volumes*, 41(2):11208–11213, 2008.

[58] Mohamed Amin Ben Sassi and Antoine Girard. Computation of polytopic invariants for polynomial dynamical systems using linear programming. *Automatica*, 48(12):3114–3121, 2012.

[59] Andreas Freuer, Marcus Reble, Christoph Böhm, and Frank Allgöwer. Efficient model predictive control for linear periodic systems. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems-MTNS*, volume 5, pages 1403–1409, 2010.

[60] P Falb and William Wolovich. Decoupling in the design and synthesis of multivariable control systems. *IEEE transactions on automatic control*, 12(6):651–659, 1967.

[61] I. Kolmanovsky and E. G. Gilbert. Maximal output admissible sets for discrete-time systems with disturbance inputs. 3:1995–1999 vol.3, Jun 1995.

[62] Dale E Seborg, Duncan A Mellichamp, Thomas F Edgar, and Francis J Doyle III. *Process Dynamics and Control*. John Wiley & Sons, 2010.

[63] E. Bristol. On a new measure of interaction for multi-variable process control. *IEEE Transactions on Automatic Control*, 11(1):133–134, Jan 1966.

[64] Morten Hovd and Sigurd Skogestad. Simple frequency-dependent tools for control system analysis, structure selection and design. *Automatica*, 28(5):989–996, 1992.

[65] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons, 2005.

[66] Sigurd Skogestad and K Havre. The use of RGA and condition number as robustness measures. *Computers & chemical engineering*, 20:S1005–S1010, 1996.

[67] Petter TøNdel, Tor Arne Johansen, and Alberto Bemporad. An algorithm for multi-parametric quadratic programming and explicit mpc solutions. *Automatica*, 39(3):489–497, 2003.

[68] M Herceg, M Kvasnica, C Jones, and M Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, 2013.

[69] Yeung Sam Hung and AG MacFarlane. *Multivariable feedback: a quasi-classical approach*. Springer-Verlag New York, Inc., 1982.

Appendices

Appendix A

Matlab Code for Distillation Process

In the distillation process example, we test two cases, one with large condition number, another with small condition number. We will only illustrate the code for large condition number system, the code for small condition number case is similar to that of large condition number case.

```
% Assume we know the state-space of F system: A_f, B_f, C_f, D_f,
% State-space of F^(-1): A_finv, B_finv, C_finv, D_finv and
% State-space of two decoupled systems: A1,B1, C1, D1
% and A2, B2, C2, D2
% Sample time is 10ms. Simulation length is 10,000 timesteps.
% We assume all states are known.

time=0.01 * [0:10000]';
r1 = 1*stepfun(time,0.2);
r2 = 1*stepfun(time,0.2);
r = [r1, r2];

% build O_inf for two decoupled system
[Hx1, Hv1, h1, jstar1] = Oinf_builder(A1, B1, C1,
                         D1 ,S1, s1, e1);
[Hx2, Hv2, h2, jstar2] = Oinf_builder(A2, B2, C2,
                         D2 ,S2, s2, e2);

% implement SRG to decoupled system
x_finv=zeros(size(A_finv,1),1);
xfinv_later=zeros(size(A_finv,1),1);
```

```
v1 = zeros(10000,1);x1 = zeros(size(Hx1,2),1);
v2 = zeros(10000,1);x2 = zeros(size(Hx2,2),1);

for t=1:10000

    % get r1' and r2'
     xfinv_later = A_finv*x_finv + B_finv*r(t,:)';
    r_new(t+1,:) = C_finv * x_finv + D_finv * r(t,:)';
    x_finv=xfinv_later;
    r_new(t+1,1) = r1_new(t+1);
    r_new(t+1,2) = r2_new(t+1);

    % find kapa through Hx, Hv, h, jstar,
    %A, B, C, D, r' and x
    kapa1 = findkapa(Hx1, Hv1, h1, jstar1,
                    r_new(t+1), x1, v1(t));
    kapa2 = findkapa(Hx2, Hv2, h2, jstar2,
                    r_new(t+2), x2, v2(t));

    % using the equation: v(t+1) = v(t) +
    % kapa * (r(t+1)-v(t)) to
    % get governed input to closed-loop system
    v1(t+1) = v1(t) + kapa1*(r_new(t+1,1)-v1(t));
    v2(t+1) = v2(t) + kapa2*(r_new(t+1,2)-v2(t));

    % observer code (if needed) goes here

    x1_later = A1 * x1 + B1* v1(t+1);
    y1(t+1,:) = C1 * x1 + D1 * v1(t+1);
    x1=x1_later;

    x2_later = A2 * x2 + B2* v2(t+1);
    y2(t+1,:) = C2 * x2 + D2* v2(t+1);
    x2 = x2_later;
end

% get the inputs to the closed-loop system
x_f=zeros(size(A_f,1),1);xf_later=zeros(size(A_f,1),1);
for t=1:10000
    v=[v1(t);v2(t)];
```

```
    xf_later = A_f*x_f + B_f*v;
    u(t,:) = C_f*x_f + D_f*v;
    x_f = xf_later;
end

%-------------- function2: build O_inf ---------------%
function [Hx, Hv, h,jstar] = Oinf_builder(A,B,C,D,S,s,e)

nu = size(A,1)
I=eye(nu);

% first two rows of Hx
Hx=[0*S*C;S*C];

% first two rows of Hv
Hv=[S*(C*(I-A)^(-1)*B+D);S*D];

% first two rows of h
h=[(1-e)*s;s];
a=[Hx,Hv];
b=h;

% we assume upper bound of 400 timesteps
for m=1:400

    Hx=[Hx;S*C*A^(m)];
    Hv=[Hv;S*(C*(I-A^(m))*(I-A)^(-1)*B+D)];
    h=[h;s];

    num1_1 = size(a,1);
    num2_1 = size(b,1);

    fun=[-(Hx(num1_1+1,:)),-(Hv(num1_1+1,:))];
    [x,FVAL] = linprog(fun,[a;-fun],[b;h(num2_1+1)+100]);
    if ((-FVAL)<=h(num2_1+1))
        break;
    else
        a=[a;Hx(num1_1+1,:),Hv(num1_1+1,:)];
        b = [b;h(num2_1+1,:)];
    end
```

```
end
 jstar = m;
end

%--------------- function3: find kapa ----------------%
function kapa = findkapa(Hx, Hv, h, jstar, r, x, v)

% solve linear program: kapa*Hv*(r(t)-v(t)) <= h-Hx*x-Hv*v(t)
% to find kapa.
    a = Hv*(r-v);
    b = h-Hx*x-Hv*v;

    k=1; % Algorithm 1 to find kapa.
    for i=1:jstar +2
     if a(i) > 0
         k=min(k,b(i)/a(i));
        end
    end
    k=max(k,0);
    kapa = k;
end
```