2022

Reference Governors for MIMO Systems and Preview Control: Theory, Algorithms, and Practical Applications

Yudan Liu
University of Vermont

Reference Governors for MIMO Systems and Preview Control: Theory, Algorithms, and Practical Applications

A Dissertation 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 Doctor of Philosophy
Specializing in Electrical Engineering

August, 2022

Defense Date: June 2nd, 2022
Dissertation Examination Committee:

Hamid Reza Ossareh, Ph.D., Advisor
Safwan Wshah, Ph.D., Chairperson
Luis Duffaut Espinosa, Ph.D.
Tian Xia, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

Abstract

The Reference Governor (RG) is a methodology based on predictive control for constraint management of pre-stablized closed-loop systems. This problem is motivated by the fact that control systems are usually subject to physical restrictions, hardware protection, and safety and efficiency considerations. The goal of RG is to optimize the tracking performance while ensuring that the constraints are satisfied. Due to structural limitations of RG, however, these requirements are difficult to meet for Multi-Input Multi-Output (MIMO) systems or systems with preview information. Hence, in this dissertation, three extensions of RG for constraint management of these classes of systems are developed. The first approach aims to solve constraint management problem for linear MIMO systems based on decoupling the input-output dynamics, followed by the deployment of a bank of RGs for each decoupled channel, namely Decoupled Reference Governor (DRG). This idea was originally developed in my previous work based on transfer function decoupling, namely DRG-tf. This dissertation improves the design of DRG-tf, analyzes the transient performance of DRG-tf, and extends the DRG formula to state space representations. The second scheme, which is called Preview Reference Governor, extends the applicability of RG to systems incorporated with the preview information of the reference and disturbance signals. The third subject focuses on enforcing constraints on nonlinear MIMO systems. To achieve this goal, three different methods are established. In the first approach, which is referred to as the Nonlinear Decoupled Reference Governor (NL-DRG), instead of employing the Maximal Admissible set and using the decoupling methods as the DRG does, numerical simulations are used to compute the constraint-admissible setpoints. Given the extensive numerical simulations required to implement NL-DRG, the second approach, namely Modified RG (M-RG), is proposed to reduce the computational burden of NL-DRG. This solution consists of the sequential application of different RGs based on linear prediction models, each robustified to account for the worst-case linearization error as well as coupling behavior. Due to this robustification, however, M-RG may lead to a conservative response. To lower the computation time of NL-DRG while improving the performance of M-RG, the third approach, which is referred to as Neural Network DRG (NN-DRG), is proposed. The main idea behinds NN-DRG is to approximate the input-output mapping of NL-DRG with a well-trained NN model. Afterwards, a Quadratic Program is solved to augment the results of NN such that the constraints are satisfied at the next timestep. Additionally, motivated by the broad utilization of quadcopter drones and the necessity to impose constraints on the angles and angle rates of drones, the simulation and experimental results of the proposed nonlinear RG-based methods on a real quadcopter are demonstrated.

To my parents

Acknowledgements

First and foremost, I am extremely grateful to my supervisors, Prof. Hamid Ossareh, for this invaluable advice and continuous support throughout my doctoral studies. He provided constructive feedback to improve my research abilities, as well as my analytical and critical thinking, technical writing, team collaboration, and presentation skills.

I would also like to thank the other members of my defense committee, namely, Prof. Tian Xia, Prof. Luis Duffaut Espinosa, Prof. Safwan Wshah (the committee Chair) for spending time to guide me in this dissertation and providing me with helpful feedback about my research.

Finally, I would like to thank all my colleagues at UVM, including Dr. Joycer Osorio, Dr. Sarnaduti Brahma, Ms. Yasaman Pedari, Mr. Mostafa Ali Ayubirad, and Ms. Molly Rose Kelly-Gorham. I want to thank my family and friends for their unconditional support and love.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

A control system regulates and manages the system behavior so that desired performance is achieved, such as setpoint tracking, disturbance rejection, and closed-loop stability. Almost all physical systems are subject to constraints, including physical actuator limits, safety limitations, hardware protection, and efficiency requirements. In recent decades, besides focusing on the performance of the system, practitioners have increasingly recognized the importance of constraint management. As such, numerous control strategies have been proposed to enforce constraints on system dynamics with the intention to protect the hardware components and operators from damage and keep the desired closed-loop system performance. The Reference Governor (RG) is one such a scheme, which is an add-on mechanism that modifies the setpoint (i.e., reference) to a pre-stabilized closed-loop system only if a violation of constraints is predicted; otherwise, the system performance remains unchanged. RG has numerous attractive features, such as recursive feasibility, bounded-input bounded-output

stability, convergence of output for constant setpoint signals, and computational efficiency. However, due to its structural limitation, RG may lead to a conservative response on multi-input multi-output systems (MIMO), yet most of the industrial control systems are MIMO. Also, RG only takes the current references into consideration and, thus, is unable to incorporate the preview information of references and disturbance into its design. Furthermore, RG is not capable of effectively handling system nonlinearities, which are the characteristics of almost all practical systems. This dissertation presents extensions of RG to overcome the above shortcomings and, thus, extend the applicability of RG. More specifically, this work pursues: a new RG solution for linear MIMO systems that maintains the computational simplicity of RG; a novel RG-based solution to enforce the constraints while incorporating the preview information of the reference and disturbance signals; and a new RG scheme that can be applied to nonlinear MIMO systems. Finally, motivated by the broad utilization of quadcoper drones, this dissertation also presents the implementation of the proposed nonlinear RG-based methods on quadcopter drones.

1.2 Overview of Reference Governor and Problem Statement

Reference governor (RG) is an add-on predictive control scheme that enforces pointwise-in-time state and output constraints in the closed-loop systems [3]. More specifically, a block diagram of RG is shown in Figure 1.1. As the figure shows, RG modifies, whenever is required, the reference ($v(t)$) to a well-designed stable closed-loop system $G(z)$ to enforce the constraints on the state and/or outputs. The RG employs the

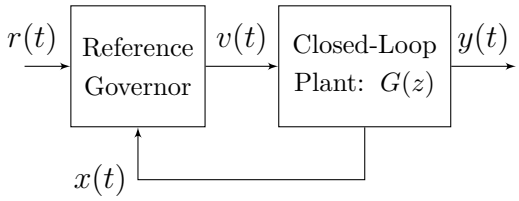

Figure 1.1: Reference governor block diagram. In this figure, $r(t)$, $u(t)$, $y(t)$, and $x(t)$ are the reference, input, constrained output, and state, respectively.

so-called Maximal Admissible set (MAS) [4], , which is defined as the set of all initial conditions and inputs that ensure constraint satisfaction for all future times. This set is computed offline. In real-time, RG computes an optimal $v(t)$ to maintain the system state inside the MAS and, thus, enforce the constraints. This is achieved by solving, at every timestep, a simple linear program (LP), whose solution can be computed explicitly.

As mentioned before, the application of RG has be limited to certain systems. Below, a detailed explanation on the problems aimed to solve in the dissertation will be presented. For the sake of clarity, the problem statement for each one of the constraint management schemes presented in this dissertation is explained individually.

1.2.1 Constraint Management for linear MIMO systems

Many control systems in practice are multiple-input and multiple-output systems (MIMO). Control and constraint management of MIMO systems have been explored in the field of controls for many decades. The control of MIMO systems has been the focus of many works in the literature, for example the Linear Quadratic Regulator (LQR), state feedback control methods, sliding mode control, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control,

and decentralized and centralized control methods, please see [5–13]. The problem of constraint management of MIMO systems has been explored as well. One route is to first find a suitable compensator to decouple the input-output dynamics, see [14–16] (for a more comprehensive review on decoupling method, please see [17]). Afterwards, a diagonal controller for the newly decoupled plant is designed. The constraint management problem is tackled by nonlinear functions (e.g., saturation functions) that maintain the constrained signal within the desired bounds. However, this approach can compromise the closed-loop stability and may not enforce state constraints. Another approach is Model Predictive Control (MPC), see [18–20], which addresses tracking problem while simultaneously enforce point-wise time state and output constraints. However, MPC tends to be computationally demanding, which has limited its applicability, especially for systems with fast dynamics and/or high order. Other approaches to solve constraint management are l_1-optimal control, see [21], barrier Lyapunov function, see [22, 23], and constrained LQR, see [24].

A computationally attractive alternative to MPC is the RG (as shown in Figure 1.1), which can be designed independently of the tracking controller and alleviates the above shortcomings of MPC. However, Standard RG uses a single decision variable in the LP to simultaneously govern all the channels of a MIMO system. As a result, it tends to have a conservative response. A modification of the RG, which performs well in MIMO systems, is the so-called Vector Reference Governor (VRG), see [25]. This technique handles constraint management by solving a quadratic program (QP) with multiple decision variables (one for each reference input). Even though VRG shares some properties with RG, its implementation demands a higher computational load in comparison with RG. This is because of the QP with multiple

decision variables that must be solved at each time step, either by implicit methods or multi-parametric explicit methods.

Problem statement: The first problem tackled in this dissertation is constraint management for linear MIMO systems as motivated above. Note that in my previous work [26], a novel RG-based approach is developed to solve this problem, namely Decoupled Reference Governor based on Transfer Function (DRG-tf), by decoupling the input/output dynamics of the system in the transfer function (Laplace) domain, followed by the implementation of a bank of RGs. However, [26] lacks a thorough analysis of DRG-tf, cannot handle systems with disturbances and noise, lacks applicability to non-square MIMO systems, and does not address the issue of observer design. This dissertation targets to fill those gaps, as well as extend the formula of DRG to systems with state-space representations.

1.2.2 Constraint Management with Preview Control

Preview control has been a subject of study in the field of control theory for many decades. The essential idea behind preview control is to incorporate known or estimated information on the future values of the disturbances or references (i.e., preview information) in the computation of the current control command. Such preview may be computed from models or may be available from measurements. As an example, in a wind turbine control application, preview information on wind velocity may be available from lidar sensors or measurements taken elsewhere in the wind farm [27]. Incorporating this information in the calculation of the control command can result in

improved system performance and cost effectiveness [28]. Other examples of preview control applications include automotive active suspension [29] and swing leg trajectories of biped walking robots [30]. In the former, preview information of the road can be used to improve ride comfort. In the latter, the preview information can be Incorporated to help robots adapt to the environment.

Common approaches for preview control are H_∞ preview control [31], LQR preview control [32], and mixed H_2-H_∞ method, just to mention a few (see [33] for a comprehensive review). These methods, however, do not allow pointwise-in-time state and control constraints enforcement, which is important to ensure safe and efficient system operation. A preview control method that can, in fact, enforce these constraints is Model Predictive Control (MPC) [28–30]. However, as mentioned in previous section, MPC tends to be numerically expensive and not suitable for systems with fast dynamics. A computationally attractive alternative to MPC is the Reference Governor (RG). However, as shown in Figure 1.1, RG uses only the value of the reference signal at the current time and thus is unable to take the preview information into account.

Problem statement: The second problem solved in this dissertation is constraint management, based on RG, to enforce output, state, and control constraints while taking into account the preview information of the reference and/or disturbance signals to further improve system performance.

1.2.3 Constraint Management for Nonlinear MIMO systems

As mentioned before, many control systems in practice are MIMO, and almost all systems are nonlinear. Existing constraint management schemes for nonlinear MIMO systems are either computationally demanding (for example, due to the use of a Nonlinear Model Predictive Control [34, 35]) or require Lyapunov-based functions [36, 37] that may be difficult to obtain in practice (also may lead to conservative response).

Recently, RG, which was originally proposed for linear system, has been extended to handle constraint management for nonlinear systems. Some of nonlinear RG schemes still rely on Lyapunov-based methods [38–40], which, as mentioned before, may result in conservative solutions depending on the application. Reference [41] is one exception, where a bisectional search together with online numerical simulations is proposed to compute optimal constraint-admissible references for a nonlinear hydrogen fuel cell application. The method does not employ Lyapunov functions but is very costly in terms of computation time. We refer to the method in [41] as the nonlinear RG (NL-RG) and leverage it in our work. While NL-RG can guarantee constraint enforcement for nonlinear systems, its response may be overly conservative for MIMO systems such as quadcopter drone since, similar to RG, only one decision variable is used to govern all the channels.

Problem statement: Thus, the third problem solved in this dissertation is constraint management, based on RG, to enforce constraints on nonlinear MIMO systems without the overly conservative response of NL-RG. This dissertation also aims

to lower the computational footprint of the developed scheme, using techniques from machine learning.

1.2.4 Constraint Management for Quadcopter Drones

Unmanned aerial vehicles (UAV) are widely used in military and commercial applications. Military applications include border security [42] and surveillance [43]. Commercial applications include agriculture monitoring [44], livestock classification and counting [45], and fire detection [46]. This increased utilization is due to the numerous advantages of UAVs, including safety, efficiency, simplicity, etc. In the control field, various strategies have been developed to stabilize and control the attitude and/or attitude rate of the quadcopter. Examples are cascade PID controller [47], linear quadratic regulator [48], sliding mode control [49], feedback linearization [50], and Model Predictive Control (MPC) [51]. For a more comprehensive review, please see [52].

The focus of this dissertation is on constraint management of nonlinear quadcopter drones, which is motivated by the fact that constraining the angles (pitch, roll, and yaw) and angle rates is typically necessary to ensure safe and robust operation. For example, constraining pitch and roll angles will prevent the quadcopter operating point from entering nonlinear regions, where linear controllers fail to stabilize the system.

Current constraint management strategies include MPC [53, 54], which may be computationally challenging, Lyapunov-based function [55, 56], and nonlinear RG

schemes [2, 40, 57], which may be conservative on MIMO systems. These methods are not satisfactory for quadcopters, whose dynamics are fast and sampling time are small. Current research on implementing RG-based approaches for a quadcopter includes [58–61]. However, these works either only consider linearized quadcopter dynamics, implement Command Governors, which have a larger computational time than RG, or use Explicit Reference Governor that has a more conservative response since it leverages Lyapunov functions.

Problem statement: The final problem solved in this dissertation is the constraint enforcement of quadcopter drones, validated in both numerical simulations and practical experiments. Our experimental platform is the Crazyflie 2.0 [62], which is a open source flying development platform.

1.3 Literature Review

In this section, a more thorough literature review will be presented on the following topics: constraint management strategies; Neural Network function approximation, which will be employed to speed up the computation time of our proposed schemes; and quadcopter control.

1.3.1 Literature Review on Constraint Management

Some common strategies for constraint management includes:

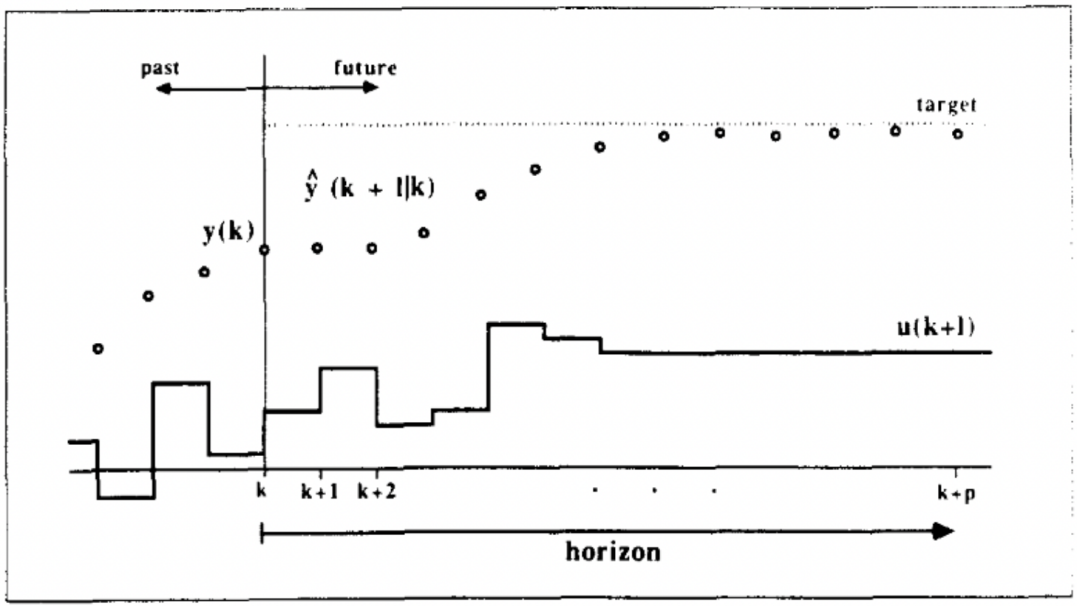

Figure 1.2: MPC [1]

Model Predictive Control (MPC)

MPC is a scheme that can simultaneously tackles the tracking problem as well as constraint enforcement. MPC has a tremendous impact on industry due to its numerous attractive features, such as: 1) it allows the system performance at current time to be optimized while also take the future performance into consideration; 2) it admits the design of multivariable feedback controllers with similar procedural complexity as single variable ones; 3) can be generalized to numerous systems with different structures, such as MIMO or SISO systems, systems with preview information, linear or nonlinear systems, etc; 4) allows for the requirements in the design of constraints on system inputs, states, and outputs.

The main idea behinds MPC is duplicated in Figure 1.2, where k is the discrete

time index. For a vector y, the notation $y(k+i|k)$ denotes the value of y predicted i steps ahead of k. At current time index k, MPC involves predicting the future trajectories of the system $\hat{y}(k+j|k)$, where $j = 0, \ldots, N-1$, over the prediction horizon (N) based on the information up to k. By solving optimization problems, MPC intends to select a best input trajectory $u(k|k), u(k+1|k), \ldots, u(t+N-1|k)$ so that the design requirements are met, such as setpoint tracking, constraint enforcement, etc. Only the first sample of the input trajectory $u(k|k)$ is applied to the system and the optimization problem solved again at the next step [63–65] to implement feedback.

Some commonly used MPC scheme are listed below.

- Distributed MPC (DMPC) [66–69]: DMPC is designed for multi-agent system where a centralized controller is not sufficient or even fails to be implemented. In the DMPC scheme, instead of using a single MPC, multiple MPC controllers, each for a particular system, are adopted to take the dynamics, constraints, objectives, interactions among the systems into consideration.

- Robust MPC [70–73]: In real world, almost all systems are affected by disturbance and noise. Robust MPC aims to stabilize the system while enforcing the state and control constraints for all possible realizations of the uncertainties and disturbances.

- Explicit MPC [74–76]: Explicit MPC remove one of the main drawbacks of MPC, namely the need to solve a mathematical optimization program online to compute the control action. Explicit MPC pursues to solve the optimization problem offline by implementing multi-parametric programming techniques and computes the optimal control action offline as an explicit function of the state

and reference.

Besides the MPC scheme listed above, stochastic MPC, where stochastic disturbances is considered, is proposed in [77, 78]. Cascade or hierarchical MPC, as reviewed in [79], is proposed to tackle the computational complexity, robustness and reliability problems, and communication bandwidth limitations of large-scale systems.

Barrier Lyapunov Functions (BLF)

BLF is a level-set function used to provides formal safety guarantees for nonlinear control systems. One property of a BLF is that it tends to infinity as its argument approaches the selected constraint, which can be naturally fit into the constraint enforcement problem [80]. By keeping the BLF bounded in the closed-loop system, it is thus guaranteed that the limits are never violated. Several works related to the BLF methods are listed below. The work presented in [81] investigates the output tracking problem as well as constraint enforcement of nonlinear switched systems. [82] proposes a control design for strict feedback nonlinear systems with time-varying output constraints. [83–87] present several works related to BLF implementation on the systems with parameter uncertainty, disturbance, and noise.

Reference Governor (RG)

Reference Governor (RG) was first proposed as a continuous-time framework [88], whose goal is to modify the control law when necessary to enforce constraints on linear continuous-time systems. However, the continuous-time RG does not involve any prediction on the system dynamics and, thus, the tracking performance can not

be optimized. Then, motivated by the work of Gilbert in 1991 [4], where the Maximal Admissible Set is presented, the prediction control was brought into RG design and the natural extensions of RG to discrete-time systems have been widely adopted. In the same year (1991), a static RG was presented in [4], where a scalar gain is used to govern the input. Because of the possibility of convergence issue, the static RG was replaced by a dynamic RG [89], where the convergence for constant references is guaranteed. In 1995, [25] proposed an extension of dynamic RG to systems with disturbance inputs. The first RG scheme for nonlinear systems was presented in 1998 [90]. After then, numerous RG-based approaches have been developed and the applicability of RG has been extended broadly. For linear system:

- Vector Reference Governor [25]: VRG is proposed with the intention to improve the system performance of RG on multi-input multi-output systems. This is achieved by adding more flexibility in the choice of $v(t)$ but at a cost of increased computational effort.

- Command Governor [91] (CG): CG is developed to speed up the response time of RG by changing the update law of $v(t)$. However, since Quadratic Programming (QP) is required to solve in CG scheme, it thus has a larger computational load than that for RG.

- Extended Command Governor (ECG) [25, 92]: ECG, a modification of CG, is proposed to offer a larger domain of attraction with respect to CG/RG by manually introducing a fictitious dynamics of v.

Besides, [93] presents a Recovery Reference Governor aiming for recovering the system from constraint violation. In [94], a RG-based method for systems with slowly time-

varying references or for enforcing slowly time-varying constraints is proposed. [95] brings a stochastic approach to RG and MAS using chance constraints.

For nonlinear systems:

- Explicit Reference Governor (ERG) [40, 96]: ERG presents a novel control law that modifies the reference of a nonlinear system to ensure the satisfaction of constraints, which is done by translating the constraints into an upper bound on the value of the Lyapunov function and manipulating the velocity of the applied reference to enforce this bound.

- Parameter Governor [57, 57]: The main idea behind Parameter Governor is to modify parameters (such as gains or offsets) in the nominal control laws to avoid violation of pointwise-in-time state and control constraints as well as to improve the overall system transient performance by optimizing a cost function over a finite horizon.

- Nonlinear Reference Governor for Fuel Cell [2]: This thesis proposes a bisectional search algorithm together with online numerical simulations of the system dynamics to find an optimal constraint-admissible reference for a nonlinear hydrogen fuel cell application.

Other than those, [97] presents a output feedback RG for nonlinear systems with unmeasurable states by utilizing an ellipsoidal region in which the state is guaranteed to lie. In [98], a Transient Robust RG is introduced to enforce constraints on nonlinear systems, where a novel Robust Output Admissible Set is introduced. [99] proposes a RG-based approach, which does not require an explicit model of the system or constraints by constructing an approximation of the MAS using online neural network

learning.

Constrained Linear Quadratic Regulator (LQR)

LQR is a state feedback controller aiming to find the optimal feedback gain to optimize the following quadratic program [100]:

$$J = \sum_{k=0}^{\infty} (\|x(k)\|_Q^2 + \|v(k)\|_R^2) \tag{1.1}$$

where x and v represent the state and the control input of the linear system, respectively. The symmetric definite matrix Q and R are usually chosen to be diagonal matrix that determines the significant of the states and control inputs on the cost function. The LQR seeks an optimal feedback gain to minimize (1.1), where the gain can be found using Ricatti equation [101]. Different from MPC, the cost function in LQR has infinite prediction horizon. Also, LQR does not involve any real-time optimization solver. However, traditional LQR method can only be applied to linear and unconstrained systems.

In recent decades, motivated by the importance of constraint management, LQR has been extended to constrained systems. In [102], a technique to compute the explicit state-feedback solution to the linear quadratic optimal control problem subject to state and input constraints is presented. Instead of using Ricatti equation to find the optimal feedback gain, [102] paper proposes a Multi-parametric quadratic programming that can enforce constraints while have lower computational burden than on-line quadratic programming solvers. [103] illustrates an explicit solution to the LQR problem subjects to constraint in order to reduce the demand for real-time

computations. However, in order to address the constraints, a possibly suboptimal strategy is developed.

Constrained H_2 and H_∞

The H_2 and H_∞ control problem consist of internally stabilizing the control system while minimizing the H_2 and H_∞ norm of its transfer function [104]. The H_2 and H_∞ technique are readily applicable to problems involving multivariate systems with coupling behavior between different channels.

The extensions of H_2 and H_∞ methods to constrained systems are proposed in [105, 106], where a linear matrix inequality (LMI) optimization problem is solved to ensure the control and output constraints are respected.

Network Function

Recently, with a broad usage of Neural Network (NN), constraint management strategies based on NN have been widely studied. The constraint-enforcement problem using NN can be achieved using three different routes. First, a traditional constraint-management strategy (i.e., barrier Lyapunov function, Moore Penrose inverse, etc) is employed to deal with the output constraints while the NNs are used to approximate the model of the system [107, 108]. As discussed before, this route may lead to conservative response or may be limited to specific systems. Second, inspired by the seminal work of Hopfield and Tank [109], various neural networks for solving linear and nonlinear programming problems have been investigated. This is achieved by utilizing projective gradient based neural network models, which are de-

rived from constrained minimization problems and complementarity problems with KKT-conditions [110–113]. Additionally, in the control field, NN can be used to approximate the functionality of MPC. The main idea behinds these papers is either modifying the structure of the NN properly [114–116] or projecting the output of the NN into an appropriately defined invariant set [99, 117], where the set can be found using Lyapunov functions.

1.3.2 Literature Review on Neural Network Function Approximation

Neural Networks (NNs), also referred to as Artificial Neural Networks (ANNs), are the method of choice for building learning algorithms. They are now being investigated for numerous tasks such as optical character recognition [118], image recognition [119, 120], system identification [121, 122], etc. Their popularity stems from their success on several challenging learning problems and some superior features they own, such as can be generalized to different data types or different applications using the same NN models, can be executed easily in most application, etc. [123]. A neural network consists of an input layer, an output layer and, in between, hidden layers. The layers are connected via nodes, and these connections form a network. Learning an input-output mapping from a set of examples is one of the most common usages of a NN. This is attained by minimizing a cost function, which is usually defined as the difference between the target output and the output given by NN, over the internal weights and bias. Several common NN models for function approximation are listed below:

- Feedforward Neural Network [124–126]: Feedforward NN refers to the NN models where there is no feedback from the outputs of the neurons toward the inputs throughout the NN.

- Recurrent Neural Network (RNN) [127–129]: Contrast to the feedforward NN, RNN model has feedback loops where data can be fed back into the input before it is fed forward again for further processing and final outputs. Some have argued that since time series data may have autocorrelation or time dependence, the RNN models which take advantage of time dependence may be more suitable than feedforward NN.

- Radial Basis Function Networks (RBF NN) [130–132]: An RBF network is a type of feedforward neural network composed of three layers, namely the input layer, the hidden layer with a nonlinear RBF activation function and a linear output layer. RBF NN gained much popularity in recent times due to their ability to approximate complex nonlinear mappings directly from the input-output data with a simple topological structure.

- Generalized Regression Neural Network (GRNN) [133–135]: GRNN network structure is similar to the RBF network except a slight modification in the hidden layer. More specifically, GNRR is a single-pass associative memory feedforward NN and uses normalized Gaussian kernels in the hidden layer as activation functions.

  Besides the NN models listed above, in [136], a new single-layer NN which is based on orthogonal functions, is presented. Spiking NN [137], which is more biologically realistic than ANNs, is also used for function approximation.

Pi-Sigma network, which is a feedforward network with a single hidden layer of linear summing units and with product units in the output layer that can uniformly approximate any continuous function defined on a compact set, is presented in [138].

1.3.3 Literature Review on Quadcopter Control

The common control strategies to stabilize quadcopter dynamics include:

- Cascade Proportional-Integral-Derivative Control (PID) [139, 140]: PID controller is the most widely used controller in engineering practice using proportional, integral and differential to zero out the error between the tracking setpoint and the actual output of the system. Cascade PID is an advanced application of the PID that can improve the control of systems that are subject to significant lag. In quadcopter application, the cascade PID controller consists of two loops: the inner (also the angular velocity or "rate") loop and the outer (also the angle or "attitude") loop, in which the outer loop regulates the inner loop.

- Linear Quadratic Regulator (LQR) [141, 142]: As mentioned before, LQR intends to solve (1.1) using Ricatti equation. However, the applicability of LQR has been limited to linear systems. Thus, to implement LQR on nonlinear quadcopter dynamics, feedback linearization or linearization around hovering model should be used.

- Model Predictive Control (MPC) [51, 143]: MPC, as mentioned before, can simultaneously address setpoint tracking issue as well as constraint enforcement

problem. However, MPC tends to require a large computation time than RG, which may not be applicable to real quadcopter drones, whose dynamics are fast and sampling time are small.

- Feedback linearization [54, 144]: Feedback linearization is a common strategy used to control nonlinear systems. This approach targets to transform the nonlinear system into an equivalent linear control system through a change of variables and a suitable control input. However, the feedback linearization can not handle the large modeling errors and disturbances effectively and requires a precisely modeling of the nonlinear systems.

Besides the control strategies listed above, adaptive control is used to stabilize the quadcopter dynamics in the presence of large/complete parameter uncertainties [145]. Fuzzy controller, which can update the control rule to maintain ideal system performance, is also utilized in [146].

1.4 Original Contributions

This dissertation contributes to the literature of constraint management in the area of dynamical systems and control. Most of the content presented in this dissertation have been published in scientific journals [147,148] or conference proceedings [148,149]. At a high level, this work contributes to the field of set-theoretic constraint management with the focus on both linear and nonlinear systems. Specifically, we develop novel RG schemes that can handle the problems mentioned in Section 1.2. By contributing to the literature with novel schemes in the RG framework, this dissertation provides practical tools and theoretical frameworks for constraint management of a broader

class of dynamical systems.

The individual contributions per topic are listed below.

Decoupled Reference Governor

Chapter 3 presents a computationally efficient solution of constraint management for MIMO systems. The solution, referred to as the Decoupled Reference Governor (DRG), maintains the highly-attractive computational features of RG while having performance comparable to Vector Reference Governors (VRG). The main contributions of Chapter 3 are:

- The transient analysis of DGR-tf as well as observer design.
- The extension of DRG formula to state space representation (i.e., DRG-ss).
- A novel extension of DRG to systems that are affected by unknown additive disturbances and parametric uncertainties is presented.
- An modification of DRG to non-square MIMO systems, which enhances the applicability of DRG.

Preview Reference Governor

Chapter 4 presents a constraint management strategy based on RG to enforce output, state, and control constraints while taking into account the preview information of the reference and disturbance signals. The strategy, referred to as the Preview Reference Governor (PRG), can outperform RG while maintaining the highly-attractive computational benefits of RG. The main contributions of Chapter 4 are:

- A novel RG-based constraint management scheme with preview capabilities, namely the PRG, which is more computationally efficient than existing methods such as MPC or Command Governors.
- An extension of PRG (Multi-N PRG) to further improve the performance of PRG by considering multiple preview horizons.
- Analysis of recursively feasibility, closed-loop stability, and convergence under constant inputs.
- Comparison of the computational footprint and performance of these schemes.
- Extensions of PRG to systems with disturbance preview, parametric uncertainties, inaccurate preview information, and multi-input systems.

Nonlinear Reference Governor

Chapter 5 presents three constraint management strategies based on RG to enforce output and state constraints on nonlinear MIMO systems. The first solution, referred to as the nonlinear decoupled reference governor (NL-DRG), can outperform NL-RG but at the cost of increased computational effort. To address the above shortcoming, the second solution, which is referred to as the modified reference governor (M-RG), is proposed. The M-RG scheme consists of the sequential application of SRGs and, thus, maintains the highly-attractive computational features of SRG. However, the performance of M-RG may be overly conservative compared to NL-DRG. To address the computational issue of NL-DRG while perusing a less conservative response than M-RG, a third solution, which is referred to as the Neural Network Decoupled Reference Governor (NN-DRG), is proposed. The NN-DRG consists of a Neural Network

to approximate the input-output mapping of NL-DRG and a Quadratic Program to ensure the constraints are satisfied at next time step. The main contributions of Chapter 5 are:

- To enforce the constraints on nonlinear MIMO systems, the NL-DRG scheme is introduced and analyzed.
- To further improve the computational footprint of NL-DRG, the M-RG is proposed.
- To reduce the computation time of NL-DRG while having performance superior than M-RG, Neural Network is brought in NL-DRG design, namely NN-DRG.
- The comparison of the computational footprint and performance of all schemes is demonstrated.
- The extensions of the proposed methods to systems with disturbances and noises are presented.

Constraint Management for Nonlinear Quadcopter Dynamic

Chapter 6 illustrates the simulation and experimental results of the proposed nonlinear RG scheme, namely NL-DRG, M-RG, and NN-DRG, to enforce the angle and angle rates of Crazyflie 2.0. The main contributions of Chapter 6 are:

- Motivated by the fact that the real Crazyflie is affected by disturbance and noise and, to ensure constraint satisfaction, the proposed methods must take those uncertainties into consideration. A technique to quantify these disturbances and noise of Crazyflie 2.0 is presented.

- The simulation and experimental results of the proposed nonlinear RG-based solutions on a real system is demonstrated.

1.5 Statement of Impact

The development of the new constraint management strategies for linear as well as nonlinear systems have implications on both theoretical and practical fronts. On the theoretical front, this work provides:

- a new computationally efficient method for constraint management of MIMO system, which has performance comparable to VRG;
- a modification of RG that is able to incorporate the preview information of references and/or disturbances;
- a nonlinear RG that can enforce constraints on a real quadcopter drone with disturbance and sensor noise.

On the practical aspect, this dissertation offers new tools to robustly design closed-loop systems with constraint enforcement capability by extending the proposed methods to systems with additional disturbances, plant/model mismatch, and inaccurate preview information (for PRG).

On the societal front, this work

- develops computationally-efficient constraint management strategies that can be used in distillation process, autonomous driving, flight control, unmanned aerial vehicles, etc;

- provides a more efficient and safe operation by imposing constraints on the outputs and/or states of the system, which improves the products we use everyday, such as autonomous vehicles, wind turbines, air conditioner, etc.

1.6 Notation

The following notations are used in this dissertation. $\mathbb{Z}_+$ denotes the set of all non-negative integers. Let $V, U \subset \mathbb{R}^n$. Then, $V \sim U := \{z \in \mathbb{R}^n : z+u \in V, \forall u \in U\}$ is the Pontryagin-subtraction (P-subtraction) ([150]). The identity matrix with dimension $i \times i$ is denoted by I_i. Given a discrete-time signal $u(t) = [u_1(t), u_2(t), \ldots, u_m(t)]^T$, the L_2 norm is defined as: $\|u\|_{L_2}^2 = \sum_{t=-\infty}^{\infty} u(t)^T u(t)$, and its L_∞ norm is represented as: $\|u(t)\|_{L_\infty} = \sup_t(\max_i |u_i(t)|)$. For a system with transfer function $F(z)$ and impulse response $f(t)$, the H_∞ norm is defined as: $\|F\|_{H_\infty} = \max_w \bar{\sigma}(F(e^{jw}))$, where $\bar{\sigma}$ represents the maximum singular value, and the L_1 norm is defined as: $\|f(t)\|_{L_1} = \max_i \sum_{j=1}^{m} \sum_{\tau=0}^{\infty} |f_{ij}(\tau)|$, where f_{ij} is the ij-th element of f, and m is the number of columns of f. The condition number of a matrix (defined by the ratio of the maximum to the minimum singular values) is denoted by γ. A zero matrix with dimension $i \times j$ is denoted as $0_{i,j}$.

1.7 Outline

The outline of the rest of the dissertation is as follows. In Chapter 2, brief reviews on Reference Governor, Maximal Admissible set, decoupling methods, Neural Network approximation, DRG-tf, and the dynamics of one-arm link robot, rollover prevention,

and quadcopter are provided. In Chapter 3, standard RG is extended to systems with multi inputs and multi outputs, namely DRG, and a thorough analysis on DRG is provided. Chapter 4 describes a novel RG formulation that can enforce the constraints while taking into account the preview information of the reference signals, namely PRG. In Chapter 5, three RG-based schemes are presented to enforce constraints on nonlinear MIMO systems. Chapter 6 demonstrates the simulation and experimental results of the proposed nonlinear RG-based methods on real quadcopter drones. Finally, Chapter 7 concludes the dissertation.

Chapter 2

Review of Reference Governor, Decoupling Methods, Neural Network Approximation, DRG-tf, Practical Examples

This chapter presents reviews on Reference Governor, Maximal Admissible set, decoupling methods, Neural Network approximation, and Decoupled Reference Governor based on Transfer Function (DRG-tf). The practical examples used in this dissertation, namely one-arm link robot, vehicle rollover prevention, and quadcopter drone, are reviewed as well.

2.1 Review of Maximal Admissible Set

To have a better understanding of Reference Governor (RG), first, a review on Maximal Admissible set (MAS) will be provided

Consider the closed-loop discrete time linear time-invariant system $G(z)$ in Figure 1.1 be represented by:

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

where $x(t) \in \mathbb{R}^n$ is the state vector, $v(t) \in \mathbb{R}^m$ is the input, and $y(t) \in \mathbb{R}^p$ is the constrained output vector. Over the output, the following polyhedral constraint is imposed:

$$y(t) \in \mathbb{Y} := \{y : Sy \leq s\} \tag{2.2}$$

where S is introduced to allow freedom to impose constraints on linear combinations of the outputs. To construct the MAS, the following assumptions are made [4]:

A. 1. *System $G(z)$ in Figure 1.1 reflects the combined closed-loop dynamics of the plant with a stabilizing controller. Accordingly, $G(z)$ is asymptotically stable (i.e., $|\lambda_i(A)| < 1, i = 1, ..., n$). Moreover, the pair (C, A) is observable, the constraint set Y is bounded, and $0 \in int\, Y$.*

The MAS, denoted by O_∞, is the set of all initial states and constant references that satisfy (2.2) for all future time steps [4]:

$$O_\infty := \{(x_0, v_0) \in \mathbb{R}^{n+m} : x(0) = x_0, v(t) = v_0, y(t) \in \mathbb{Y}, \forall t \in \mathbb{Z}_+\} \tag{2.3}$$

where $y(t)$ is the output starting from the initial state x_0 and constant input v_0:

$$y(t) = CA^t x_0 + C(I - A)^{-1}(I - A^t)Bv_0 + Dv_0 \tag{2.4}$$

The fact that the construction of MAS involves checking infinite number of inequalities, which make the MAS nearly impossible to characterize in real implementation. To overcome this challenge, [4] presents an finitely determined inner approximation of O_∞ can be obtained by tightening the steady-state constraint and introducing it as a new half-space:

$$P_{ss} := \{(x, v) : G_0 v \in \mathbb{Y}_{ss}\} \tag{2.5}$$

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 $0 < \epsilon \ll 1$. By introducing (2.5) to (2.3), the inner approximation of O_∞ can be characterized by:

$$\bar{O}_\infty = \{(x, v) : Sy(\infty) \leq (1 - \epsilon)s, Sy(j) \leq s, \forall j\} \tag{2.6}$$

As proved in [4], there exists a finite time, j^*, such that $\bar{O}_{j^*} = \bar{O}_{j^*+1}$, implying that (2.6) can be rewritten as:

$$\bar{O}_\infty = \{(x, v) : Sy(\infty) \leq (1 - \epsilon)s, Sy(j) \leq s, \quad j = 1, \ldots, j^*\} \tag{2.7}$$

Substituting (2.4) and (2.5) into (2.7), $\bar{O}_\infty$ can be redefined as:

$$\bar{O}_\infty = \{(x, v) : H_x x + H_v v \leq h\} \tag{2.8}$$

where

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

Besides the finitely deterministic of $\bar{O}_\infty$, [4] also proves that if Y is convex, compact, and polytopic, then so is $\bar{O}_\infty$. To summarize, the set $\bar{O}_\infty$ can be viewed as a polytope, which characterizes the set $(x(t), v(t))$ so that constraint management is achieved for all future time. Without further abuse the notation, in later discussion, O_∞ is referred to the inner approximation of the MAS.

2.1.1 Maximal Admissible Sets for systems with disturbances

In this section, we review the concept of robust MAS for systems affected by additive disturbances [151]:

$$\begin{aligned} x(t+1) &= Ax(t) + Bv(t) + B_w w(t) \\ y(t) &= Cx(t) + Dv(t) + D_w w(t) \end{aligned} \tag{2.9}$$

Similarly, the constraints are imposed on the output so that (2.2) is satisfied. The disturbance input satisfies $w \in \mathbb{W}$, where $\mathbb{W} \subset \mathbb{R}^d$ is a compact polytope with the origin in its interior.

In order to construct the robust MAS for system (2.9), the system output $y(t)$ is

rewritten as a function of the initial state, x_0, the constant input, $v(t) = v_0$, and the disturbances:

$$y(t) = CA^t x_0 + (C(I-A)^{-1}(I-A^t)B + D)v_0 + C\sum_{j=0}^{t-1} A^{t-j-1}B_w w(j) + D_w w(t) \tag{2.10}$$

A key ingredient to simply mathematical computation is the set operation Pontryagin-subtraction (abbreviated as P-subtraction). Defining the sets $\mathbb{Y}_t$ using the following recursion:

$$\mathbb{Y}_0 = \mathbb{Y} \sim D_w\mathbb{W}, \quad \mathbb{Y}_{t+1} = \mathbb{Y}_t \sim CA^tB_w\mathbb{W} \tag{2.11}$$

P-subtraction allows us to rewrite the requirement $y(t) \in \mathbb{Y}$, $\forall w(j) \in \mathbb{W}, j = 0, \ldots, t$ as:

$$CA^t x_0 + (C(I-A)^{-1}(I-A^t)B + D)v_0 \in \mathbb{Y}_t$$

Finally, the robust MAS is defined as:

$$\begin{aligned} \bar{O}_\infty := \{(x_0, v_0) \in \mathbb{R}^{n+m} : G_0 v_0 \in \bar{\mathbb{Y}}, \\ CA^t x_0 + (C(I-A)^{-1}(I-A^t)B + D)v_0 \in \mathbb{Y}_t\} \end{aligned} \tag{2.12}$$

where G_0 is the DC gain of (2.9) from input v to output y, and $\bar{\mathbb{Y}} := (1-\epsilon)\mathbb{Y}_t$ for some $0 < \epsilon \ll 1$ and large t. Similar to previous section, the $\bar{\mathbb{Y}}$ is introduced to ensure finite-determinism of O_∞.

The robust MAS, denoted by $\bar{O}_\infty$, is the set of all safe initial conditions and inputs, such that for any given disturbance, the output constraints are satisfied for all times.

2.2 Review of Reference Governor

In this section, three RG-based schemes will be reviewed. Section 2.2.1 review Scalar Reference Governor. Section 2.2.2 reviews Vector Reference Governor. Section 2.2.4 provides a review on nonlinear Reference Governor.

2.2.1 Review of Scalar Reference Governor

From Section 2.1, it is possible to see that the MAS contains the predictions of the outputs based on the current states and the constant inputs. Based on the predictions, the controller can foretell if a constraint may be violated and then take corrective actions over the reference. The SRG computes $v(t)$ by implementing the following dynamics:

$$v(t) = v(t-1) + \kappa(r(t) - v(t-1)) \tag{2.13}$$

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

$$\begin{aligned} \underset{\kappa\in[0,1]}{\text{maximize}} \quad & \kappa \\ \text{s.t.} \quad & v(t) = v(t-1) + \kappa(r(t) - v(t-1)) \\ & (x(t), v(t)) \in O_\infty \end{aligned} \tag{2.14}$$

where O_∞ is the MAS discussed before. Note that (2.14) implies that κ maneuvers $v(t)$ along a straight line between $v(t-1)$ and $r(t)$. More specifically, $\kappa = 0$ means that, in order to keep the system safe, $v(t) = v(t-1)$, where $v(t-1)$ is already admissible. Furthermore, $\kappa = 1$ means that no violation is predicted and, therefore,

$v(t) = r(t)$. This RG formulation ensures system stability, convergence, and recursive feasibility. For more details, see [25].

One drawback of SRG is that, for Multi-Input Multi-Output (MIMO) systems, since (2.14) solves for only one decision variable, κ, this implies that the control law will prioritize the input with higher risk of constraint violation by manipulating the single κ, which will affect all other input channels. As a results, the system performance may be deteriorated.

2.2.2 Review of Vector Reference Governor

Vector Reference Governor (VRG) extends the capabilities of SRG to MIMO systems and uses diagonal matrix $\mathbf{K}$ instead of a scalar κ:

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

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

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

where $Q = Q^\top > 0$. Because of the increased number of optimization variables and the utilization of the QP formulation, VRG has superior system performance than SRG but at a cost of increased calculation effort.

2.2.3 Review of Command Governor

The intention behinds Command Governor (CG) is, similar to VRG, minimize the Q norm between the input $r(t)$ and the governed command $v(t)$. However, instead of manipulating κ, CG tends to solve the problem by computing $v(t)$ directly:

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

As proved in [91], CG guarantees convergence for a constant reference and system stability.

2.2.4 Review of Nonlinear Reference Governor

As introduced in Section 2.1, the MAS is constructed based on a linear or almost-linear systems. Thus, the applicability of SRG, VRG, and CG has been limited to linear systems as well. Motivated by the necessity to handle nonlinearities in practical applications, numerous extensions of RG theory to nonlinear systems are explored in recent decades. In this section, a detailed review on nonlinear reference governor proposed in [2] is provided. Moreover, the method in [2] is referred as the nonlinear RG (NL-RG) and is leveraged in this thesis.

The essential idea behinds NL-RG is to replace the MAS with online numerical simulation of the system dynamics and find the κ (shown in (2.14)) using bisectional

search algorithm. More specifically, denoted the nonlinear stabilized system as:

$$\begin{aligned} x(t+1) &= f(x(t)) + g(x(t))v(t), \\ y(t) &= h(x(t)) \end{aligned} \tag{2.15}$$

The main idea behinds the bisectional search algorithm is explained below. At current time step t, the decision variable κ is found by simulating the nonlinear dynamics (2.15) forward in time over a finite time horizon N. If the constraints are violated for any time during the simulation, then κ will be reduced and simulation reinitiated. If all the constraints are satisfied for the simulated trajectory, then, κ will be increased to minimize the tracking error. The detailed information to compute the κ is shown in Figure 2.1.

NL-RG inherits the drawback of SRG since only a scalar decision variable κ is used. Moreover, because of the online iterations to find κ, NL-RG tends to be more computationally demanding compared to SRG, VRG, and even CG.

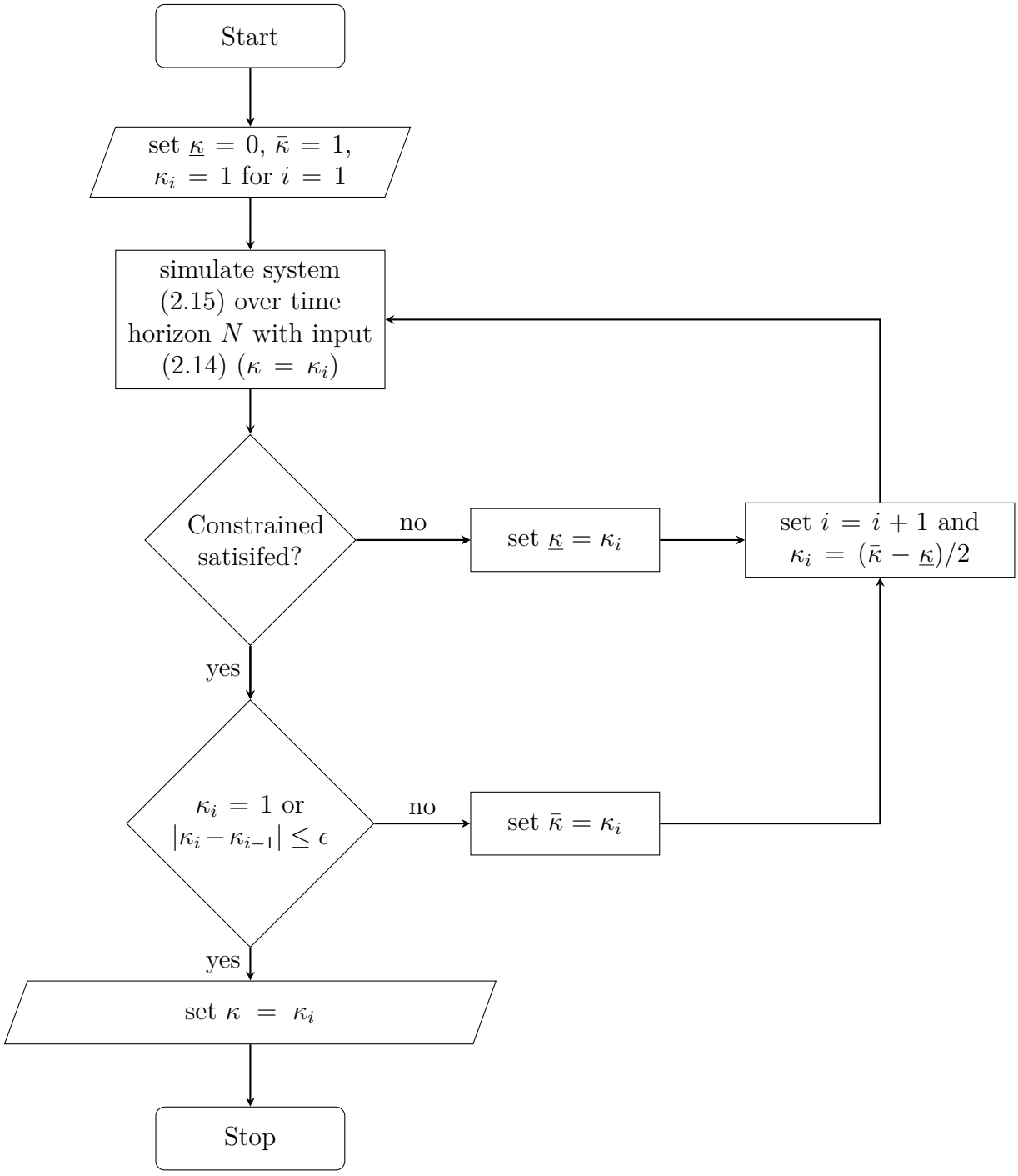

Figure 2.1: Bisectional search algorithm for determining κ at current time step t, where ϵ represents the tolerance for convergence test [2].

2.3 Decoupling Methods

In this section, two decoupling methods will be reviewed, one based on transfer functions ([152]) and the other based on state space representation ([153]).

2.3.1 Decoupled Methods Based on Transfer Function

Consider the square coupled system $G(z)$ defined as:

$$\begin{bmatrix} Y_1(z) \\ \vdots \\ Y_m(z) \end{bmatrix} = \underbrace{\begin{bmatrix} G_{11}(z) & \dots & G_{1m}(z) \\ \vdots & \ddots & \vdots \\ G_{m1}(z) & \dots & G_{mm}(z) \end{bmatrix}}_{G(z)} \begin{bmatrix} V_1(z) \\ \vdots \\ V_m(z) \end{bmatrix} \tag{2.16}$$

where Y_i and V_i are the $\mathcal{Z}$-transforms of y_i and v_i, respectively. The system $G(z)$ consists of diagonal subsystems with dynamics $G_{ii}(z)$ and off-diagonal (interaction) subsystems with dynamics $G_{ij}(z), i \neq j$. A decoupled system is perfectly diagonal (i.e., each output depends on only one input). The decoupling the system is achieved by adding a filter, $F(z)$, before $G(z)$, so that the product $G(z)F(z)$ yields a diagonal transfer function matrix $W(z) := G(z)F(z)$ ([152]). By doing so, each output Y_i depends only on the new input V_i through: $Y_i(z) = W_{ii}(z)V_i(z)$, where $W_{ii}(z)$ is the i-th diagonal elements of $W(z)$.

In this dissertation, two structures for $W(z)$ are reviewed, which lead to the following two decoupling methods:

- 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 and the inverse filter are defined as:

$$F(z) = G^{-1}(z)W(z), \quad F^{-1}(z) = W^{-1}(z)G(z) \tag{2.17}$$

- Identity Method: The filter $F(z)$ is found such that $W(z)$ equals the identity matrix. The filter and the inverse filter are defined as:

$$F(z) = G^{-1}(z), \quad F^{-1}(z) = G(z) \tag{2.18}$$

Notice that in both methods, the elements of either $F(z)$ or $F^{-1}(z)$ (or both) may be improper transfer functions because of $G^{-1}(z)$ and $W^{-1}(z)$. If this is the case, they cannot be implemented on a LTI systems. In order to make them proper, we multiply $F(z)$ and $F^{-1}(z)$ by time-delays of the form $\frac{1}{z^{\beta}}$, where β refers to how much time delay should be added to make the transfer functions proper.

Remark 1. *In the above discussion, the matrix $W(z)$ is assumed to be diagonal, which implies that every y_i depends only on v_i. This, however, is only one possible structure for $W(z)$. It is also possible to decouple the system by having each y_i depend on one v_j, $j \neq i$. In this case, the $W(z)$ will be constructed such that every row will have only one non-zero element. Similarly, each column will also have one non-zero*

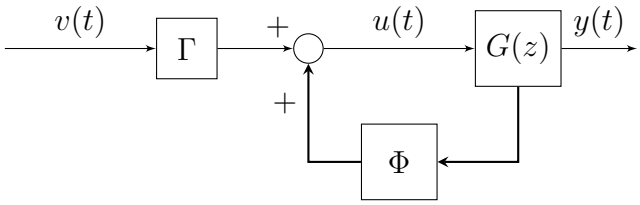

Figure 2.2: State feedback decoupling

element.

2.3.2 Decoupled Methods Based on State Feedback

In this section,the input-output decoupling via state-feedback, as presented in [153, 154], will be described. Consider a discrete-time MIMO coupled system, G, given in state-space form using (2.1). In this section, it is assumed that the input $v \in \mathbb{R}^m$ and output $y \in \mathbb{R}^m$.

In the remainder of this discussion, it is assumed that no direct feed through between v and y (i.e., $D = 0$) as required by [153, 154]. Note that the case where $D \neq 0$ can be handled as well (e.g., see [155]), but for the sake of simplicity, the case where $D = 0$ will only be presented.

The block diagram of state-space decoupling is duplicated in Figure 2.2. The decoupling is achieved by modifying the feedback gain Φ and input factor Γ. More specifically, the substitution of $u = \Phi x + \Gamma v$, where Φ is an $m \times n$ matrix and Γ is an

$m \times m$ matrix, into (2.1) results in:

$$x(t+1) = \underbrace{(A + B\Phi)}_{\bar{A}} x(t) + \underbrace{B\Gamma}_{\bar{B}} v(t), \quad y(t) = Cx(t) \tag{2.19}$$

Let $d_1, d_2, \ldots, d_m$ be defined by:

$$d_i = \min\{j : C_i A^j B \neq 0, j = 0, 1, \ldots, n-1\}$$

where C_i denotes the i-th row of C. If $C_i A^j B = 0$ for all $j = 0, 1, \ldots, n-1$, then we set $d_i = n - 1$. Let $A^* \in \mathbb{R}^{m \times n}$ and $B^* \in \mathbb{R}^{m \times m}$ be defined by:

$$A^* = \begin{bmatrix} C_1 A^{d_1+1} \\ \vdots \\ C_m A^{d_m+1} \end{bmatrix}, B^* = \begin{bmatrix} C_1 A^{d_1} B \\ \vdots \\ C_m A^{d_m} B \end{bmatrix} \tag{2.20}$$

It is proved in [153] that there exist a pair of matrices Φ and Γ that decouple the system from v to y if and only if B^* is nonsingular.

Below, two structures for Φ and Γ are studied, which lead to the following two decoupling methods:

- Identity method: The pair

$$\Phi = -B^{*-1} A^*, \quad \Gamma = B^{*-1} \tag{2.21}$$

  leads to $y_i(t + d_i + 1) = v_i(t)$, which means that the i-th output depends only on the i-th input with one or more time delays.

- Pole-assignment method: We can decouple the system while simultaneously

assigning the poles of the decoupled system by using the following choice of Γ and Φ:

$$\Phi = B^{*-1}\left[\sum_{k=0}^{\delta} M_k C A^k - A^*\right], \quad \Gamma = B^{*-1} \tag{2.22}$$

where $\delta = \max d_i$ and M_k are $m \times m$ diagonal matrices that are designed to assign the poles at specific locations. For more details, please see [153]. Note that not all of the eigenvalues of $\bar{A}$ can be arbitrarily assigned. However, it is shown in [153] that if $m + \sum_{i=1}^{m} d_i = n$, then, all the poles of the decoupled system can be assigned.

2.4 Review of Neural Network Approximation

Discover the explicit input-output mapping of a function is one of the most common performance in Neural Network (NN) construction. A literature review on several NN models to solve function approximation problem is presented in Section 1.3.2. For the sake of simplicity, in this section, feedforward NN with one hidden layer will be reviewed. A simple structural diagram of feedforward NN is shown in Figure 2.3. It can be seen that the NN consists of three layers: input layer, output layer, and, in between, hidden layer. The layers are connected with node, which forms a network. Let the number of neurons in the hidden layer be denoted by N_{hidden}. The input and output of the NN are denoted by $r_{NN} \in \mathbb{R}^{n_r}$ and $y_{NN} \in \mathbb{R}^{n_y}$, respectively. For a feedforward NN model with a single layer, the output can be characterized by

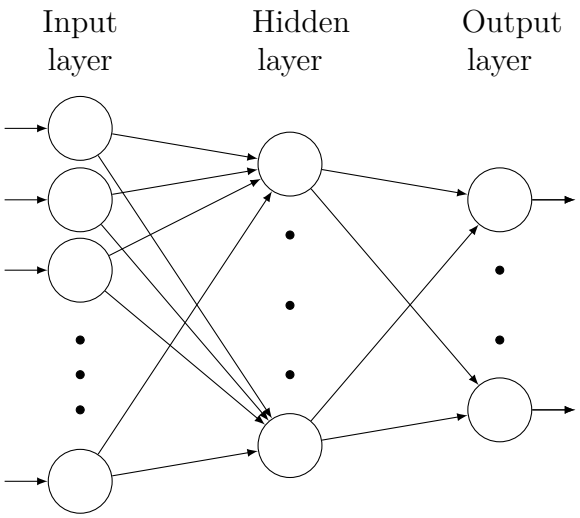

Figure 2.3: A simple structure of Feedforward Neural Network

following:

$$y_{NN} = w_1 \cdot \sigma(w_2 \cdot r_{NN} + b_2) + b1 \tag{2.23}$$

where $w_1 \in \mathbb{R}^{n_y \times N_{hidden}}$ and $b_1 \in \mathbb{R}^{n_y \times 1}$ represent the weight connecting the hidden layer to the output layer and the bias in the output layer, respectively. The $w_2 \in \mathbb{R}^{n_r \times N_{hidden}}$ and $b_2 \in \mathbb{R}^{N_{hidden} \times 1}$ refer to the weight connecting the input layer to the hidden layer and the bias in the input layer, respectively. In (2.23), $\sigma(\cdot)$ represents the activation function. Several common utilized activation function includes:

- Linear function: $\sigma(x) = x$
- Sigmoid function: $\sigma(x) = \frac{1}{1+e^{-x}}$
- ReLU function:

$$\sigma(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Neural Network approaches are trained by presenting sets of input-output data pairs. Then, the internal weights and bias (i.e., w_1, w_2, b_1, and b_2) are optimized to capture the implicit mapping from the inputs v_{NN} to the outputs y_{NN}. This process of adjusting weights and biases, from supplied data, is called training and the used data is called the training set. The training process can be broadly classified into two typical categories: Supervised learning, such as back-propagation, and Unsupervised learning [156]. Back-propagation, which is applied to multilayer perceptrons is the most popular and well studied training algorithm. It is a gradient-descendent method that minimizes the mean square error of the difference between the network outputs and the targets in the training set:

$$J(f) = \frac{1}{M} \sum_{i=1}^{M} \|y_i - y_{\mathrm{NN,i}}\|$$

where v_i and $v_{\mathrm{NN,i}}$ represent the target output and the output given by NN with the same input, respectively. The M represents the number of data samples. Once the NN is trained, it is able to predict the correct outputs corresponding to the "unseen" data.

2.5 Review of Decoupled Reference Governor based on Transfer Function Decoupling (DRG-tf)

In this section, a brief explanation on DRG-tf will be introduced. As shown in [26], DRG-tf, which is a RG-based scheme that can enforce constraints on linear MIMO systems, maintains the highly-attractive computational features of SRG compared to VRG. The block diagram of DRG-tf is illustrated in Figure 2.4. The main idea behinds DRG-tf is based on decoupling the system dynamics ($G(z)$) using the method described in Section 2.3 to obtain a completely diagonal system $W(z)$:

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

Afterwards, m independent scalar reference governors (SRG) for the resulting decoupled subsystems are implemented to ensure the constraints are satisfied, as required, followed by coupling the dynamics using $F(z)^{-1}$ to cancel the steady-space tracking error caused by $F(z)$. As mentioned in above, the $F(z)$ or $F^{-1}(z)$ (or both) may be improper transfer functions and time-delays are added to make the transfer functions proper. Note that if this is the case, the system response will be delayed under the DRG-tf scheme, even if no constraint violation is likely. This is a caveat of the DRG-tf approach; however, if the sample time is small enough, the introduced delay would be

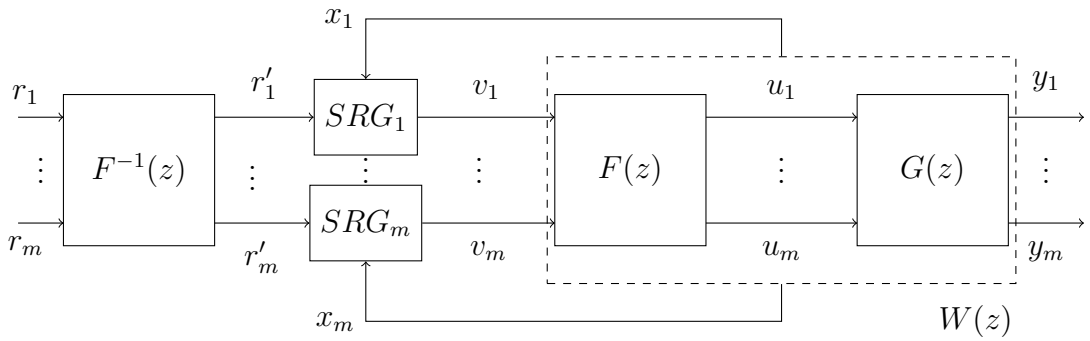

Figure 2.4: Block Diagram for DRG-tf

negligible. Also note that $G^{-1}(z)$ might introduce unstable poles to $F(z)$ or $F^{-1}(z)$, which will cause the system to become unstable. Further assumptions are introduced later to avoid such situations.

The following assumptions are made to construct DRG-tf:

A. 2. *System $G(z)$ in Figure 2.4 reflects the combined closed-loop dynamics of the plant with a stabilizing controller. Consequently, $G(z)$ is asymptotically stable. Furthermore, we assume that all diagonal subsystems of the decoupled system $W(z)$ are also asymptotically stable.*

A. 3. *$G(z)$ in Figure 2.4 is invertible and has a stable inverse.*

A. 4. *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.*

Consider the system in Figure 2.4 with $G(z)$ given in (2.16). To design the m different SRGs, the maximal admissible set (MAS) for each W_{ii}, denoted by $O_\infty^{W_{ii}}$, is required. To obtain these sets, the minimal state-space realization of each subsystem

W_{ii} (i.e., both controllable and observable) is used, and its MAS is computed as:

$$\begin{aligned} O_\infty^{W_{ii}} :=& \{(x_{i_0}, v_{i_0}) \in \mathbb{R}^{n_i+1} : x_i(0) = x_{i_0}, \\ & v_i(t) = v_{i_0}, y_i(t) \in \mathbb{Y}_i, \forall t \in \mathbb{Z}_+\} \end{aligned} \tag{2.24}$$

where x_i and n_i are the state and the order of W_{ii}, respectively. The DRG-tf formulation is based on the sets $O_\infty^{W_{ii}}$. Specifically, the inputs v_i are defined, similar to (2.14), by:

$$v_i(t) = v_i(t-1) + \kappa_i(r_i'(t) - v_i(t-1)) \tag{2.25}$$

where κ_i are computed by m independent linear programs (LP):

$$\begin{aligned} \underset{\kappa_i \in [0,1]}{\text{maximize}} \quad & \kappa_\text{i} \\ \text{s.t.} \quad & v_i(t) = v_i(t-1) + \kappa_i(r_i'(t) - v_i(t-1)) \\ & (x_i(t), v_i(t)) \in O_\infty^{W_{ii}} \end{aligned} \tag{2.26}$$

We showed in [26] and [147] 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 SRG theory. Specifically, for a constant signal $r(t) = r$, $r'(t)$ converges (because of stability of F^{-1}), which implies that $v(t)$ converges (because of stability of SRGs). Thus, the system of Figure 2.4 is guaranteed to be stable.

In [26], two DRG-tf formulations based on different decoupling methods is introduced, namely the diagonal and the identity methods (as explained in Section 2.3). One possible shortcoming of DRG-tf is that, because of the additional $F(z)$ and $F^{-1}(z)$, the tracking performance may be deteriorated. However, as shown in [26], at steady state, the closeness of u and r and, hence, the performance of DRG-tf,

depends on the decoupling filter, $F(z)$ (as shown in Algorithm 1). Additionally, the distillation process example is implemented to illustrate that DRG-tf has a superior performance than SRG and an efficient algorithm to compute DRG-tf is provided.

Besides all the designing and analysis of DRG-tf in [26], the DRG-tf scheme still lacks of a thorough analysis on the transient performance, observer design, and the extensions to systems with disturbance and model mismatch. Also, DRG-tf as proposed above only works on square MIMO system (i.e., the number of inputs is equal to the number of outputs), the modification of DRG-tf to non-square MIMO systems is left to be explored.

Theorem 1. *Given the system of Figure 2.4, 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 and* F_0 *is the DC-gain of* $F(z)$.

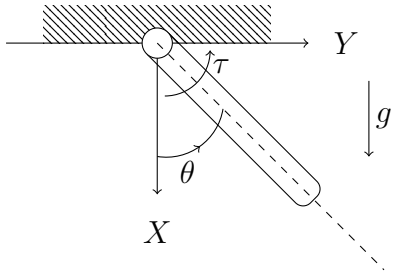

Figure 2.5: One-link arm.

2.6 Review of One-arm Link Robot Dynamics, Roll-over Prevention, and Quadcopter Dynamics

In this section, a review on the one-link arm robot dynamics, vehicle rollover prevention example, and quadcopter dynamics will be given. Also, these three practical examples are used to illustrate the performance of the proposed methods.

2.6.1 One-arm Link Robot

The one-link arm robot, shown in Figure 2.5, has sate space form [157]:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{3g}{2l_r} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{3}{m_r l_r^2} \end{bmatrix} \tau, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.27}$$

where m_r is the mass of the robot arm, g represents the gravitational acceleration, and l_r refers to the arm length. The $x_1 \triangleq \theta$, $x_2 \triangleq \dot{\theta}$ are the states and τ (i.e., external torque) is the control input. To stabilize the one-link arm dynamics, a state-feedback

controller with a pre-compensator is applied to ensure that the output, θ, tracks the desired angle, v, perfectly, that is:

$$\tau = \Phi v - \Gamma \begin{bmatrix} x1 \\ x2 \end{bmatrix}.$$

2.6.2 Vehicle Rollover prevention

A rollover accident may occur during extreme driving scenario, such as excessive speed during cornering, obstacle avoidance, and abrupt lane change maneuvers, where rollover occurs as a result of the wheel forces induced during these maneuvers. It is however, possible to prevent such a rollover incident by monitoring the car dynamics and applying proper steering control. The work in [158] proposes a model for rollover prevention where the input is the steering angle and the performance output is the so-called Load Transfer Ratio LTR, which is related to tire lift-off and can be considered as an indicator of impending vehicle rollover. Before describing the motion of the vehicle model, several notations are needed to be introduced. As presented in [158], the equation of motion corresponding to vehicle model is as follows:

$$\dot{\xi} = A\xi + B\delta, \quad y = C\xi \tag{2.28}$$

Table 2.1: Vehicle Model Parameters

parameter	description	value
m_h	vehicle mass (kg)	1224
v_h	vehicle speed (km/s)	144
σ_h	steering angle (radians)	270240
J_{xx}	roll moment of inertia at CG (kg $\cdot$ m^2)	362.6
J_{zz}	yaw moment of inertia at CG (kg $\cdot$ m^2)	1280
l_v	longitudinal CG position w.r.t. front axle (m)	1.102
l_h	longitudinal CG position w.r.t. rear axle (m)	1.25
T	vehicle track width (m)	1.51
h	distance of CG from roll axis (m)	0.375
c_v	suspension damping coefficient (N $\cdot$ m $\cdot$ s/rad)	4000
k_v	suspension spring stiffness (N $\cdot$ m $\cdot$ s/rad)	36075
C_v	linear tire stiffness for front tire (N/rad)	90240
C_h	linear tire stiffness for rear tire (N/rad)	180000

where $\xi = \begin{bmatrix} v_y & \dot{\Psi} & \dot{\phi} & \phi \end{bmatrix}^T$ corresponds to the lateral velocity, yaw rate, roll rate, and roll angle, respectively. The state space matrix is defined as:

$$A = \begin{bmatrix} -\frac{\sigma_h J_{xeq}}{m_h v_h J_{xx}} & \frac{\rho J_{xeq}}{m_h h J_{xx}} - v & -\frac{hc_v}{J_{xx}} & \frac{h(m_h g h - k_v)}{J_{xx}} \\ \frac{\rho}{J_{zz} v_h} & \frac{-\gamma v_h}{J_{zz} v_h} & 0 & 0 \\ \frac{-h\sigma_v}{J_{xx} v_h} & \frac{h\rho}{J_{xx} v_h} & \frac{-c_v}{J_{xx}} & \frac{m_h g h - k_v}{J_{xx}} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{c_v J_{xeq}}{m_h J_{xx}} & \frac{c_v l_v}{J_{zz}} & \frac{hc_v}{J_{xx}} & 0 \end{bmatrix}^T,$$

$$C = \begin{bmatrix} 0 & 0 & \frac{-2c_v}{m_h g T} & \frac{-2k_v}{m_h g T} \end{bmatrix} \tag{2.29}$$

where $J_{x_{eq}}, \sigma_v, \rho$, γ, and k_v are defined as:

$$J_{x_{eq}} = J_{xx} + m_v h^2, \quad \sigma_h = C_v + C_h,$$

$$\rho = C_h l_h - C_v l_v, \quad \gamma = C_h l_h^2 + C_v l_v^2, \quad k_v = C_v l_v^2 + C_h l_h^2,$$

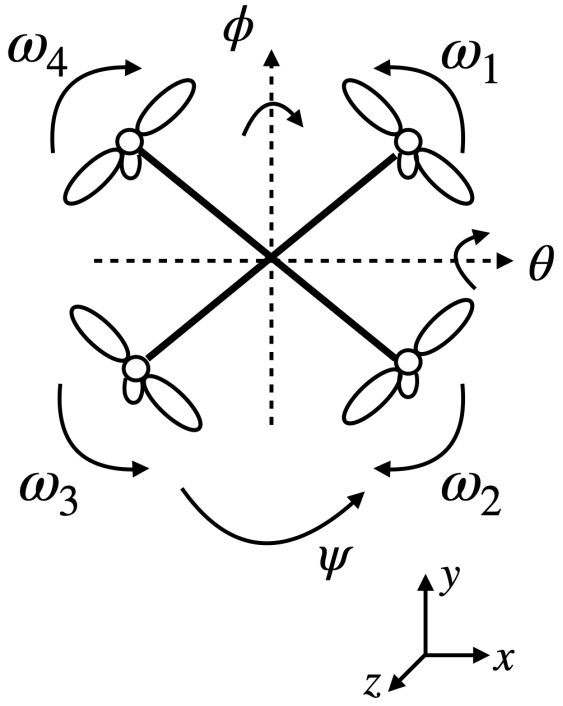

Figure 2.6: The structure of Crazyflie

The output, y, represents the LTR. As shown in [158], to prevent rollover accident, LTR is required to be within $[-1, 1]$.

2.6.3 Quadcopter Dynamics

In this section, a briefly explanation on the modeling and control of the Crazyflie 2.0, which is the experimental platform to examine the proposed control strategies (i.e., NL-DRG, M-RG, and NN-DRG), will be introduced.

Modeling of Quadcopter

A structural diagram of the Crazyflie is shown in Figure 2.6. The dynamical states are the roll angle (ϕ), pitch angle (θ), yaw angle (ψ), roll angular velocity (p), pitch angular velocity (q), and yaw angular velocity (r). The roll, pitch, and yaw angles are defined in the earth-fixed reference, while the angular rates are defined in the body frame. The translation from body-frame to earth-frame can be obtained by [159]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.30}$$

Note that this relationship is only valid when the pitch angle (i.e., θ) is not ± 90 degrees. At $\theta = \pm 90^\circ$, the so-called "Gimbal lock" occurs. In this dissertation, we will constrain θ away from $\pm 90^\circ$ so this is not a limitation. In general, quaternion representation of angles would overcome Gimbal lock.

The actual structure of Crazyflie is almost symmetric around x and y axis but asymmetric in z axis. However, for simplicity, it is still assumed that the inertia matrix is diagonal (also, as will be shown later, the diagonal inertial matrix is sufficient to obtain a satisfactory model of the Crazyflie):

$$J = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{2.31}$$

where I_{xx}, I_{yy}, and I_{zz} represent the inertia matrix in the x axis, y axis, and z

axis, respectively. Using the Newton-Euler equations, the dynamics of the angular velocities can be found to be:

$$\begin{aligned} \dot{p} &= \frac{I_{yy} - I_{zz}}{I_{xx}} qr + \frac{\tau_\phi}{I_{xx}}, \quad \dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}} pr + \frac{\tau_\theta}{I_{yy}} \\ \dot{r} &= \frac{I_{xx} - I_{yy}}{I_{zz}} pq + \frac{\tau_\psi}{I_{zz}} \end{aligned} \tag{2.32}$$

The total thrust and the body torques generated by the propellers can be related to the motor speeds according to [160]:

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \\ f_t \end{bmatrix} = \underbrace{\begin{bmatrix} -lC_T/\sqrt{2} & -lC_T/\sqrt{2} & lC_T/\sqrt{2} & lC_T/\sqrt{2} \\ -lC_T/\sqrt{2} & lC_T/\sqrt{2} & lC_T/\sqrt{2} & -lC_T/\sqrt{2} \\ -C_D & C_D & -C_D & C_D \\ C_T & C_T & C_T & C_T \end{bmatrix}}_{\Gamma} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.33}$$

where d, C_T, and C_D represent the arm length, thrust coefficient, and aerodynamic drag coefficient, respectively. And ω_i, where $i = 1, \ldots, 4$, represents the rotation speed of i-th motor. By using the Newton's law, the altitude dynamics can be written as:

$$\ddot{z} = -g + \frac{f_t}{\cos\phi \cos\theta}.$$

Next, a linearization model of the quadcopter dynamics from the torque and thrust to the outputs of angles and angle rates (i.e., (2.30) and (2.32)) will be introduced. Define u_τ and x as the control input and state vector, respectively (i.e., $u_\tau = [f_t, \tau_\phi, \tau_\theta, \tau_\psi]^T$ and $x = [\dot{z}, p, q, r, z, \phi, \theta, \psi]^T$). The equilibrium point is chosen to be the hover con-

dition:

$$x_e = \begin{bmatrix} 0 & 0 & 0 & 0 & z_e & 0 & 0 & 0 \end{bmatrix}^T \quad u_{\tau_e} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T \tag{2.34}$$

Consequently, the state space form of the linear dynamics model can be described by:

$$\dot{x} = Ax + Bu_\tau \tag{2.35}$$

where

$$A = \begin{bmatrix} 0_{4\times4} & 0_{4\times4} \\ I_{4\times4} & 0_{4\times4} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \\ \hline & & 0_{4\times4} & \end{bmatrix}$$

Note that by applying a Taylor's first order expansion and taking into account the equilibrium state vector specified in (2.34), the transformation from the thrust and torques to the motor speed (as shown in (2.33)) can be linearized as [160]:

$$\begin{bmatrix} f_t \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = 2\sqrt{\frac{mg}{4C_T}}\Gamma \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{2.36}$$

Cascade PID Control of Crazyflie

The overall block diagram of Crazyflie is shown in Figure 2.7. As can be seen, the tracking setpoints are desired roll angle (ϕ_d), pitch angle (θ_d), yaw rate ($\dot{\psi}_d$), and altitude (z_d). Note that the reason for controlling ϕ, θ, and $\dot{\psi}$ is that ϕ and θ are influential for stability and will directly affect the trajectory of the quadcopter. How-

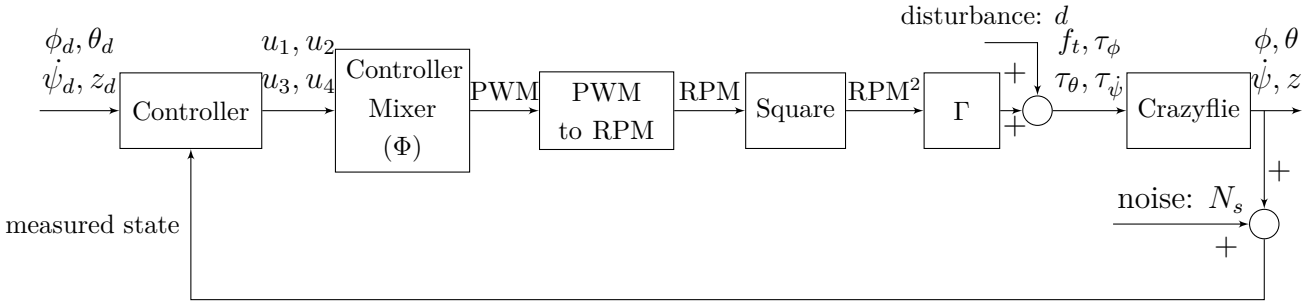

Figure 2.7: Control system block diagram.

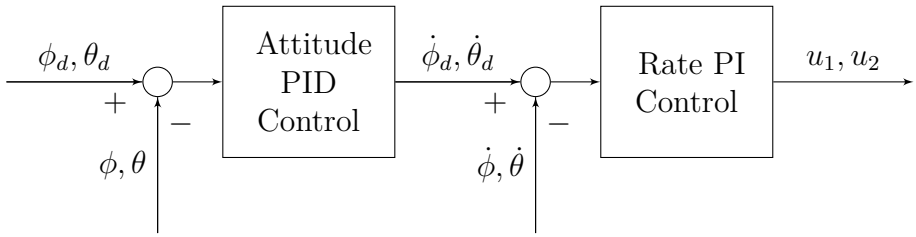

Figure 2.8: Cascaded PID controller for the roll angle and pitch angle. The PID gains for the attitude controller are denoted by $K_{P,\phi}$, $K_{I,\phi}$ $K_{D,\phi}$ and $K_{P,\theta}$, $K_{I,\theta}$, $K_{D,\theta}$ for roll angle and pitch angle, respectively. The rate controller is a PI controller with gains denoted by $K_{P,p}$, $K_{I,p}$ and $K_{P,q}$, $K_{I,q}$ for roll angle and pitch angle, respectively

ever, yaw angle ψ is not as important as other angles and, for the ease of designing, $\dot{\psi}$ is chosen to be controlled instead. The constraints are imposed on the output, namely $\phi \in [-5°, 5°]$, $\theta \in [-5°, 5°]$, and $\dot{\psi} \in [-10°/sec, 10°/sec]$. Detailed information on the controllers is provided in Figures 2.8 to 2.10. The controller gains and quadcopter parameters are presented in TABLE 2.2.

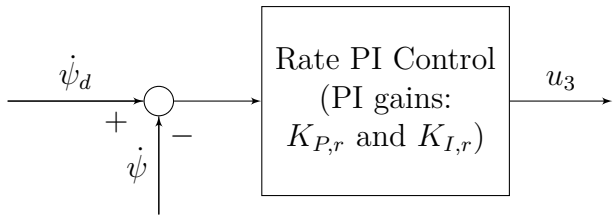

Figure 2.9: The block diagram of the rate controller for $\dot{\psi}$.

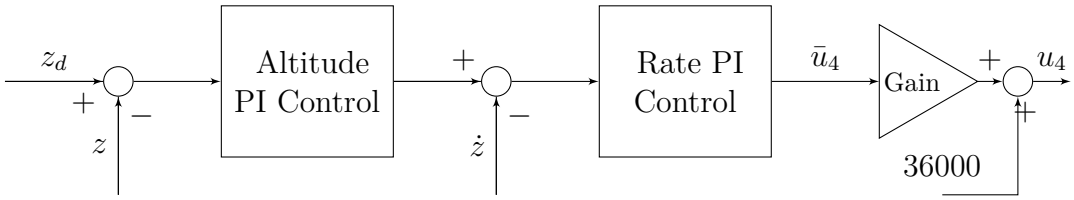

Figure 2.10: The block diagram for altitude controller. Gain $= 1000$*. The bias* 36000 *is used to compensate for the effect of gravity. The gains for the altitude PI controller is denoted by* $K_{P,z}$ *and* $K_{I,z}$*. The rate controller is also a PI controller with gains* $K_{P,\dot{z}}$ *and* $K_{I,\dot{z}}$

Table 2.2: Quadcopter Parameters and PID Gains

parameter	value
g	9.81
m	0.03
l	0.045
C_T	8.511×10^{-7}
C_D	1.221×10^{-8}
I_{xx}	1.138×10^{-5}
I_{yy}	1.138×10^{-5}
I_{zz}	2.95×10^{-5}

parameter	value
$K_{P,\phi} = K_{P,\theta}$	350
$K_{I,\phi} = K_{I,\theta}$	550
$K_{D,\phi} = K_{D,\theta}$	3.5
$K_{P,p} = K_{P,q}$	6
$K_{I,p} = K_{I,q}$	4
$K_{P,r}$, $K_{I,r}$	100, 16.7
$K_{P,z}$, $K_{I,z}$	2, 0.5
$K_{P,\dot{z}}$, $K_{I,\dot{z}}$	25, 15

To move and rotate the quadcopter appropriately, the output of the controller (i.e., u_1, u_2, u_3, and u_4) has to be distributed to four motors using a proper mixing matrix (i.e., Φ in Figure 2.7), which is defined as:

$$\begin{bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{bmatrix} = \underbrace{\begin{bmatrix} -0.5 & -0.5 & -1 & 1 \\ -0.5 & 0.5 & 1 & 1 \\ 0.5 & 0.5 & -1 & 1 \\ 0.5 & -0.5 & 1 & 1 \end{bmatrix}}_{\Phi} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{2.37}$$

where $PWM_i \in [0, 65535], i = 1, \ldots, 4$ represents the Pulse-width modulation (PWM) signal that is used to control the i-th motor. From (2.37), it can be clearly seen that,

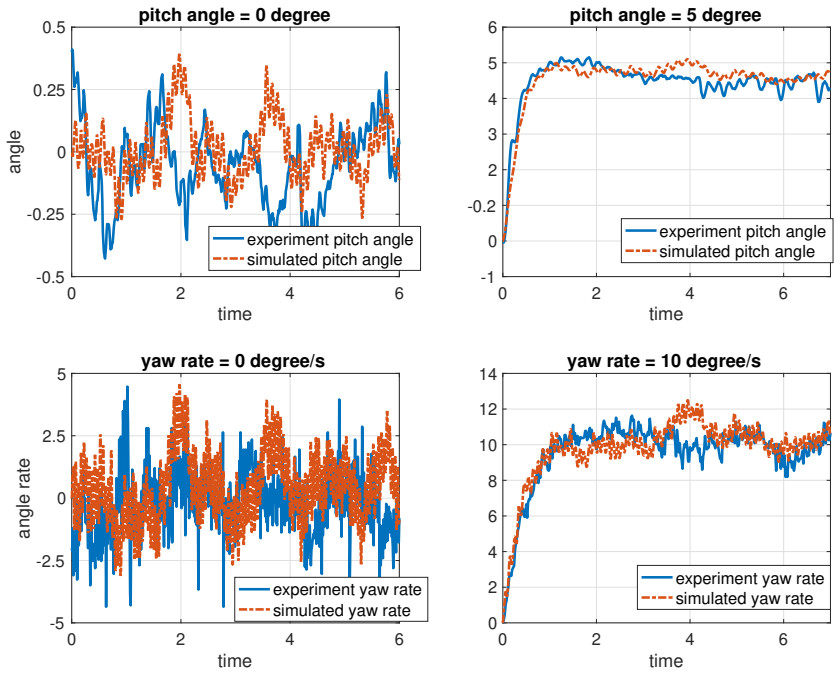

Figure 2.11: Comparison between simulation and experiment results. The desired setpoint is shown in the title of each subfigure.

a command $u_1 > 0$ will be distributed to motors 3 and 4, leading a reduction of the power supplied to motors 1 and 2 and an increase of the power supplied to motors 3 and 4. Thus, a appropriate angular momentum will be obtained in the roll direction (as shown in Figure 2.6). Moreover, the experiments proved that the angular speeds of the motors (i.e., RPM) have a linear relationship with the PWM inputs, following the equation [62]:

$$\text{RPM} = 0.041 \times \text{PWM} + 380.84 \tag{2.38}$$

Finally, N_s in Figure 2.7 is the measurement noise (i.e., IMU noise), while d models the effects of modeling uncertainties and disturbances. To characterize N_s and d for the experimental setup, the hovering data from the Crazyflie is collected.

It is observed that, for the roll angle and pitch angle, the N_s can be modeled by a normally distributed random sequence with mean as 0 and variance as 0.01, and the disturbance d can be modeled by a sine wave with amplitude 100 and frequency 20π. For the yaw rate, similarly, N_s can be modeled by a normally distributed random sequence with mean as 0 and variance as 0.05. The disturbance sine wave for yaw rate has amplitude 50 and frequency 40π. Given these frequencies of d, the disturbance is most likely due to the non-rigid structural modes of the Crazyflie. The comparison between the simulation results and the experiment results conducted on the Crazyflie is shown in Figure 2.11. As can be seen, the model as shown in Figure 2.7 is able to capture the overall behavior (i.e., key features of the response that are of importance for constraint management) of the real Crazyflie.

Chapter 3

Decoupled Reference Governor

The transient analysis and observer design of DRG-tf (as explained in Section 2.5) are presented in this Chapter. Also, the Decoupled Reference Governor based on state-space decoupling (DRG-ss) is introduced. Extensions of DRG-tf and DRG-ss to systems with disturbance and model mismatch, and systems with non-square input/output channels will be provided.

3.1 Decoupled Reference Governor based on Transfer Function Decoupling: DRG-tf

As explained in Section 2.5, DRG-tf is based on decoupling the system dynamics using the method described in Section 2.3 to obtain a completely diagonal system $W(z)$, followed by implementing m independent scalar reference governors (SRGs) for the resulting decoupled subsystems. Finally, to ensure that $u_i = r_i$ when r_i is

constraint-admissible, $F(z)^{-1}$ is introduced to cancel the effects of $F(z)$ while also guarantee that u_i and r_i are close if r_i is not constraint-admissible. Because of the additional $F(z)$ and $F^{-1}(z)$, one challenge with DRG-tf is quantifying the transient performance of the system dynamics. State estimation is another challenge. More specifically, how can the states of the decoupled subsystems be obtained and fed back to the SRGs? In the following section, we elaborate on DRG-tf with a special focus on the above challenges.

3.1.1 Observer design

In this section, we consider the case where the states of W_{ii} (see Figure 2.4) are not measured. Consequently, an observer will be required to estimate the states. One option is to use an open loop observer for each W_{ii}. To explain, let $(A_{ii}, B_{ii}, C_{ii}, D_{ii})$ be a minimal realization of W_{ii}. An open loop observer can be designed by computing the state estimate recursively:

$$\hat{x}_i(t+1) = A_{ii}\hat{x}_i(t) + B_{ii}v_i(t) \tag{3.1}$$

where $\hat{x}_i$ is the estimate of the state x_i. In real-time, the SRGs in the DRG-tf formulation use $\hat{x}_i$ instead of x_i. Note that the open loop observer works properly only when the system model and the initial conditions are both accurately known, which is not always hold in real scenario.

To improve upon the open loop observer, feedback can be implemented from the measured output, as is done in standard observer design. We consider two observer design strategies below. The first approach assumes that all y_i are measured, which

leads to m decoupled observers, and the second assumes that some y_i are not measured, necessitating a centralized observer.

Decoupled observers:

To begin with, suppose that all y_i are available for measurement. In this case, m decoupled Luenberger observers can be designed as follows:

$$\hat{x}_i(t+1) = A_{ii}\hat{x}_i(t) + B_{ii}v_i(t) + L_i(y_i(t) - C_{ii}\hat{x}_i(t) - D_{ii}v_i(t)) \tag{3.2}$$

where L_i is designed to assign the eigenvalues of $A_{ii} - L_iC_{ii}$ in the unit circle (since discrete time LTI systems is considered). In the real-time implementation of DRG-tf, the state that feeds back to SRG_i is $\hat{x}_i$.

A challenge with the decoupled observer strategy explained above is that of selecting the initial conditions for each $\hat{x}_i$. Indeed, if the observers are not initialized properly, the DRG-tf scheme may not be able to enforce the constraints. We provide a solution to this problem below, for the case where the initial condition of $G(z)$, denoted by x_0, is known precisely. We will treat the case of unknown x_0 later.

Our solution is to modify the input to $G(z)$ so that the effects of x_0 is explicitly canceled. To see how this can be done, note that the output of $G(z)$ with initial condition x_0 can be written as:

$$y(t) = CA^tx_0 + (C(I - A^t)(I - A)^{-1}B + D)u_0$$

where A, B, C, and D are the state space matrices of $G(z)$. Denote by $M(z)$ the

$\mathcal{Z}$-transform of CA^t for the sake of simplicity of notation. Note that $M(z)$ represents the initial condition response of the system. In order to get $Y(z) = W(z)V(z)$ with 0 initial condition, where W is a desired diagonal matrix as before, we define $U(z)$ as:

$$U(z) = F(z)V(z) - G^{-1}(z)M(z)x_0 \tag{3.3}$$

where $F(z) = G(z)^{-1}W(z)$. This will effectively cancel the initial conditions and result in a completely decoupled system with respect to the system dynamics and initial condition. The observers given in (3.2) with 0 initial condition now be applied as before. Note that the inverse filter $F^{-1}(z)$ in Figure 2.4 need not be altered.

Centralized observer

Now consider the more interesting case, where some y_i (see Figure 2.4) are not measured. Since the dynamics from governed input v to output y are still required to be decoupled, m SRGs can still be implemented in the DRG-tf formulation. However, since independent measurements on y_i are not available, we can not design m decoupled observers for each W_{ii} as mentioned before, and must instead design one centralized observer for W. This, in turn, implies that the SRGs must use one MAS different from (2.24). To elaborate on these ideas, let $y(t)$, as before, denote the constrained output vector, and the measured output vector be denoted by y_m . Let (A, B, C, D), (A, B, C_m, D_m), and (A_f, B_f, C_f, D_f) be realizations of $G(z)$ from u to y, $G(z)$ from u to y_m, and $F(z)$, respectively. Note that the states of $F(z)$, denoted by x_f, are known at the time of implementation so they do not need to be estimated. To estimate the states of $G(z)$, x, an observer is designed using feedback on the

measurements y_m:

$$\hat{x}(t+1) = A\hat{x}(t) + Bu(t) + L(y_m(t) - C_m\hat{x}(t) - D_m u(t)) \tag{3.4}$$

Using the above, the states of the entire system $W(z)$, i.e., $x_w = (x_f, x)$, can be estimated by $\hat{x}_w = (x_f, \hat{x})$. Note that initialization of this observer is simple if the initial condition of $G(z)$, i.e., x_0 is known: in this case, the initial condition of the observer is set to $(0, x_0)$.

Recall that to construct $O_\infty^{W_{ii}}$ (see (2.24)), the state-space model of the i-th diagonal subsystem of W, W_{ii}, is required. However, the states of each individual W_{ii} is not directly available from the centralized observer, which is why the SRGs can no longer use the $O_\infty^{W_{ii}}$ set. To solve this issue, the following realization of W, which is the augmented dynamics of $F(z)$ and $G(z)$, is use :

$$\begin{aligned} x_w(t+1) &= \underbrace{\begin{bmatrix} A_f & 0 \\ BC_f & A \end{bmatrix}}_{A_w} x_w(t) + \underbrace{\begin{bmatrix} B_f \\ BD_f \end{bmatrix}}_{B_w} v(t) \\ y(t) &= \underbrace{\begin{bmatrix} DC_f & C \end{bmatrix}}_{C_w} x_w(t) + \underbrace{DD_f}_{D_w} v(t) \end{aligned} \tag{3.5}$$

Using (3.5), the state-space model of W_{ii} is given by: $(A_w, B_w(:,i), C_w(i,:), D_w(i,i))$, where $B_w(:,i)$ is the i-th column of B_w, $C_w(i,:)$ is the i-th row of C_w, and $D_w(i,i)$ is the (i,i)-th element of D_w. Thus, we construct $O_\infty^{W_{ii}}$ based on the state-space realization $(A_w, B_w(:,i), C_w(i,:), D_w(i,i))$ and, for real-time implementation, each SRG uses the state of the *entire* system (i.e., $\hat{x}_w = (x_f, \hat{x})$) as feedback.

Finally, for the case where the initial conditions are not known, either observer ((3.2) or (3.4)) can still be used to estimate the states; however, during the transient

phase of the observer, the states may be incorrect, which may lead to constraint violation. To remedy this issue, one can "robustify" $O_\infty^{W_{ii}}$, or alternatively, one could allow the transients to subside before running the system with the DRG-tf.

3.1.2 Analysis of DRG-tf

In this section, the steady-state analysis of DRG-tf [26] is extended and the transient performance of DRG-tf is explored. The analysis of this section relies on the H_∞ and L_1 norm of $F(z)$. Because of the delays introduced in $F(z)$ and/or $F^{-1}(z)$ to make them proper, care must be taken in interpreting the results, as we show below.

Theorem 2. *For the system of Figure 2.4, the following relationship holds:*

$$\|u(t+\beta_1) - r(t-\beta_2)\|_{L_2} \leq \|F\|_{H_\infty} \|v - r'\|_{L_2} \tag{3.6}$$

where β_1 and β_2 are the number of delays added to make F and F^{-1} proper, respectively.

Proof. By Parseval's theorem, $\|u - r\|_{L_2} = \|U - R\|_{H_2}$ and $\|v - r'\|_{L_2} = \|V - R'\|_{H_2}$. where R', R, U, and V are the $\mathcal{Z}$-transforms of r', r, u, and v, respectively. From Figure 2.4 the following equations hold:

$$U(z) = \frac{1}{z^{\beta_1}} F(z) V(z), R'(z) = \frac{1}{z^{\beta_2}} F(z)^{-1} R(z) \tag{3.7}$$

Then,

$$\begin{aligned}
&\|z^{\beta_1}U(z) - z^{-\beta_2}R(z)\|_{H_2}^2 \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} \|F(e^{jw})((V(e^{jw}) - R'(e^{jw}))\|_2^2 dw \\
&\leq \frac{1}{2\pi}\int_{-\pi}^{\pi} (\|F(e^{jw})\|_2 \|V(e^{jw}) - R'(e^{jw})\|_2)^2 dw
\end{aligned}$$

where $\|.\|_2$ refers the Euclidean norm. Since $\|F\|_{H_\infty} = \max_w \bar{\sigma}(F(e^{jw}))$, we have that:

$$\begin{aligned}
&\frac{1}{2\pi}\int_{-\pi}^{\pi} (\|F(e^{jw})\|_2 \|V(e^{jw}) - R'(e^{jw})\|_2)^2 dw \\
&\leq \|F\|_{H_\infty}^2 \|V - R'\|_{L_2}^2
\end{aligned} \tag{3.8}$$

By Parseval's theorem, the result follows. □

Note that (3.6) can be rewritten as:

$$\|u(t) - r(t - \beta_2 - \beta_1)\|_{L_2} \leq \|F\|_{H_\infty} \|v - r'\|_{L_2}$$

This equation shows that the average distance between u and the delayed version of r is bounded by the average distance between v and r' scaled by $\|F\|_{H_\infty}$. Thus, if $\|F\|_{H_\infty}$ is small, the DRG-tf and VRG will perform similarly in transient.

Above algorithm only discusses the performance of DRG-tf in time averages, below, another theorem is provide to demonstrate that the peak of the distance between u and r is related to $\|f\|_{L_1}$, where f is the impulse response matrix of $F(z)$ and $\|f\|_{L_1}$ refers to the L_1 norm of f.

Theorem 3. *For the system of Figure 2.4, the following relationship holds with re-*

spect to the L_1 norm:

$$\|u(t+\beta_1) - r(t-\beta_2)\|_{L_\infty} \leq \|f\|_{L_1} \|v - r'\|_{L_\infty} \tag{3.9}$$

Proof. Based on the inverse $\mathcal{Z}$-transform of (3.7), we have:

$$\begin{aligned} &|u_i(t+\beta_1) - r_i(t-\beta_2)| = |f_i(t) * v(t) - f_i(t) * r'(t)| \\ &= |\sum_{\tau=-\infty}^{\infty} \sum_{j=1}^{m} f_{ij}(\tau)(v_j(t-\tau) - r'_j(t-\tau))| \\ &\leq \sum_{\tau=-\infty}^{\infty} \sum_{j=1}^{m} |f_{ij}(\tau)(v_j(t-\tau) - r'_j(t-\tau))| \\ &\leq \|v - r'\|_{L_\infty} \sum_{\tau=-\infty}^{\infty} \sum_{j=1}^{m} |f_{ij}(\tau)| \end{aligned} \tag{3.10}$$

where f_{ij} refers to the ij-th element of f, $*$ denotes the convolution operator, and in the last inequality, we have used the fact that $\|v - r'\|_{L_\infty}$ is the maximal value of $|v_j(t) - r'_j(t)|$ over j and over t. Taking the maximum of both sides of the above with respect to i, we get:

$$\max_i |u_i(t+\beta_1) - r_i(t-\beta_2)| \leq \|f\|_{L_1} \|v - r'\|_{L_\infty} \tag{3.11}$$

and the result follows. □

This theorem implies that if $\|f\|_{L_1}$ is small, then the DRG-tf will perform similarly to VRG in transient. If, however, $\|f\|_{L_1}$ is large, no conclusion can be drawn.

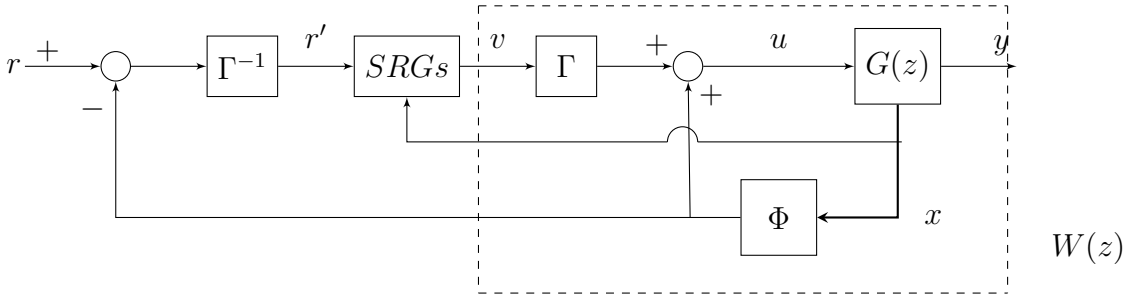

Figure 3.1: DRG-ss block diagram. r, r', v, u, y represent $[r_1, r_2, \ldots, r_m]^T$, $[r'_1, r'_2, \ldots, r'_m]^T$, $[v_1, v_2, \ldots, v_m]^T$, $[u_1, u_2, \ldots, u_m]^T$, and $[y_1, y_2, \ldots, y_m]^T$, respectively.

3.2 Decoupled Reference Governor based on State Feedback Decoupling: DRG-ss

Decoupled Reference Governor based on State Feedback Decoupling, namely DRG-ss, is based on decoupling the input-output dynamics as shown in Figure 3.1 by using state feedback, where the feedback matrices Φ and Γ are properly chosen as discussed in Section 2.3. Similar to DRG-tf, then, m decoupled SRGs, whose goal is the same as the SRGs in DRG-tf, is introduced. Finally, to make sure that $u_i = r_i$ when r_i is constraint-admissible, x is fed back through $\Gamma^{-1}(r - \Phi x)$. The following assumptions are made for the development of DRG-ss:

A. 5. *Similar to A. 2, system $G(z)$ in Figure 3.1 is asymptotically stable.*

A. 6. *B^* matrix in* (2.20) *is nonsingular.*

Consider the system in Figure 3.1, where state feedback decoupling method is applied to get a diagonal system, W, which has state space form $(\bar{A}, \bar{B}, C, 0)$ given

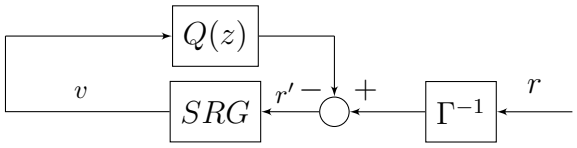

Figure 3.2: Rearrangement of Figure 3.1.

by (2.19). Note that the feedthrough matrix D is assumed to be 0 as discussed in Section 2.3.2, but this assumption can be relaxed. A state-space realization for each decoupled subsystem, W_{ii}, is given by: $(\bar{A}, \bar{B}(:,i), C(i,:), 0)$, where $\bar{B}(:,i)$ is the i-th column of $\bar{B}$, and $C(i,:)$ is the i-th row of C. Next, for each decoupled subsystem, the MAS , denoted by $O_\infty^{W_{ii}}$, is constructed as:

$$\begin{aligned} O_\infty^{W_{ii}} :=& \{(x_{w_0}, v_{i_0}) \in \mathbb{R}^{n+1} : x_{w_0} = x_w(0), \\ & v_i(t) = v_{i_0}, y_i(t) \in \mathbb{Y}_i, \forall t \in \mathbb{Z}_+\} \end{aligned} \tag{3.12}$$

where x_w represents the state of W. As for implementation, the SRGs within DRG-ss compute the inputs, v_i, to the decoupled system the same as (2.25) and κ_i is computed using the same LP as shown in (2.26).

Because of the additional feedback loop (i.e., $-\Phi x$ shown in Figure 3.1), the stability of DRG-ss is not guaranteed (unlike DRG-tf). Below, we provide a sufficient condition for stability of the DRG-ss scheme.

The block diagram of DRG-ss (Figure 3.1) can be rearranged as shown in Figure 3.2, where

$$Q(z) = \Gamma^{-1}\Phi(I - G_x(z)\Phi)^{-1}G_x(z)\Gamma$$

and $G_x(z) = (zI - A)^{-1}B$. From Small Gain Theorem ([161]), if there exist four

constants J_1, J_2, K_1, and K_2, with $J_1 J_2 < 1$, such that:

$$\|v\| \leq K_1 + J_1\|r'\|, \qquad \|Q(z)v\| \leq K_2 + J_2\|v\| \tag{3.13}$$

then, the system is bounded input bounded output stable (i.e., BIBO). While $\|\cdot\|$ can be chosen to be any signal norm, we use the ∞-norm in the discussion that follows. Recall that in the SRG optimization (2.26), κ_i satisfies: $0 \leq \kappa_i \leq 1$, which implies that:

$$\begin{aligned}\|v(t)\| &= \|v(t-1) + K(r'(t) - v(t-1))\|_\infty \\ &\leq \|(I-K)v(t-1)\|_\infty + \|Kr'(t)\|_\infty \\ &\leq \|v(t-1)\|_\infty + \|r'(t)\|_\infty\end{aligned}$$

where K is diagonal matrix with κ_i as its main-diagonal elements. Since v is bounded (because $O_\infty^{W_{ii}}$ is compact), we have that $\|v(t-1)\|_\infty \leq \bar{M}$ for some $\bar{M} > 0$. Thus, $\|v(t)\|_\infty \leq \bar{M} + \|r'\|_\infty$ (i.e., $J_1 = 1$, $K_1 = \bar{M}$). Then, from small gain theorem, the system is BIBO stable if there exist a K_2 and $J_2 < 1$, such that: $\|Q(z)v\|_\infty \leq K_2 + J_2\|v\|_\infty$. Recall that the induced system norm $\|q\|_{L_1}$, where q is the impulse response matrix of $Q(z)$, is defined as: $\|q\|_{L_1} = \sup \frac{\|Qv\|_\infty}{\|v\|_\infty}$. Then, for J_2 to exist, the following inequality needs to be satisfied:

$$\|q\|_{L_1} < 1$$

In summary, the DRG-ss scheme is BIBO stable if $\|q\|_{L_1} < 1$. It is important to note that $Q(z)$ depends on Γ and Φ. Thus, stability must be checked after Φ and Γ have been designed, which means that iterations might be needed if the stability condition above is not satisfied. Finally, asymptotic stability can also be proved by

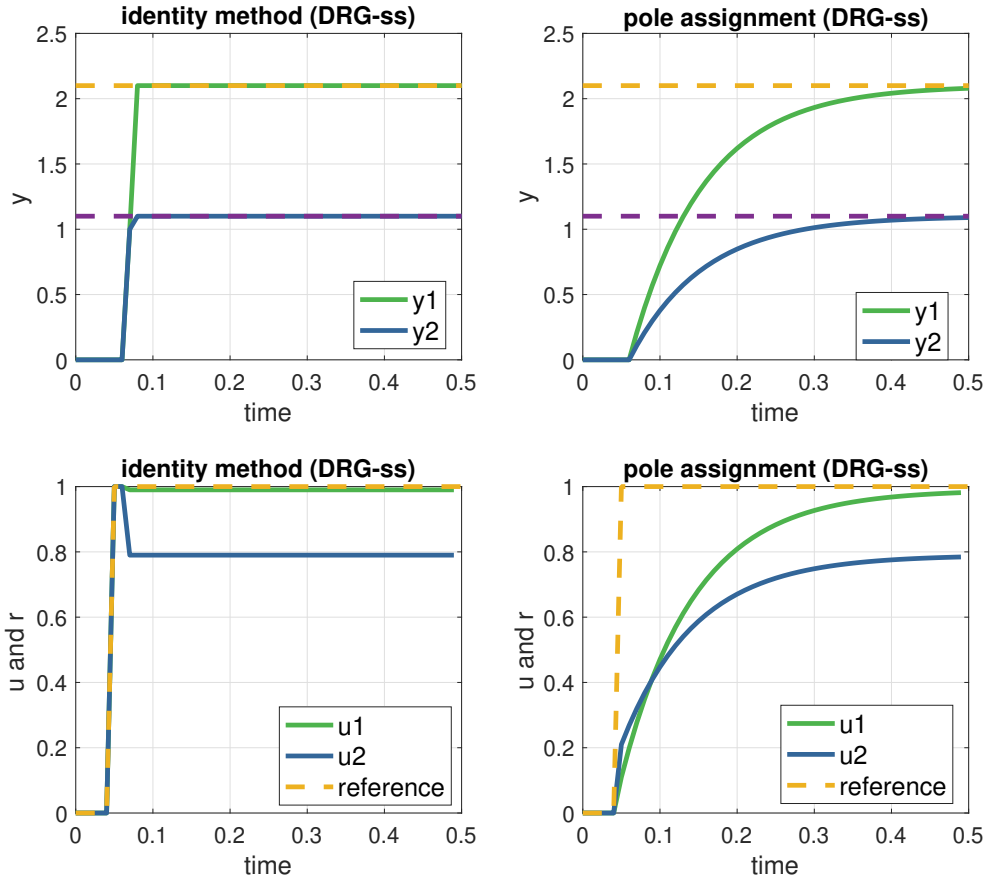

Figure 3.3: Simulation results of DRG-ss. The purple and yellow dashed lines on the top two plots represent the constraints on the outputs.

applying the results from absolute stability ([162]) to the system of Figure 3.2 and using the fact that $0 \leq \kappa_i \leq 1$.

3.2.1 Illustration Example

In this section, an example will be provided to demonstrate the performance of DRG-ss, where the two decoupling methods in Section 2.3.2 are applied to decouple the

system. Consider the system G given by:

$$A = \begin{bmatrix} 0.1 & 1 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.14}$$

The pair (2.21) and (2.22) are used to find Φ and Γ, and proceed to compute $O_\infty^{W_{11}}$ and $O_\infty^{W_{22}}$ based (3.12) (for the identity and pole assignment methods, respectively). Note that for pole assignment method, we choose $M_k = \text{diag}(0.9, 0.9)$ to locate two of the poles of W at 0.9. The constraint set is defined as $\mathbb{Y} := \{(y_1, y_2) : y_1 \leq 2.1, y_2 \leq 1.1\}$. We simulate the response of this system to a step of size 1 in both r_1 and r_2. The simulation results are depicted in Figure 3.3.

Figure 3.3 (top) reveals that the outputs are within the constraints for both identity and pole assignment methods. Note that, from the bottom plots of Figure 3.3, there is a gap between u and r. Later, this gap will be investigated.

As a final remark, similar to the identity method for DRG-tf, while the identity method for DRG-ss is simpler and computationally superior to the pole assignment method, it may lead to large oscillations for underdamped systems [26].

3.2.2 Analysis of DRG-ss

In this section, similar to the analyses of DRG-tf, the steady-state and transient performance of DRG-ss will be analyzed.

Steady-state Analysis

Motivating by the steady-state halfspace in (2.5) for a generic system, the steady-state constraint for $O_\infty^{W_{ii}}$ can be defined similarly. More specifically, the projection of the steady-state halfspace onto the v_i coordinate is considered, which results in:

$$V_{ss}^{W_{i,i}} := \{v_i \in \mathbb{R} : W_{ii_0} v_i \in \mathbb{Y}_{i,ss}\} \tag{3.15}$$

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

$$V_{ss}^{W} := V_{ss}^{W_{1,1}} \times V_{ss}^{W_{2,2}} \times \cdots \times V_{ss}^{W_{m,m}} \tag{3.16}$$

We now compare the above set with the steady-state constraint-admissible input set of system G, i.e., the projection of the steady-state halfspace onto the u coordinate, which arises in Vector Reference Governor (VRG) applications. This set, noted by U_{ss}, is defined by:

$$U_{ss} := \{u \in \mathbb{R}^m : G_0 u \in \mathbb{Y}_{ss}\}. \tag{3.17}$$

Below, we present a theorem to relate U_{ss} and V_{ss}^W.

Theorem 4. *For the system of Figure 3.1, and U_{ss} and V_{ss}^W defined in* (3.17) *and* (3.16)*, the following relation holds*

$$V_{ss}^{W} = C(I-\bar{A})^{-1}\bar{B}(C(I-A)^{-1}B)^{-1} \times U_{ss} \tag{3.18}$$

where $\bar{A} = A + B\Phi$ and $\bar{B} = B\Gamma$.

Proof. Given the state-space realization $(A, B, C, 0)$ for $G(z)$, the DC-gain of G from u to y is given by $G_0 = C(I - A)^{-1}B$. Similarly, the DC-gain of W from v to y is given by $W_0 = C(I - (A + B\Phi))^{-1}B\Gamma$. Therefore, the relationship between W_0 and G_0 is as follows:

$$W_0 = C(I - (A + B\Phi))^{-1}B\Gamma(C(I - A)^{-1}B)^{-1} \times G_0$$

The proof follows from the definitions of U_{ss} and V_{ss}^W. □

This theorem shows that if r is not admissible with respect to system G (i.e., $r \notin U_{ss}$), then, after feeding through Γ^{-1}, r' must also not be admissible with respect to the system W (i.e., $r' \notin V_{ss}^W$).

Before, we mentioned one requirement for DRG, which was u and r should be as close as possible. From Figure 3.1, we see that v and r' are as close as possible, but u and r may not be close. Below, we provide a theorem to quantify the closeness of u and r in steady state.

Theorem 5. *For the system of Figure 3.1, the following relation holds at steady state:*

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

where $\|.\|$ refers to any vector norm and its associated induced matrix norm.

Proof. At steady state, we have that $u = \Gamma v + \Phi x$ and $r = \Gamma r' + \Phi x$. Therefore: $\|u - r\| = \|\Gamma v - \Gamma r'\| = \|\Gamma(v - r')\| \leq \|\Gamma\|\|v - r'\|$. This proves the right hand inequality. To show the left hand inequality, write $\|v - r'\| = \|\Gamma^{-1}u - \Gamma^{-1}r\| =$

$\|\Gamma^{-1}(u-r)\| \leq \|\Gamma^{-1}\|\|u-r\|$. This can be re-written as $\|\Gamma^{-1}\|^{-1}\|v-r'\| \leq \|u-r\|$, which concludes the proof. □

The above theorem shows that $\|u-r\|$ is bounded above and below by $\|v-r'\|$ scaled by $\|\Gamma\|$ and $\|\Gamma^{-1}\|^{-1}$, which are known *a-priori* and can be changed based on different design requirements. More specifically, if $\|\Gamma\|$ is small, then, small $\|v-r'\|$ implies small $\|u-r\|$, which is desirable. Also, if $\|\Gamma^{-1}\|^{-1}$ is large, then small $\|v-r'\|$ implies large $\|u-r\|$, which is undesirable. In the case of large $\|\Gamma\|$ or small $\|\Gamma^{-1}\|^{-1}$, no definite conclusion can be made. Note that the steady-state analysis of v is similar to that in DRG-tf (see [26]), except that instead of having $r' = F_0^{-1}r$ in DRG-tf, we have $r' = \Gamma^{-1}(r - \Phi x)$ in DRG-ss.

Transient Analysis

Recall from Figure 3.1 that the following relationship holds:

$$r' = \Gamma^{-1}(r - \Phi x), \quad u = \Gamma v + \Phi x \tag{3.19}$$

From these equations, the following theorem emerges.

Theorem 6. *For the system in Figure 3.1, the following inequalities hold:*

$$\|u-r\|_{L_2} \leq \sqrt{\sum_{i,j} \Gamma_{ij}^2} \times \|v-r'\|_{L_2} \tag{3.20}$$

$$\|u-r\|_{L_\infty} \leq m \times \max_{i,j} |\Gamma_{ij}| \times \|v-r'\|_{L_\infty} \tag{3.21}$$

where Γ_{ij} *is the* ij*-th element of* Γ.

Proof. From (3.19), the following equation holds: $u - r = \Gamma(v - r')$. Then,

$$\|u - r\|_{L_2}^2 = \|\Gamma(v - r')\|_{L_2}^2 = \sum_{t=0}^{\infty} \sum_{i=1}^{m} (\Gamma_i(v(t) - r'(t)))^2$$

where Γ_i refers to the i-th row of Γ. By Cauchy-Schwarz inequality, we have:

$$\sum_{t=0}^{\infty} \sum_{i=1}^{m} (\Gamma_i(v(t) - r'(t)))^2 \leq \sum_{t=0}^{\infty} \sum_{i=1}^{m} \|\Gamma_i\| \|v(t) - r'(t)\|$$
$$\leq \sum_{i=1}^{m} \|\Gamma_i\| \sum_{t=0}^{\infty} \|v(t) - r'(t)\| = \sum_{i,j} \Gamma_{ij}^2 \|(v - r')\|_{L_2}^2$$

Taking the square root of both sides proves (3.20). Next, we will show the proof of (3.21). We have that:

$$\|u - r\|_{L_\infty} = \|\Gamma(v - r')\|_{L_\infty} = \sup_{t \geq 0} (\max_i |\Gamma_i(v - r')|)$$
$$\leq \sup_{t \geq 0} (\max_{i,j} |m\Gamma_{ij}|)(\max_i |(v - r')|) = \max_{i,j} |m\Gamma_{ij}| \|(v - r')\|_{L_\infty}$$

Then, (3.21) follows. □

The theorem presents the relationship between $v - r'$ and $u - r$ and shows that if the elements of Γ are small, then the distance between u and r would also be small. This implies that tracking will not be significantly deteriorated as compared with VRG.

3.3 Robust DRG

In previous discussion, the $G(z)$ is assumed to be not affected by disturbances and noise, which is unlikely to happen in real world application. In Section 2.1.1, a brief explanation of how SRG can be modified to handle systems affected by unknown disturbances and sensor noise is explained. Essentially, MAS is "robustified" (i.e., shrunk) to account for the worst-case realization of the disturbances. In this section, we extend these ideas to DRG-tf and DRG-ss, where we show that an initial pre-processing is required to have the system in the form (2.9). Secondly, we consider the scenario where the system model is uncertain, where we present an innovative solution for handling these systems.

3.3.1 DRG for Systems with Unknown Disturbances

In this section, the robustified DRG to systems with unknown but bounded disturbance is presented.

DRG-tf for systems with unknown disturbances

Assumed that the system (2.16) is affected by an unknown disturbance $d(t) \in \mathbb{R}^d$:

$$Y(z) = G(z)U(z) + G_w(z)D(z) \tag{3.22}$$

where $D(z)$ is the $\mathcal{Z}$-transform of $d(t)$. Consistent with the literature of SRG, it is assumed that $d \in \mathbb{D}$, where $\mathbb{D}$ is a compact polytopic set.

In this section, DRG-tf with the diagonal decoupling method explained in Section 2.5 is considered (the identity decoupling method can be applied similarly). Under Assumption A.3, the filter $F(z)$ defined in (2.17) is characterized. This leads to each y_i described by:

$$Y_i(z) = G_{ii}(z)V_i(z) + \sum_{j=1}^{d} G_{w_{ij}}(z)D_j(z)$$

is decoupled from input v to output y, but not from disturbance d to output y. To address this issue, the dynamics of each y_i is first converted to state-space form:

$$\begin{aligned} x_i(t+1) &= A_i x_i(t) + B_i v_i(t) + B_{w_i} d(t) \\ y_i(t) &= C_i x_i(t) + D_{w_i} d(t) \in \mathbb{Y}_i \end{aligned} \tag{3.23}$$

For each subsystem (3.23) we now proceed to compute the corresponding robust MAS using the procedure described in Section 2.1.1. Other than the modification of MAS, the implementation process of DRG-tf remains unchanged.

Below, a illustration example will be presented to demonstrate the performance of robust DRG-tf. We consider the following system:

$$G(z) = \begin{bmatrix} \frac{0.9}{(z-0.2)^2} & \frac{0.05}{3z+1} \\ \frac{3}{(2z-1)^2} & \frac{0.4}{z-0.6} \end{bmatrix} \tag{3.24}$$

and

$$G_w(z) = \begin{bmatrix} \frac{0.2}{(z-0.5)^2(3z+1)} \\ \frac{0.3}{(2z+1)(z-0.7)^2} \end{bmatrix}$$

The constraints are imposed on the outputs so that $-1.2 \leq y_1 \leq 1.2$ and $-3.9 \leq y_2 \leq 3.9$. The robust DRG-tf is implemented for this system assuming that the disturbance satisfies $d \in \mathbb{D} := [-0.1, 0.1]$. For the purpose of simulations, the disturbance is generated randomly and uniformly from the interval $[-0.1, 0.1]$.

The results of DRG-tf with disturbance are shown in Figures 3.4. In the top subplots, "y_1 coupled" and "y_2 Coupled" refer to the response of the system without DRG-tf (i.e., r applied to G directly), which shows that, without the DRG-tf implemented, the constraints are violated, as expected. These results confirm that DRG-tf is able to satisfy the constraints in the presence of disturbances. As can be seen from the plots, the disturbance affects both outputs (the outputs appear noisy). Interestingly, the disturbance does not affect u for DRG-tf (see Figure 3.4), but it affects u for DRG-ss (see u in Figure 3.5). The reason for this behavior can be explained as follows: it can be seen from Figure 3.1 that the outer feedback in DRG-ss may transmit the effects of disturbances and sensor noise to r'. As a result of this, the effect of the disturbance on the output may be higher in DRG-ss than in DRG-tf. This may be a decisive argument to select between DRG-ss and DRG-tf, since the latter does not show this type of behavior.

DRG-ss for systems with unknown disturbances

To decouple the system (2.9) from the inputs u to the outputs y, the pole assignment decoupling method explained in Section 2.3 is proposed; similar results can be obtained for the identity decoupling method. The decoupled system to characterized

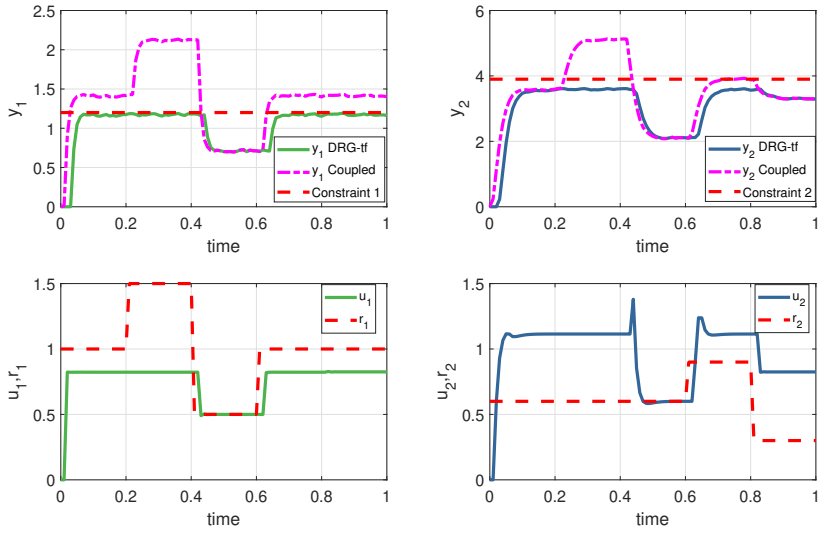

Figure 3.4: DRG-tf with disturbance.

as:

$$\begin{aligned} x(t+1) &= (A+B\Phi)x(t) + B\Gamma v(t) + B_w d(t) \\ y(t) &= Cx(t) + D_w d(t) \in \mathbb{Y} \end{aligned} \tag{3.25}$$

where Φ and Γ are computed based on (2.22), and v is the input obtained from the SRGs (see Figure 3.1). The i-th decoupled subsystem can then be written as:

$$\begin{aligned} x(t+1) &= \bar{A}x(t) + B_i v_i(t) + B_w d(t) \\ y_i(t) &= C_i x(t) + D_{w_i} d(t) \in \mathbb{Y}_i \end{aligned} \tag{3.26}$$

where $\bar{A} = A + B\Phi$, B_i is the i^{th} column of $B\Gamma$, C_i is the i^{th} row of C, and D_{w_i} is the i^{th} row of D_w. Based on (3.26) we create the corresponding robust MAS for the i-th subsystem. The DRG-ss implementation is otherwise unchanged.

Next, a simple example will be presented to show the performance of robust DRG-ss. We consider again system (3.14) with the output constraints: $y_1 \leq 2.1, y_2 \leq 1.1$.

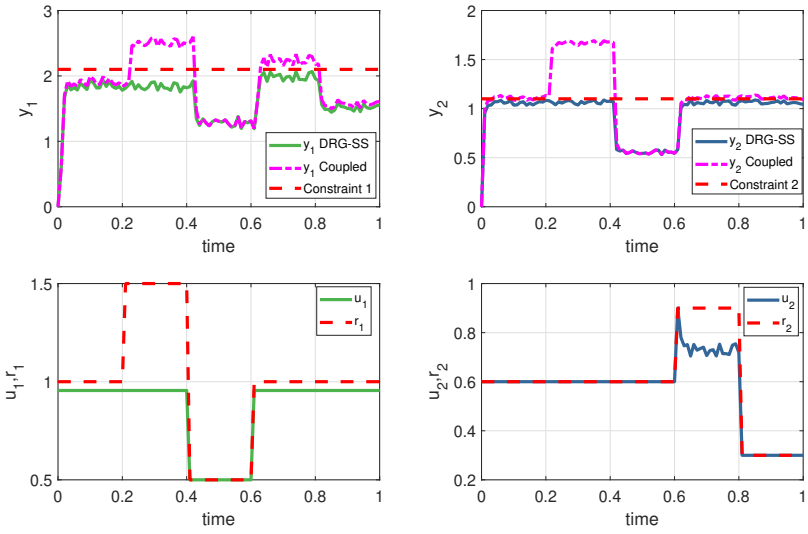

Figure 3.5: DRG-ss with disturbance.

Assume D_w is zero, $B_w = [1.3, 0.3, 2.51]^\top$, and that the disturbance also satisfies $d(t) \in \mathbb{D} := [-0.1, 0.1]$. We decouple the system using the pole assignment method, i.e., placing the closed-loop poles at 0.1.

The results of robust DRG-ss with disturbance are shown in Figure 3.5. In the top subplots of these figures, "y_1 couple" and "y_2 Coupled" refer to the response of the system without DRG-ss (i.e., r applied to G directly), which shows that, with DRG-ss, the constraints are satisfied, as required. As interesting behavior can be realized by comparing the performance of DRG-tf (3.4) and DRG-ss (3.5) It can be seen, from the comparison, the disturbance does not affect u for DRG-tf but it affects u for DRG-ss (see u in Figure 3.5). The reason for this behavior can be explained as follows: it can be seen from Figure 3.1 that the outer feedback in DRG-ss may transmit the effects of disturbances and sensor noise to r'. As a result, the effect of the disturbance on the output may be higher in DRG-ss than in DRG-tf. This may

be a decisive argument to select between DRG-ss and DRG-tf.

Remark 2. *For a system in which the states are not measured, a standard observer may not provide accurate estimation of the state if unknown disturbances is considered in the design. In such a case, the observer developed in [163] can be implemented, where the observer has the ability to take the error introduced by unknown disturbances into consideration.*

3.3.2 DRG with Parametric Uncertainty

In this section, we sketch the DRG scheme to the system $G(z)$ in Figures 2.4 and 3.1 with parametric uncertainty, that is, matrices A and B are uncertain or vary in time. For simplicity, in this discussion, it is assumed that the matrix C is known and $D = 0$. The approach we develop is similar to [164]. Note that we consider parametric uncertainties in the state-space representation since RG approach is a time-domain approach. Therefore, frequency domain uncertainties will not be investigated. Furthermore, we assume that the uncertain/time-varying closed-loop system is asymptotically stable. Thus, stability is still not a concern in DRG-tf, but additional analysis must be carried out to ensure stability of DRG-ss since additional feedback loop is involved in the DRG-ss designing.

For this discussion, reconsider system $G(z)$ but with parametric uncertainty on the A and B matrices, which leads to the square linear system given by:

$$\begin{aligned} x(t+1) &= A(t)x(t) + B(t)u(t) \\ y(t) &= Cx(t) \in \mathbb{Y} \end{aligned} \tag{3.27}$$

In [164], in order to compute the robust MAS for this type of systems, it is assumed that the pair $(A(t), B(t))$ belongs to a given uncertainty polytope defined by the convex hull of the matrices $(A^{(j)}, B^{(j)})$, that is

$$(A(t), B(t)) \in \text{conv}\{(A^{(1)}, B^{(1)}), \ldots, (A^{(N)}, B^{(N)})\},$$

where N is the number of vertices in the uncertainty polytope ([165]). Applying the proposed method in [164] directly to DRG, however, may not guarantee constraint satisfaction because the parametric uncertainties will prevent us from perfectly decoupling the system. To explain, suppose a nominal pair of A and B matrices is selected from a convex hall, and the Φ and Γ presented in (2.21) or (2.22) are used to decouple this nominal system. Since the matrices of the actual system will be different from the nominal ones, this decoupling process results in:

$$x(t+1) = \bar{A}(t)x(t) + \bar{B}(t)v(t), \quad y(t) = Cx(t) \tag{3.28}$$

where the pair $(\bar{A}(t), \bar{B}(t))$ satisfies:

$$(\bar{A}(t), \bar{B}(t)) \in \text{conv}\{(\bar{A}^{(1)}, \bar{B}^{(1)}), \ldots, (\bar{A}^{(N)}, \bar{B}^{(N)})\}, \tag{3.29}$$

where $\bar{A}^{(j)} = A^{(j)} + B^{(j)}\Phi$, $\bar{B}^{(j)} = B^{(j)}\Gamma$. Obviously, a same pair of Γ and Φ is incapable to decouple all the matrices in the uncertainty polytope. This implies that DRG implemented on (3.28) can not achieve perfect decoupling and thus may not enforce the constraints.

To address the above issue, a novel margin in each $O_\infty^{W_{ii}}$ to robustify every channel

against these coupling dynamics is introduced. To elaborate this idea, consider the dynamics of the i-th output of (3.28):

$$\begin{aligned} x(t+1) &= \bar{A}(t)x(t) + \bar{B}_i(t)v_i(t) + B_w(t)\bar{v}(t) \\ y_i(t) &= C_i x(t) \end{aligned} \tag{3.30}$$

where C_i is the i-th row of C, $\bar{B}_i(t)$ corresponds to the i^{th} column of $\bar{B}(t)$, $B_w(t)$ represents all columns of $\bar{B}(t)$ except the i^{th} one, and $\bar{v}(t)$ represents the vector containing all inputs except the i-th one, i.e., vector of all v_k's, $k \neq i$. Our solution described below treats $\bar{v}$ as an unknown bounded disturbance. To accomplish this, we quantify a lower and an upper bound on $\bar{v}$ and robustify $O_\infty^{W_{ii}}$ base on these bounds using results similar to Section 2.1.1. Specifically, to find the bounds, we leverage the fact that each element of $\bar{v}(t)$, $\bar{v}_k$, is the output of an SRG (i.e., monotonically increasing or decreasing), whose goal is to enforce the constraints on the k-th output (i.e., $y_k(t) \in \mathbb{Y}_k$). Thus, we can define upper and lower bounds on each element of $\bar{v}$ using the steady-state constraints (3.15):

$$\begin{aligned} \bar{v}_k^{\max} &= \max\{\bar{v}_k : W_{kk_0}^{(j)}\bar{v}_k \in (1-\epsilon)\mathbb{Y}_k, j = 1, \ldots, N\} \\ \bar{v}_k^{\min} &= \min\{\bar{v}_k : W_{kk_0}^{(j)}\bar{v}_k \in (1-\epsilon)\mathbb{Y}_k, j = 1, \ldots, N\} \end{aligned} \tag{3.31}$$

where $W_{kk_0}^{(j)}$ represents the DC gain of the system from the k-th input to the k-th output given the pair $(\bar{A}^{(j)}, \bar{B}^{(j)})$. Since each $\bar{v}_k(t) \in [\bar{v}_k^{\min}, \bar{v}_k^{\max}]$, we can now treat $\bar{v}(t)$ in (3.30) as an unknown bounded disturbance to create a robust MAS set for the i-th channel, which can be achieved using the ideas from Section 2.1.1 (for unknown disturbances) and references [164, 165] (for polytopic uncertainties). Implementation

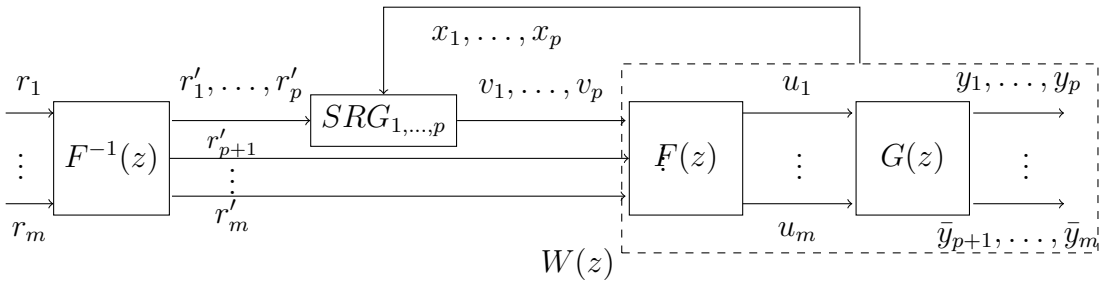

Figure 3.6: DRG-tf block diagram for non-sqaure systems with larger number of inputs. $\bar{y}_{p+1}, \ldots, \bar{y}_m$ represent the outputs that are manually added to system $G(z)$ to transfer it into a square system.

of DRG using the above structure of MAS's will ensure that the system is robust to the plant/model mismatch and, thus, the constraints will be satisfied. It is important to notice that this approach may lead to conservative results depending on how much the MAS is shrunk (i.e., how large the model mismatch is). However, if the system is "almost" decoupled (i.e., the nominal system is close to the actual one), then the shrinkage will be negligible. For the sake of brevity, numerical examples and further analysis on this topic will appear in our future work.

3.4 Extension of DRG to non-square MIMO systems

In this section, the extension of DRG to non-square MIMO systems, i.e., systems where the number of inputs is either larger or smaller than the number of outputs, will be introduced. For better explanation, we will treat these cases separately in the following discussion. Generally speaking, we achieve this by either introducing fictitious outputs to transform the system into a square one (see Figure 3.6), or only

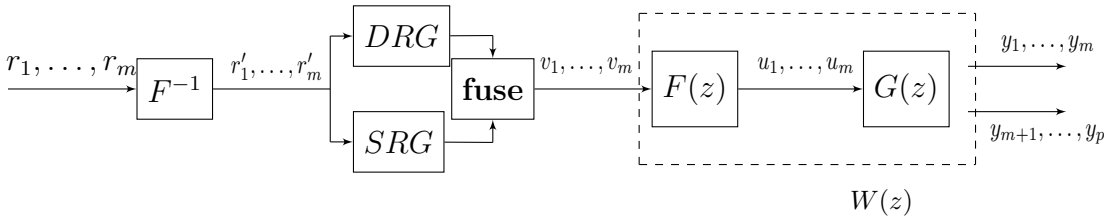

Figure 3.7: DRG-tf block diagram for non-sqaure systems with larger number of inputs.

decoupling a square subsystem of it (see Figure 3.7). For the sake of clarity, we will only focus on the extension of DRG-tf with the diagonal method; the same process can be applied to DRG-tf with identity method and DRG-ss scheme.

3.4.1 Systems with larger number of inputs

Recall that the main idea behinds the extension of DRG to non-square systems with m inputs and p outputs ($m > p$) is to transfer the system into a square one. To elaborate this idea, we manually introduce $m - p$ outputs, $\bar{Y}_{p+1}, \ldots, \bar{Y}_m$, leading to the square system $\widetilde{G}$, described below:

$$\begin{bmatrix} Y_1(z) \\ \vdots \\ Y_p(z) \\ \bar{Y}_{p+1}(z) \\ \vdots \\ \bar{Y}_m(z) \end{bmatrix} = \underbrace{\begin{bmatrix} G_{c_{1,p}} & G_{c_{p+1,m}} \\ 0_{m-p,p} & \bar{G} \end{bmatrix}}_{\widetilde{G}} \begin{bmatrix} U_1(z) \\ \vdots \\ U_p(z) \\ \vdots \\ U_m(z) \end{bmatrix} \tag{3.32}$$

where $\bar{G}$ is an $(m-p) \times (m-p)$ transfer matrix representing the fictitious outputs, and $G_{c_{1,p}}$ and $G_{c_{p+1,m}}$ denote the first p columns of G and the last $(m-p)$ columns of G, respectively.

The choice of the fake dynamics (i.e., $[0_{m-p,p} \quad \bar{G}]$) in (3.32) is not unique. The

reason for choosing $[0_{m-p,p} \quad \bar{G}]$ is that $\widetilde{G}^{-1}$ and F can be easily obtained through block matrix inversion ([166]), and the structure of F is easy to study, as will be explained below. For the diagonal method in DRG-tf, the decoupled system W (see Figure 3.6) is constructed as:

$$W = \left[\begin{array}{ccc:c} G_{11}(z) & \dots & 0 & \\ \vdots & \ddots & \vdots & 0_{p,(m-p)} \\ \underbrace{0(z) \quad \dots \quad G_{pp}(z)}_{W_p} & & & \\ \hdashline & & & \\ 0_{(m-p),p} & & & \bar{G}_w \end{array}\right] \tag{3.33}$$

where $\bar{G}_w$ is a $(m-p) \times (m-p)$ transfer function matrix that is chosen such that it has a stable inverse, so that F^{-1} can be computed (see (2.17)). Note that the choice of $\bar{G}_w$ will not affect the system performance since $\bar{G}_w$ corresponds to the fictitious outputs.

Recall that the actual outputs of the system are $Y_1, \dots, Y_p$ and the constraints are on these outputs. So, as Figure 3.6 shows, only p different SRGs are needed to ensure these outputs satisfy the constraints and there is no need to design SRGs for $\bar{G}_w$. Finally, F^{-1} is introduced to ensure that u is close to r, as before. By choosing $\widetilde{G}$ and W as shown in (3.32) and (3.33), F can be written as:

$$F = \begin{bmatrix} G_{c_{1,p}}^{-1} W_p & -G_{c_{1,p}}^{-1} G_{c_{p+1,m}} \\ 0_{m-p,p} & \bar{G}^{-1} \bar{G}_w \end{bmatrix} \tag{3.34}$$

An interesting case is that the $\bar{G}$ is chosen to be equal to the $\bar{G}_w$, then, $\bar{G}^{-1}\bar{G}_w$ in (3.34) will become an identity matrix, which implies that F is unrelated to the choice of $\bar{G}$. Of course, for this to hold, $\bar{G}$ needs to be invertible to ensure that (3.34) exists.

Remark 3. *As can be seen from* (3.34)*, if* $\bar{G} \neq \bar{G}_w$*, then* F *is related to the designing of both* $\bar{G}$ *and* $\bar{G}_w$*. This implies that a proper set of* $\bar{G}$ *and* $\bar{G}_w$ *can be selected such that the norm of* F *is small, which as discussed in Section 3.1.2, will lead to a small distance between* u *and* r *(see Figure 3.6) and, thus, good tracking performance.*

Since $G(z)$ has been transformed into a square system, the same analysis presented in Section 3.1.2 can be applied to study the steady-state and transient performance of DRG-tf for non-square systems. Hence, further analysis will not be provided here.

3.4.2 Systems with larger number of outputs

Assume system $G(z)$ in Figure 3.7 has m inputs and p outputs, with $p > m$. Instead of decoupling the entire $G(z)$, only a square subsystem of G is decoupled.

Without loss of generality, it is assumed that the square subsystem corresponds to the first m outputs of G, but the method proposed below can be applied to other square subsystems as well. Let us denote the $m \times m$ square subsystem of G as G_m. Same as DRG-tf for square systems (see Section 2.3), F is designed to decouple G_m, resulting in the diagonal subsystem, W_m, shown below:

$$W_m = \begin{bmatrix} G_{m_{11}}(z) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0(z) & \dots & G_{m_{mm}}(z) \end{bmatrix} \tag{3.35}$$

Then, the whole system W (i.e., GF) can be described by:

$$W = \left[\begin{array}{c} W_m \\ \hdashline \underbrace{FG_{m+1,p}}_{W_p} \end{array}\right] \tag{3.36}$$

where $G_{m+1,p}$ represents the last $(p-m)$ rows of G.

As can be seen, from Figure 3.7, one DRG (which contains m decoupled SRGs) for W_m is designed to ensure that the outputs $y_1, \ldots, y_m$ satisfy the constraints. Afterwards, a single SRG for W_p is implemented to ensure that the outputs $y_{m+1}, \ldots, y_p$ satisfy the constraints. The challenge of this approach is that two sets of v's are computed: one by the DRG and one by the SRG (as shown in Figure 3.7). Thus, one question is raised: how can the two sets of v's be "fused" together while all outputs are satisfied the constraints. There are several ways to accomplish this task. The most straightforward solution is to select the smallest κ among the $m+1$ different κ's, denoted as $\bar{\kappa}$, that is:

$$\bar{\kappa} = \mathbf{min}(\kappa_1, \ldots, \kappa_{m+1}) \tag{3.37}$$

and the update law for v becomes:

$$v(t+1) = v(t) + \bar{\kappa}(r'(t) - v(t)).$$

With the above $\bar{\kappa}$, the convexity of the MAS guarantees that the constraints for all outputs are satisfied and the solutions from the DRG and SRG are unified. However, the response of this approach may be conservative since the smallest κ is utilized. An alternative way to fuse the v's is explained follows. First, denote the set of v's given

by the SRG as v_s and the set of v's given by the DRG as v_d. We solve an RG-like LP (see (2.14)) to find the point in $O_\infty^{W_m}$ that is closest to v_s (recall that $O_\infty^{W_m}$ refers to the MAS for W_m), denoted as v_{t_1}. Similarly, we solve another LP to find the closest point to v_d in $O_\infty^{W_p}$, where $O_\infty^{W_p}$ represents the MAS for W_p, denoted as v_{t_2}. Note that v_{t_1} and v_{t_2} are both constraint-admissible for all outputs since they are in $O_\infty^{W_p}$ and $O_\infty^{W_m}$ at the same time.

Finally, the actual set of v's that is applied to $F(z)$ is characterized by:

$$v = \begin{cases} v_{t_1} & \text{if } \|r' - v_{t_1}\| \leq \|r' - v_{t_2}\| \\ v_{t_2} & \text{otherwise} \end{cases}$$

By choosing v as above, it is guaranteed that the constraints for all outputs are satisfied. Also, the second approach has a less conservative response than that given by the first approach but at a cost of higher computational effort since two more LPs are required to be solved. Finally, F^{-1} is introduced to ensure that u is close to r, as before.

Chapter 4

Preview Reference Governor

Recall from Section 1.2.2, Preview Reference Governor (PRG) intends to solve pointwise-in-time state and control constraints while taking into account the preview information of the reference and/or disturbances signals. More specifically, as shown in the black diagram of PRG (Figure 4.1), it is assumed that the preview information of the reference signal is available, that is $r(t), r(t+1), \ldots, r(t+N)$, are known to the controller at time t, where N refers to the "preview horizon". Similar to SRG, the goal of PRG is to select $v(t)$ as close as possible to $r(t)$ such that the output constraints are satisfied for all times. Meanwhile, the PRG scheme must take into consideration the preview information of $r(t)$ in order to improve the transient performance as com-

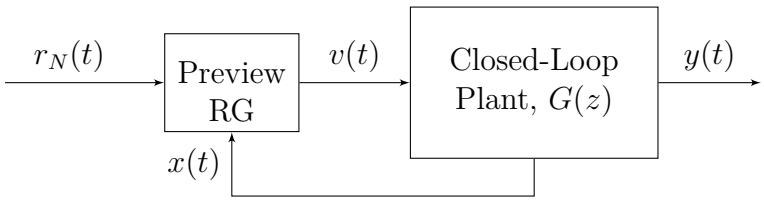

Figure 4.1: Preview Reference Governor block diagram. $r_N(t)$ represents the lifted reference over the preview horizon, i.e., $r_N(t) = (r(t), \ldots, r(t+N))$.

pared to SRG. To achieve these goals, first, $r(t)$ and $v(t)$ are lifted from $\mathbb{R}$ to $\mathbb{R}^{(N+1)}$ in order to describe them over the entire preview horizon. Next, the system G is represented based on the lifted input and a new construction of Maximal Admissible set (MAS) is calculated based on this new representation. Finally, PRG is formulated as an extension of SRG, where a new optimization problem is solved based on the new MAS and a new update law is used to compute $v(t)$. In this chapter, first, the PRG for single-input system will be introduced. With the simulation of PRG on the onr-link arm robot example, one drawback of PRG is demonstrated that is PRG may calculate conservative inputs under certain conditions, especially for long preview horizons. Then, an extension of PRG, namely Multi-N PRG, is proposed to remedy this issue. The modification of PRG to systems with inaccurate preview information, and systems with disturbance and noise is also presented. Finally, two solutions to handle multi-inputs systems is proposed to extend the applicability of PRG.

4.1 Preview Reference Governors for Single Input Systems

In this section, the PRG for single-input systems is introduced and analyzed. Additionally, PRG is compared with SRG using the one-link arm robot example to demonstrate that it can significantly improve the transient performance while enforcing the constraints.

Consider the system $G(z)$ shown in Figure 4.1. Assume $r(t)$, $r(t+1)$, ..., $r(t+N)$ are available at time t, where $r(t) \in \mathbb{R}$ is the current value of the setpoint and

$r(t+1), \ldots, r(t+N) \in \mathbb{R}$ are the preview information ($N \geq 0$ represents the preview horizon). We now define the lifted signals $r_N(t) \in \mathbb{R}^{(N+1)}$ (shown in Figure 4.1) and $v_N(t) \in \mathbb{R}^{(N+1)}$, as follows:

$$r_N(t) = (r(t), \ldots, r(t+N)), \quad v_N(t) = (v(t), \ldots, v(t+N)) \tag{4.1}$$

Using the lifted signals, $G(z)$ can be equivalently expressed as:

$$\begin{aligned} x(t+1) &= Ax(t) + \underbrace{\begin{bmatrix} B & 0 & \ldots & 0 \end{bmatrix}}_{\widetilde{B}} v_N(t), \\ y(t) &= Cx(t) + \underbrace{\begin{bmatrix} D & 0 & \ldots & 0 \end{bmatrix}}_{\widetilde{D}} v_N(t) \end{aligned} \tag{4.2}$$

Next, a detailed explanation on constructing the Maximal Admissible Set (MAS) for the lifted system will be provided. Recall from (2.3) that in order to characterize MAS in the SRG framework, it is assumed that $v(t)$ is held constant for all future time. This assumption ensures that the optimization problem (2.14) will always have a feasible solution, namely $\kappa = 0$. In order to extend these ideas to PRG, we assume that $v(t)$ may vary within the preview horizon to extend the flexibility in choosing the v, but is held constant beyond the preview horizon. Therefore, the dynamics of $v_N(t)$ are selected to be:

$$v_N(t+1) = \bar{A} v_N(t) \tag{4.3}$$

with initial condition $v_N(0) = v_{N_0}$, where $\bar{A}$ is defined by:

$$\bar{A} = \left[\begin{array}{cccc:c} 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \\ \hdashline \underbrace{0 \quad 0 \quad \dots \quad 0}_{N} & & & & 1 \end{array}\right] \tag{4.4}$$

This choice of $\bar{A}$, together with the definition of $v_N(t)$ in (4.1), enforce that for all $t \geq N$, $v(t) = v(N)$. With these dynamics, the new MAS is defined as:

$$\begin{aligned} O_\infty^N :=& \{(x_0, v_{N_0}) \in \mathbb{R}^{n+(N+1)} : x(0) = x_0, v_N(0) = v_{N_0} \\ & v_N(t+1) = \bar{A} v_N(t), y(t) \in \mathbb{Y}, \forall t \in \mathbb{Z}_+\} \end{aligned} \tag{4.5}$$

Similar to (2.6), a finitely determined inner approximation of the new MAS can be computed by tightening the steady-state constraint (to prove finite determinism, note that after N time steps, $v(t)$ converges to a constant, which reduces the problem to that in the standard MAS theory). In the rest of this dissertation, with an abuse of notation, the O_∞^N is used to denote the finitely determined inner approximation of the MAS.

With the new MAS (O_∞^N) defined, we are now ready to present the PRG formulation. Recall from SRG theory, it computes $v(t)$ using the update law in (2.13), where κ is obtained by solving the LP shown in (2.14). Note from (2.13) that $v(t) \in \mathbb{R}$ can be viewed as the internal state of the SRG. To extend these ideas to PRG, $(N+1)$ new states is introduced in PRG formulation, where the state update law is shown as

follows:

$$v_N(t) = \bar{A}v_N(t-1) + \kappa(r_N(t) - \bar{A}v_N(t-1)) \tag{4.6}$$

where κ is the solution of the following linear program:

$$\begin{aligned} \underset{\kappa\in[0,1]}{\text{maximize}} \quad & \kappa \\ \text{s.t.} \quad & v_N(t) = \bar{A}v_N(t-1) + \kappa(r_N(t) - \bar{A}v_N(t-1)) \\ & (x(t), v_N(t)) \in O_\infty^N \end{aligned} \tag{4.7}$$

An explicit algorithm, similar to the one in SRG [149], can be developed to solve this LP efficiently. At every time step, the PRG solves the above LP to compute κ, updates $v_N(t)$ using (2.14), and applies the first element of $v_N(t)$ to the system G.

Remark 4. *When $N = 0$, PRG reduces to SRG because $\bar{A}$ turns into the scalar with value* 1*. Therefore, PRG is a proper extension of SRG.*

PRG inherits the properties of SRG, including bounded-input bounded output stability, convergence, and recursive feasibility, as shown in the following proposition.

Proposition 1. *The PRG formulation is recursively feasible, bounded-input and bounded-output stable (BIBO), and for a constant r, $v(t)$ converges.*

Proof. To show recursively feasibility, consider the update law (4.6). As can be seen, $\kappa = 0$ implies that $v_N(t) = \bar{A}v_N(t-1)$, which matches the dynamic of v_N that is assumed in O_∞^N. Positive invariance of O_∞^N implies that if the pair $(x(t-1), v_N(t-1))$ is admissible, then $(x(t), v_N(t-1))$ is also admissible. As a result, $\kappa = 0$ is always a feasible solution to the LP in (4.7), implying recursively feasibility of the PRG. As for BIBO stability, recall that $v_N(t)$ is the convex combination between $\bar{A}v_N(t-1)$ and

the current reference $r_N(t)$. Thus, if $r(t)$ is bounded, then so is $v_N(t)$. This, together with the asymptotic stability of G, implies BIBO stability of the system. To prove the convergence property, assume $r(t) = r, \forall t \in \mathbb{Z}_+$. It can be shown that, from (4.6), every element in $v_N(t)$ is monotonic $\forall t \geq N$ and bounded by r. Thus, $v_N(t)$ must converge to a limit. □

Remark 5. *For the case where the state of $G(z)$ (i.e., $x(t)$ shown in Figure 4.1) is not measured, standard Luenberger observers can be designed to estimate the state.*

Illustration Example

Next, the illustration of the PRG using the one-link arm robot, shown in Figure 2.5, is presented. A state-space model of the arm is given by:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -14.7 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} \tau, \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.8}$$

where $x_1 \triangleq \theta$, $x_2 \triangleq \dot{\theta}$ are the states and τ (i.e., external torque) is the control input. For this example, we assume that both states are measured.

To design a controller and consequently implement the PRG, the system model (4.8) is first discretized at $T_s = 0.01s$. Then, a state-feedback controller is applied to ensure that the output, θ, tracks the desired angle, v, perfectly (see Section 2.6.1), that is: $\tau = 66.67v - 61.77x_1 - 9.64x_2$. This results in the closed-loop system G shown in Figure 4.1. In this discussion, a trajectory-following maneuver is simulated, wherein the PRG is implemented to ensure that the output, θ, remains within $[-45°, 45°]$. We assume that the preview information is available from the given pre-set trajectory and

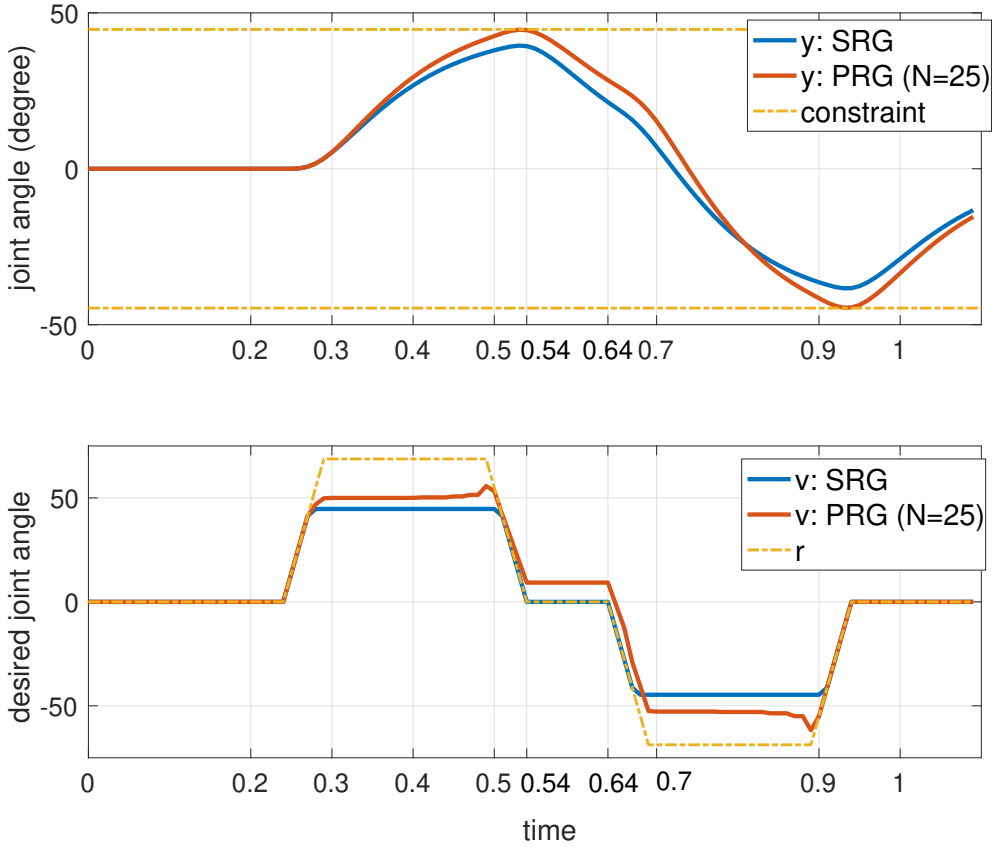

Figure 4.2: Comparison of PRG and SRG. The blue lines represent the results of SRG and the red lines refer to the results of PRG. The top plot shows the outputs and the bottom plot shows the control inputs and the setpoint.

the preview horizon, N, is chosen to be 25 (i.e., 0.25 seconds).

The comparison between the performance of PRG and SRG is shown in Figure 4.2. As can be seen, $v(t)$ given by PRG is closer to $r(t)$ when $t \in [0.3, 0.5]$, implying that PRG is less conservative than SRG. This is because when $t = 0.3$, the PRG has the future information that the reference would drop down to 0 in future 25 time steps so that it allows a larger $v(t)$ than SRG. For the same reason, $v(t)$ given by PRG is less conservative when $t \in [0.7, 0.9]$.

Meanwhile, Figure 4.2 shows a limitation of PRG. Specifically, it can be seen that when $t \in [0.54, 0.64]$, $v(t)$ given by PRG can not reach $r(t) = 0$, even though 0 is an admissible input. To explain the root cause of this behavior, note that when $t = 0.54$, the preview information available to the PRG is that $r(t)$ drops down to $-69°$ at $t = 0.7$ and stays constant, and the increase of $r(t)$ back to 0 at $t = 0.9$ is beyond the preview horizon. To enforce the lower constraint for $t > 0.7$, the PRG calculates a κ smaller than 1. However, since κ affects all elements of $v_N(t)$ (see (4.6)), this leads to $v(t) \neq r(t)$ when $t \in [0.54, 0.64]$ and, thus, a more conservative solution is obtained.

It is important to realize that the above limitation of PRG can be addressed by decreasing the preview horizon N. However, if N is too short, the tracking performance of the system might not be improved significantly as compared with SRG. To ensure a superior system performance while addressing the above limitation of PRG, an extension of PRG is provided in the following section.

4.1.1 Multi-Horizon PRG (Multi-N PRG)

To overcome the limitation of PRG, this section introduces a modification of PRG, namely Multi-Horizon Preview Reference Governor (Multi-N PRG). As the name suggests, instead of having a single preview horizon, multiple preview horizons are considered. More specifically, for each horizon, a MAS is characterized, and multiple PRGs are solved at each time step, one for each MAS. A technical challenge for this approach is that $v_N(t)$'s for different preview horizons will have different dimensions and different interpretations. In addition, a practical challenge of Multi-N PRG is that storing multiple MASs will lead to a significant increase in the memory require-

ments. Our novel solution overcomes these challenges by unifying the v_N's so that only one MAS is required, and fusing the different PRG solutions in a special way to guarantee constraint satisfaction and recursively feasibility.

Initially, consider the following set of q preview horizons: $N_1 < N_2 < \ldots < N_q$. Let $O_\infty^{N_i}$ be defined identical to (4.5), i.e., the MAS of the lifted system with preview horizon N_i. We first reveal that there exists a simple relationship between $O_\infty^{N_i}$, $i = 1, \ldots, q-1$, and $O_\infty^{N_q}$. Recall from (4.5) and (4.1) that in order to construct $O_\infty^{N_i}$, it is assumed that $v(t)$ is held constant beyond the preview horizon (i.e., after $t = N_i$). Similarly, to construct $O_\infty^{N_q}$, it is assumed that $v(t)$ is held constant after $t = N_q$. Thus, $O_\infty^{N_i}$ and $O_\infty^{N_q}$ have the following relationship:

$$(x, v_{N_i}) \in O_\infty^{N_i} \Leftrightarrow (x, [v_{N_i}^T, \underbrace{v_{N_i}(N_i+1)^T, \ldots, v_{N_i}(N_i+1)^T}_{N_q - N_i \text{ terms}}]^T) \in O_\infty^{N_q}$$

where $\Leftrightarrow$ denotes the bidirectional implication and $v_{N_i}(N_i+1)$ refers to the last element in v_{N_i}. Therefore, given this relationship, only $O_\infty^{N_q}$ is needed to be computed and stored.

Next, the Multi-N PRG formulation is described, which solves q PRGs at each time step, one for each N_i. As justified in the previous section, all PRGs use the same MAS, namely $O_\infty^{N_q}$. More specifically, recall from Section 4.1 that PRG contains an internal state with an update law given by (4.6). Similarly, an internal state, denoted here by $\widetilde{v} \in \mathbb{R}^{(N_q+1)}$, is also introduced in the Multi-N PRG scheme. The i-th PRG in the Multi-N PRG framework assumes the update law:

$$\widetilde{v}(t) = M_i(\bar{A}\widetilde{v}(t-1) + \kappa_i(r_{N_q}(t) - \bar{A}\widetilde{v}(t-1)) \tag{4.9}$$

where $\bar{A}$ is defined the same as (4.4) but with $N = N_q$, and $r_{N_q}(t)$ is the lifted version of r at time t (defined as in (4.1)). The matrix M_i is introduced to enforce the control inputs beyond the preview horizon N_i to be constant (this is to maintain consistency with the fact that the i-th PRG has a preview horizon of length N_i). This is achieved by constructing M_i as:

$$M_i = \begin{bmatrix} I_{(N_i+1)} & 0_{(N_i+1)\times(N_q-N_i)} \\ \widetilde{I}_{(N_q-N_i)\times(N_i+1)} & 0_{(N_q-N_i)\times(N_q-N_i)} \end{bmatrix}$$

where $\widetilde{I}_{(N_q-N_i)\times(N_i+1)}$ is a matrix with zeros everywhere except the rightmost columns, which are given by $[1, 1, \ldots, 1]^T$. In the update law (4.9), for the i-th PRG , the scalar κ_i is computed by solving the following LP:

$$\begin{aligned} \underset{\kappa_i \in [0,1]}{\text{maximize}} \quad & \kappa_{\text{i}} \\ \text{s.t.} \quad & \widetilde{v}(t) = M_i(\bar{A}\widetilde{v}(t-1) + \kappa_i(r_{N_q}(t) - \bar{A}\widetilde{v}(t-1)) \\ & (x(t), \widetilde{v}(t)) \in O_\infty^{N_q} \end{aligned} \tag{4.10}$$

Finally, in order to obtain the most superior performance among different preview horizons, the solutions calculated in different PRGs are fused by taking the maximum value among $\{\kappa_i\}, i = 1, \ldots, q$, denoted by κ_{i^*}. The index that corresponding to κ_{i^*} is denoted by i^*. Then, the final update law of Multi-N PRG is shown as follows:

$$\widetilde{v}(t) = M_{i^*}(\bar{A}\widetilde{v}(t-1) + \kappa_{i^*}(r_{N_q}(t) - \bar{A}\widetilde{v}(t-1))$$

This formulation maintains recursively feasibility. Indeed, suppose PRG i^* is the PRG that computes κ_{i^*} at time step t. At next time step $t+1$, the same PRG can

calculate a feasible solution due to the recursively feasibility of the PRG formulation.

Note that not all PRGs in the Multi-N PRG formulation is guaranteed to have feasible solutions at every time step. Moreover, if a feasible solution to these PRGs does not exist, κ_i's for these PRGs are set to be 0.

Illustration Example

In this section, the performance of Multi-N PRG on the one-link arm robot is demonstrated. The dynamics of the one-link arm robot with controller can be found in Section 4.1.

The simulation results of Multi-N PRG on the one-link arm robot example are shown in Figure 4.12. For comparison, the simulation results of PRG with $N = 25$ is also provided. For the sake of illustration, we consider the extreme scenario where the Multi-N PRG uses all preview horizons between 0 and 25; i.e., $q = 26$ and $N_1 = 0$, $N_2 = 1, \ldots, N_{26} = 25$.

It can be seen, from Figure 4.12, the outputs for both Multi-N PRG and PRG satisfy the constraints, as expected. However, from the bottom plot of Figure 4.12, when $t \in [0.54, 0.64]$, $v(t)$ given by Multi-N PRG reaches $r(t)$ while $v(t)$ computed by PRG is above $r(t)$. The reason for this behavior can be explained as follows. When $t \in [0.54, 0.64]$, the PRG corresponding to $N_1 = 0$ computes $\kappa = 1$, which leads to $v(t) = r(t)$. Note that the actual performance improvement of Multi-N PRG (as seen on the output plot) is not large for this specific example but it could be large in general.

Next, the impact of the choice of q in the Multi-N PRG on the system performance will be discussed. For the sake of illustration, consider two extreme cases: first, N_1

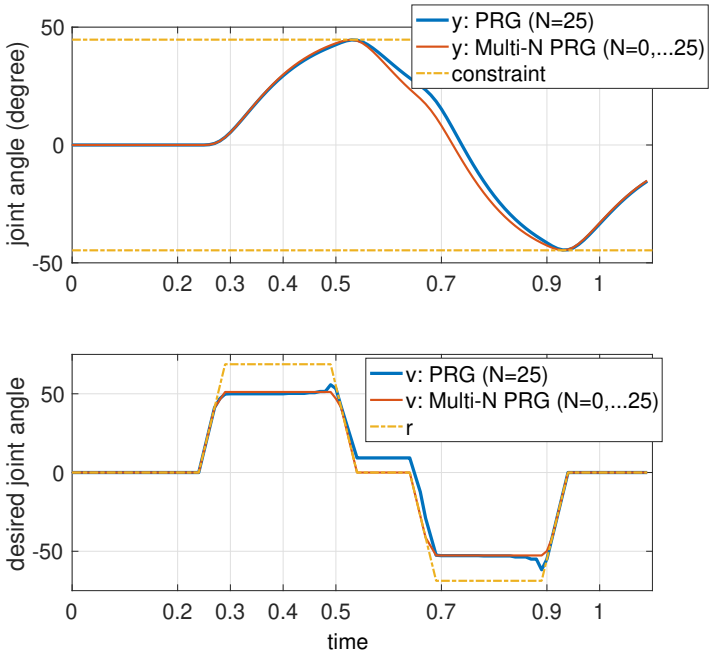

Figure 4.3: Comparison of Multi-N PRG and PRG with $N = 25$.

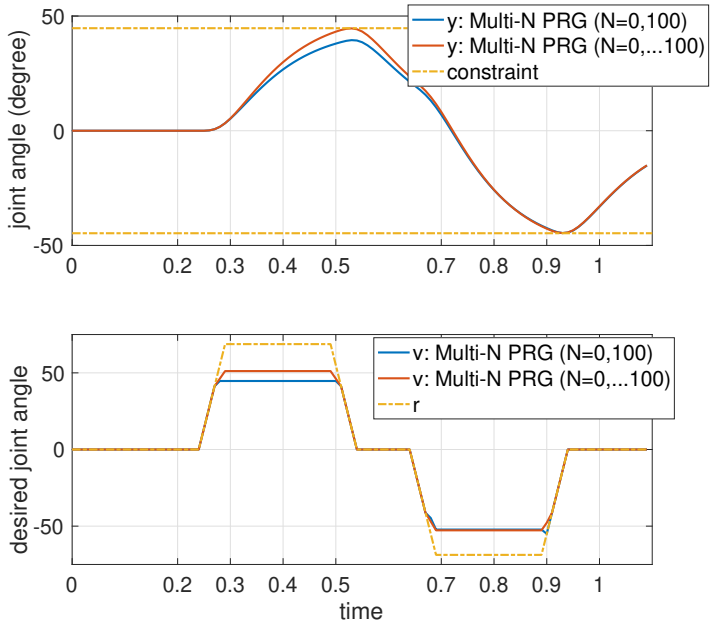

Figure 4.4: Comparison of Multi-N PRG with $q = 2$ and $q = 101$.

and N_2 are chosen to be 0 and 100 (i.e., $q = 2$); second, $N_1, \ldots, N_{100}$ are chosen to be all numbers from 0 to 100 (i.e., $q = 101$). The reason $N_q = 100$ is selected is that the preview horizon in this case is 1 second (recall the sample time is $T_s = 0.01$), which is longer than the variation of the reference in our specific example, and can clearly show the difference on the system performance between the two cases. The comparison between the two cases implemented on the one-link arm robot example is shown in Figure 4.4, which demonstrates performance improvements when a larger q is used. Generally, in the designing aspect, when the preview horizon is longer than the expected variations of the reference, it is better to use the Multi-N PRG formulation with a larger q. However, this will lead to an increase in the computational burden, which is discussed later.

4.2 Robust Preview Reference Governor

In previous discussion, PRG is deigned for a perfectly known system with accurate preview information, which is unlikely to happen in real scenario. In this section, the extensions of PRG to systems with disturbance preview, systems with parametric uncertainties, as well as preview-information uncertainties, are introduced.

4.2.1 Preview Reference Governor with Disturbance Preview

In previous sections, we consider systems in which the preview information of the reference signal is available. However, there are situations wherein preview information of disturbance signals may be available as well. For example, in printing systems, the effect of paper, which modeled as disturbance, on the heating or charging systems are known with some preview, since the timing of the paper leaving the printing tray is precisely controlled [167]. Motivated by this example, in this section, we consider systems where preview information of disturbances is available within a given preview horizon. For simplicity, the preview for the reference signal is not considered in the discussion, though the results can be combined with those of the previous sections to consider preview on both references and disturbances.

Consider a system with additive bounded disturbance given by:

$$\begin{aligned} x(t+1) &= Ax(t) + Bv(t) + B_w w(t) \\ y(t) &= Cx(t) + Dv(t) + D_w w(t) \end{aligned} \tag{4.11}$$

where $w \in \mathbb{W}$ and $\mathbb{W}$ is a compact polytopic set with origin in its interior. Essentially, the disturbance preview is incorporated into the PRG formulation as follows: the MAS is characterized as a function of the current state, input, and the known disturbances within the preview horizon. The set is then shrunk to account for the worst-case realization of the unknown disturbance beyond the preview horizon. Specifically, the

robust MAS for systems with disturbance preview can be defined as:

$$\begin{aligned} O_\infty^w = &\{(x_0, v_0, w_0, \dots, w_N) : x(0) = x_0, v(0) = v_0, w(i) = w_i, \\ &0 \le i \le N, y(t) \in \mathbb{Y}, \forall t \in \mathbb{Z}_+, \forall w(j) \in \mathbb{W}, j > N\} \end{aligned} \tag{4.12}$$

To compute this set, define $\mathbb{Y}_t = \mathbb{Y}$ for $t = 0, \dots, N$, $\mathbb{Y}_{N+1} = \mathbb{Y} \sim D_w\mathbb{W}$, and $\mathbb{Y}_{t+1} = \mathbb{Y}_t \sim CA^{t-N-1}B_w\mathbb{W}$ for $t \ge N+1$, where $\sim$ represents the Pontryagin subtraction. Then, the condition $y(t) \in \mathbb{Y}$ in (4.12) can be characterized equivalently by:

$$CA^t x_0 + \left(\sum_{k=0}^{t-1} CA^k B + D\right) v_0 + h_d(t) \in \mathbb{Y}_t$$

where $h_d(t)$ is defined as:

$$h_d(t) = \begin{cases} \sum_{k=0}^{t-1} CA^{t-1-k}B_w w(k) + D_w w(t) & \text{if } t \le N \\ \sum_{k=0}^{N} CA^{t-1-k}B_w w(k) & \text{if } t > N \end{cases}$$

In summary, by shrinking the MAS to take the worst case disturbance into consideration, robust PRG guarantees constraints satisfaction for all values of disturbance beyond the preview horizon.

Proposition 2. *The robust PRG formulation to disturbance previews is BIBO stable, recursively feasible, and for a constant r, $v(t)$ converges.*

Proof. Similar to the proof for Proposition 1 □

Illustration Example

In this section, the system performance of the robust PRG with preview disturbance (as explained above) on the one-link arm robot example will be presented. The disturbance is assumed to be generated from the joint torque, i.e. $B_w = B$ and $D_w = 0$. We also assume that the disturbance satisfies $w \in \mathbb{W} := [-0.1, 0.1]$. For the purpose of simulations, the disturbance is generated randomly and uniformly from the interval $[-0.1, 0.1]$.

The results of PRG with the disturbance as the preview information are shown in Figure 4.5. Two preview horizons are chosen, namely $N = 20$ and $N = 50$, in order to illustrate the relationship between the system performance and the length of the preview horizon. For a better comparison, Figure 4.5 also shows the response of robust SRG with unknown bounded disturbance (i.e., no preview).

The following observations can be made. First and most importantly, the output from all methods satisfy the constraints, as expected. Second, it can be seen that $v(t)$ given by PRG is closer to $r(t)$ (less conservative) than that of SRG. The reason for this behavior is that the constraint set for PRG with disturbance preview is less conservative than that for SRG with unknown disturbance. Third, the longer the preview horizon, the better the performance (less conservative) obtained.

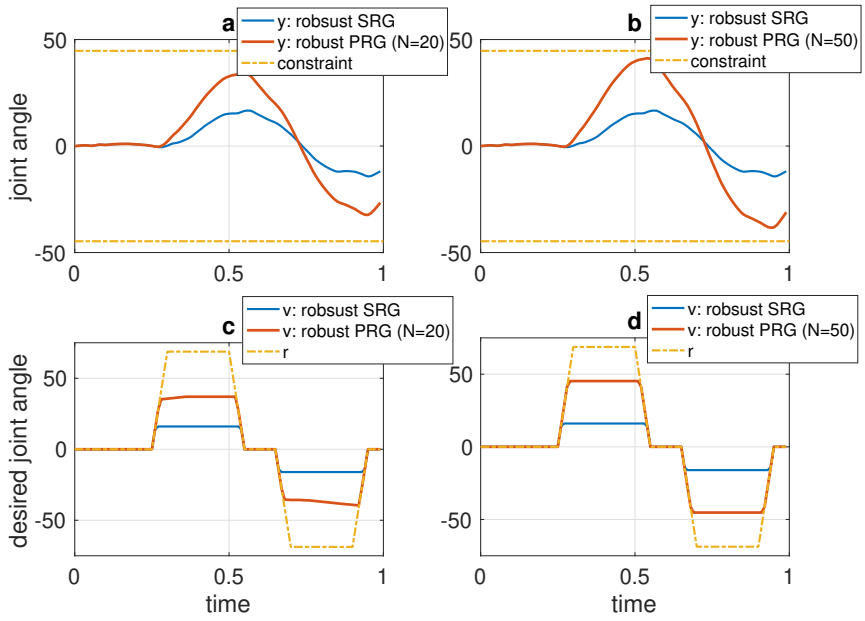

Figure 4.5: Comparison of PRG with disturbance preview and SRG with bounded unknown disturbance.

4.2.2 Robust PRG for systems with Parametric Uncertainties

Reconsider system $G(z)$ in Figure 4.1, but now with modeling uncertainty on the A and B matrices:

$$\begin{aligned} x(t+1) &= A(t)x(t) + B(t)v(t) \\ y(t) &= Cx(t) + Dv(t) \end{aligned} \tag{4.13}$$

where the pair $(A(t), B(t))$ is assumed to belong to an uncertainty polytope defined by the convex hull of the matrices:

$$(A(t), B(t)) \in \text{conv}\{(A^{(1)}, B^{(1)}), \ldots, (A^{(L)}, B^{(L)})\}$$

where L is the number of vertices in the uncertainty polytope. As shown in [165], a robust MAS can be constructed for this uncertain system. We note that a similar idea can be implemented for the PRG. Similar to (4.2), we write the dynamics in terms of the lifted input, but now we consider the uncertainties:

$$\begin{aligned} x(t+1) &= A(t)x(t) + \underbrace{\begin{bmatrix} B(t) & 0 & \dots & 0 \end{bmatrix}}_{\widetilde{B}(t)} v_N(t), \\ y(t) &= Cx(t) + \widetilde{D}v_N(t) \end{aligned} \tag{4.14}$$

Then, the pair $(A(t), \widetilde{B}(t))$ must belong to a convex hull of the following matrices:

$$(A(t), \widetilde{B}(t)) \in \text{conv}\{(A^{(1)}, \widetilde{B}^{(1)}), \dots (A^{(L)}, \widetilde{B}^{(L)})\}$$

where $\widetilde{B}^{(l)} = \begin{bmatrix} B^{(l)} & 0 & \dots & 0 \end{bmatrix}, \quad l = 1, \dots, L$. By doing this, the method proposed in [165] can be used to compute the robust MAS for PRG with system uncertainties. For the sake of brevity, numerical examples will not be provided in this section.

4.2.3 Robust PRG for systems with Uncertain Preview Information

In previous designing, we assumed that the preview information is accurate along the preview horizon. However, this assumption might not hold in practice if the preview information comes from inaccurate measurements or uncertain models. In this section, an extension of PRG that can handle inaccurate preview information of references is presented. A brief explanation on the effectiveness of inaccurate preview

information on the system performance will be provided below. As can be seen from (4.7), the inaccurate values in $r_N(t)$ will result in the incorrectly calculation of $v_N(t)$. The implication of this is that, in the next time step, $v_N(t+1)$ will be computed based on the delayed version of this incorrect $v_N(t)$, which, for example, might cause $v(t)$ to change before $r(t)$ does. This behavior is unacceptable for most systems. Note that the constraints will still be satisfied; however, as argued above, performance may suffer. Thus, our solution in this section focuses on avoiding this loss of performance when the preview information is inaccurate.

To begin, we assume that at time t, $r(t)$ is accurately known, but there is uncertainty on the value of r along the preview horizon, i.e., $r(t+1), \ldots, r(t+N)$ are inaccurate. To accommodate this uncertainty, the update law of $v_N(t)$ (4.4) is modified to:

$$\bar{A} = \begin{bmatrix} 1-\lambda_1 & \lambda_1 & 0 & \ldots & 0 \\ 1-\lambda_1 & \lambda_1-\lambda_2 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1-\lambda_1 & \lambda_1-\lambda_2 & \lambda_2-\lambda_3 & \ldots & \lambda_N \\ 1-\lambda_1 & \lambda_1-\lambda_2 & \lambda_2-\lambda_3 & \ldots & \lambda_N \end{bmatrix} \tag{4.15}$$

where $\lambda_i \in [0,1]$ are tuning parameters to account for preview uncertainties. This implies that instead of using the delayed structure of (4.4), $v_N(t)$ now evolves such that the value of v at each time step along the preview window is a convex combination of the value at that time and the values in the previous times. This effectively incorporates a "forgetting" term into the formulation to counteract the uncertainties in the preview information. The parameter λ_i is tuned according to the relative level of uncertainty on the i-th preview information, $r(i)$. Specifically, if the uncertainty

on $r(i)$ is large, λ_i is chosen to be close to 0. If this is the case, PRG with the new $\bar{A}$ matrix (i.e., (4.15)) will behave similarly to SRG. If r_i is very accurate, on the other hand, λ_i will be chosen close to 1. Then, PRG with the new $\bar{A}$ will turn to standard PRG. Therefore, robust PRG is a proper extension of both standard PRG and SRG. Typically, the level of uncertainty on r_i increases along the preview horizon, which means that the sequence of λ_i is increasing.

The construction of O_∞^N and the computation of v_N are the same as (4.5) and (4.10), except that $\bar{A}$ is changed from (4.4) to (4.15).

Proposition 3. *The robust PRG formulation to inaccurate preview information is BIBO stable, recursively feasible, and for a constant r, $v(t)$ converges.*

Proof. The proof of BIBO stable and recursively feasibility are the same as the proof for Proposition 1. To prove the convergence property, assume $r(t) = r, \forall t \in \mathbb{Z}_+$. Note that $\lim_{j\to\infty} \bar{A}^j = \bar{A}_0$, where $\bar{A}_0$ is a static matrix, since $\bar{A}$ shown in (4.15) is a stochastic matrix. Then, from (4.6), it can be shown that every element in $v_N(t)$ is monotonic increasing after $\bar{A}^j$ converges and bounded by r. Thus, $v_N(t)$ must converge to a limit. □

Illustration Example

Now, we will illustrate the performance of robust PRG to uncertain preview information using the one-link arm robot example (see the example in Section 4.1). First, a comparison on the Maximal Admissible set (MAS) between standard PRG and robust PRG is presented, followed by the numerical simulation of robust PRG.

Suppose, first, that the preview horizon, N, is equal to 1 for the ease of illustration.

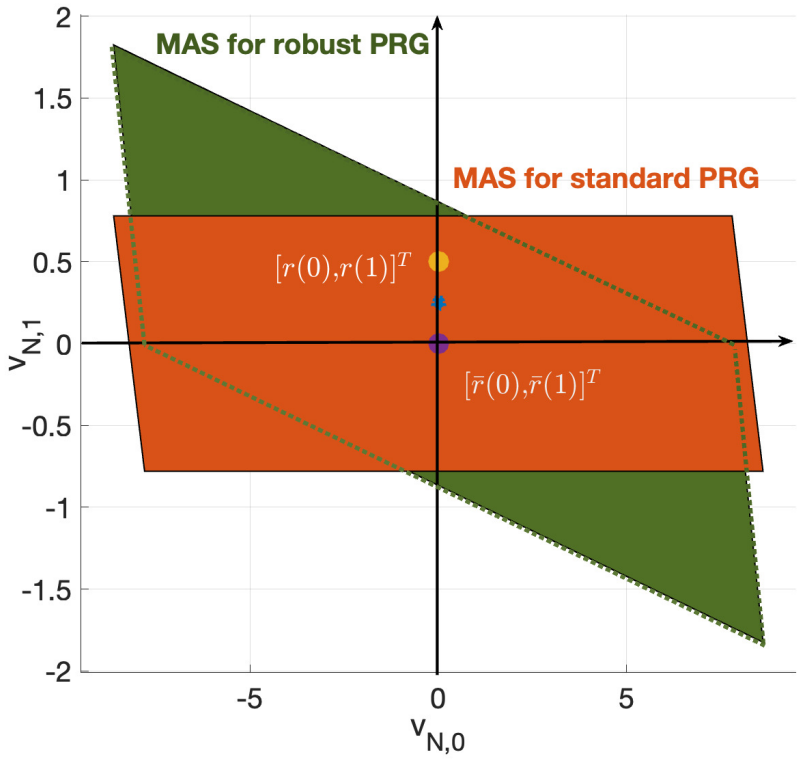

Figure 4.6: The slice of O_∞^N at $x = 0$. $v_{N,0}$ and $v_{N,1}$ represent the first and second element in v_N, respectively.

The slice of O_∞^N at $x = 0$ is shown in Figure 4.6. The red region corresponds to the slice of O_∞^N given by the standard PRG at $x = 0$. The green region represents the slice of O_∞^N given by the robust PRG at $x = 0$, where $\lambda_1 = 0.2$ is selected. Note that in this case, the matrix $\bar{A}$ has only one λ. The situation we consider is as follows. Suppose at $t = 0$, the preview information is given by $r_N(0) := (r(0), r(1)) = (0, 0.5)$ (yellow dot in Figure 4.6). However, assume that the *actual* preview information is $(\bar{r}(0), \bar{r}(1)) = (0, 0)$ (purple dot), which is unknown to the PRG. Note that $r(0) = \bar{r}(0)$ since current information is assumed to be accurate. Obviously, $v_N(0)$'s given by standard PRG and robust PRG will be equal to $r_N(0)$ (i.e., $v_N(0) = (0, 0.5)$) since

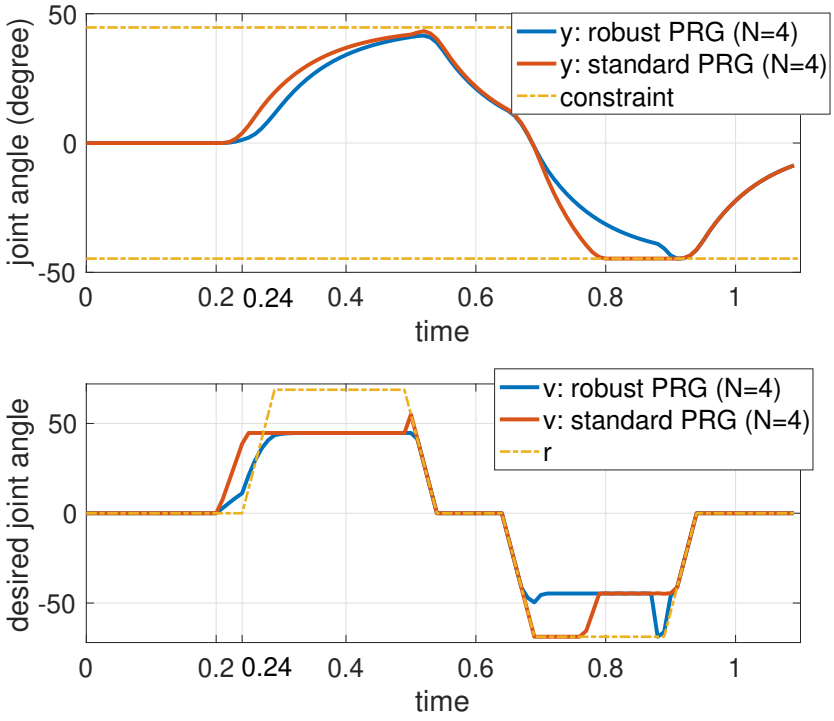

Figure 4.7: Comparison of standard PRG and robust PRG. Red lines represent the simulation results of standard PRG and blue lines refer to the simulation results of robust PRG.

$r_N(0)$ is in both MASs (red and green regions in Figure 4.6). In the next time step ($t = 1$), if $\kappa \neq 1$, $v_N(1)$ given by standard PRG is a convex combination between the delay version of $v_N(0)$ (i.e., $(0.5, 0.5)$) and $r_N(1)$ (see (4.7)). However, for robust PRG, $v_N(1)$ is a convex combination between $\bar{A}v_N(0) = (0.45, 0.45)$ and $r_N(1)$. Note that if the uncertainty is large, λ_1 would be chosen close to 1 and $\bar{A}v_N(0)$ will be close to $(0, 0)$. By doing so, robust PRG decreases the impact of the wrong preview information on the computation of current v_N by multiplying the previous information with values smaller than 1 (i.e., λ's).

We now perform numerical simulations of the one-link arm robot example. In this discussion, the preview horizon is changed from $N = 1$ to $N = 4$ for the sake

of illustration. We consider the scenario where the assumed preview information is larger than the actual preview information along the preview horizon. The $\bar{A}$ is chosen to be:

$$\bar{A} = \begin{bmatrix} 0.1 & 0.9 & 0 & 0 & 0 \\ 0.1 & 0.15 & 0.75 & 0 & 0 \\ 0.1 & 0.15 & 0.3 & 0.45 & 0 \\ 0.1 & 0.15 & 0.3 & 0.35 & 0.1 \\ 0.1 & 0.15 & 0.3 & 0.35 & 0.1 \end{bmatrix} \tag{4.16}$$

We acknowledge that there are other possibilities to choose $\bar{A}$ and finding the optimal set of λs is outside of the scope of this dissertation and will be explored in future work. The comparison on the system performance between standard PRG and robust PRG is duplicated in Figure 4.7. It can be seen that when $t \in [0.21, 0.24]$, $v(t)$ given by robust PRG (blue line) is closer to $r(t)$ than that given by standard PRG (red line). The reason for the behavior can be explained as follows. At $t = 0.20$, $v_N(20)$'s given by standard PRG and robust PRG are both equal to $r_N(20)$. Recall that the first element in $r_N(20)$ is accurate and the rest elements in $r_N(20)$ are inaccurate. Then, in the next time step, for standard PRG, $v(21)$ is calculated as a convex combination between the delayed version of $r_N(20)$ (inaccurate) and $r_N(21)$. From simulation results, with $\kappa = 0.25$, $v(21)$ is calculated as 0.13. For robust PRG, $v(21)$ is computed as a convex combination between $\bar{A}r_N(20)$, where $\bar{A}$ is shown in (4.16), and $r_N(20)$. With $\kappa = 0.69$, $v(21) = 0.05$. Also, by increasing the value of λ_1 in (4.16) (i.e., 0.1), $v(21)$ would become closer to $r(21) := 0$. However, the tracking performance would not be improved significantly as compared with SRG.

4.3 Computational considerations

In this section, a comparison between the computation time of PRG, Multi-N PRG, and SRG by simulating the one-link arm robot example with all three methods will be presented. Recall that SRG and PRG require the solution to one linear program (LP) at each time step, while Multi-N PRG requires the solutions to q LP's ($q = 26$ for our example). However, generic LP solvers is not used to solve them. Instead, we use the Algorithm presented in [149] to solve the LPs in SRG and Algorithm 3 below to solve the LPs in PRG. The LPs in Multi-N PRG can be solved using a similar algorithm as Algorithm 3. The notation in Algorithm 3 is as follows. It is assumed that O_∞^N is finitely determined and given by polytopes of the form (2.8). In addition, j^* denotes the number of rows of H_x, H_v, and h.

Algorithm 1 Custom Explicit PRG Algorithm

let $a = H_v(r_N(t) - \bar{A}v_N(t-1))$
let $b = h - H_x x(t) - H_v \bar{A} v_N(t-1)$
set $\kappa = 1$
for $i = 1$ to j^* **do**
  if $b(i) < 0$ **then**
    $\kappa = 0$
  else
    if $a(i) > 0$ **then**
      $\kappa = \min(\kappa, b(i)/a(i))$
    end if
  end if
end for
$\kappa = \max(\kappa, 0)$

We simulated the one-link arm robot example with all three methods, namely SRG, PRG, and Multi-N PRG. All simulations were performed for 150 time steps

Table 4.1: Comparison of the computation time between SRG, PRG, Multi-N PRG, and CG for one-link arm robot example

	SRG	**PRG** ($N = 25$)
avg	6.8×10^{-7}s	3.1×10^{-5}s
max	9.2×10^{-7}s	4.74×10^{-5}s
	Multi-N PRG ($N_q = 25$)	**CG** ($N = 25$)
avg	6.54×10^{-4}s	6.3×10^{-3}
max	8.5×10^{-4}s	0.0162

in Matlab on an Apple Macbook Pro with 1.4 GHz Intel Core i5 processor and 8 GB memory. To eliminate the effects of background processes running on the computer, each of the above experiments were run 10 times and the averages were calculated. The per-timestep averages and maximums of each of the three methods were calculated. The results are shown in Table 4.1. As can be seen, PRG runs two orders of magnitude slower than SRG (because the matrices that arise in the computations are larger). The Multi-N PRG is slower by one order of magnitude (because q PRGs are solved at each time step).

Finally, to provide a comparison of these computation times with those of other existing preview control methods, we simulate the one-link arm robot example with the PRG replaced by a Command Governor (as explained in Section 2.2.3). The QP is solved at every time step using explicit Multi-Parametric Quadratic Programming (MPQP), which is introduced in [75]. The computation time of CG is shown in Table 4.1. As can be seen, CG is one order of magnitude slower than Multi-N PRG. Note that we also implemented online QP solver for the CG, provided by MPT3 Toolbox in Matlab and Gurobi, and found that the computation times for both cases were longer than that of MPQP.

4.4 PRG FOR MULTI-INPUT SYSTEMS

In this section, we will introduce the extension of PRG to multi-input systems. One simple and straightforward solution is to apply the PRG ideas directly to multi-input systems. However, as will be shown in an example later, this approach might lead to a conservative response since PRG uses a single decision variable κ to simultaneously govern all the channels of the multi-input system. To address this issue, we propose another solution, which combines the PRG idea with the Decoupled Reference Governor (DRG) scheme (explained in Section 3.2). Detailed information will be introduced below.

To begin with, suppose that system $G(z)$ in Figure 4.1 has m inputs (i.e., $v(t), r(t) \in \mathbb{R}^m$). Let us denote the preview horizon for the m different inputs (i.e., $r_1, \ldots, r_m$) by $N_1, \ldots, N_m$, respectively. Define the lifted signals r_N and v_N as follows:

$$\begin{aligned} r_N(t) &= (r_1(t), \ldots, r_1(t+N_1), \ldots, r_m(t), \ldots, r_m(t+N_m)) \\ v_N(t) &= (v_1(t), \ldots, v_1(t+N_1), \ldots, v_m(t), \ldots, v_m(t+N_m)) \end{aligned} \tag{4.17}$$

The dynamics of $v_N(t)$ can be selected to be the same as (4.3) but with $\bar{A}$ constructed by:

$$\bar{A} = \begin{bmatrix} \bar{I}_{N_1} & \cdots & 0_{N_1 \times N_m} \\ \vdots & \ddots & \vdots \\ 0_{N_m \times N_1} & \cdots & \bar{I}_{N_m} \end{bmatrix} \tag{4.18}$$

where $\bar{I}_{N_i}$ $(i = 1, \ldots, m)$ is defined the same as (4.3), with N replaced by N_i. The construction of O_∞^N and the calculation of v_N are the same as (4.5) and (4.7), except that $\bar{A}$ is modified to (4.18).

Below, an illustrative example will be provided to show that this approach works

as required but might lead to a conservative response. Consider a two-link arm robot, which has dynamics as follows [168]:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -0.46 & -0.62 & 0 & 0 \\ 0.25 & -6.62 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.78 & -0.04 \\ 0.04 & 0.13 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{4.19}$$

where τ_1 and $x_1 := \theta_1$ represent the external torque and the joint angle for the first link, respectively. Similarly, τ_2 and $x_2 := \theta_2$ are the torque and the joint angle for the second link, respectively. The constraints are imposed on the joint angles: $\theta_1 \in [-60°, 60°]$ and $\theta_2 \in [-60°, 60°]$. To implement PRG, the system is first discretized at $T_s = 0.01s$. Then, a state feedback controller is designed to ensure that θ_1 and θ_2 track desired setpoints, v_1 and v_2, respectively, that is:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} 769.23 & 0 \\ 0 & 3333.3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} - \begin{bmatrix} 750 & 155 & 59 & 19 \\ -226 & 2867 & -18 & 350 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

The preview horizons for both references are chosen to be 40 (i.e., $N_1 = N_2 = 40$). The simulation results of PRG on the two-link arm robot are shown in Figure 4.8. As can be seen, the outputs both satisfy the constraints, as required. However, when $t \in [0.8, 1.2]$, v_2 give by PRG can not reach r_2, even though y_2 (i.e., red line in the top plot) is far from the lower constraint. This is caused by the fact that y_1 (i.e., the blue line in the top plot of Figure 4.8) reaches the constraint when $t \in [0.8, 1.2]$,

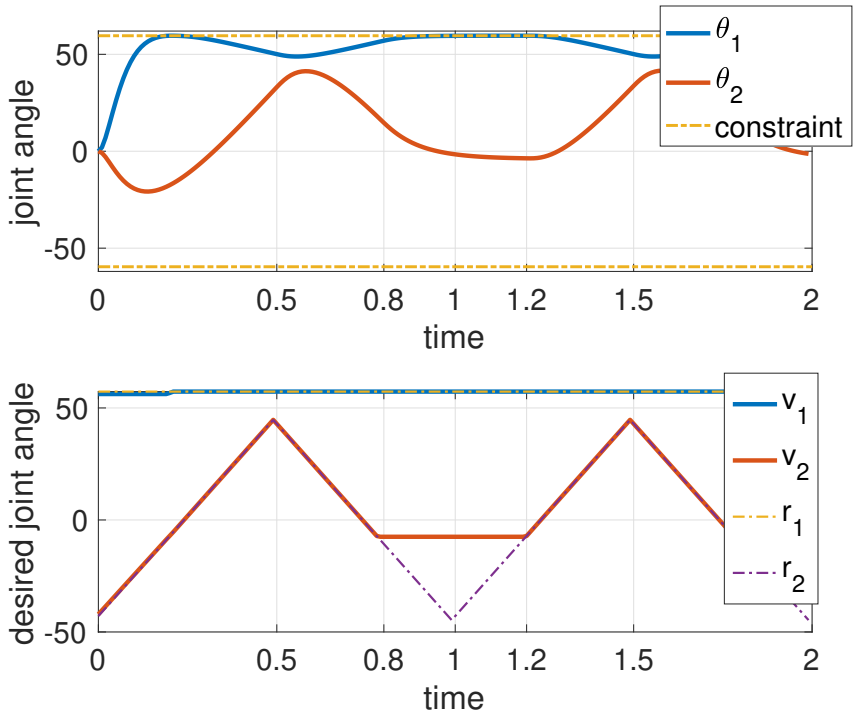

Figure 4.8: Simulation results of PRG for the two-link arm robot. The blue lines represent the response of joint 1 *and the red lines represent the response of joint* 2*.*

implying that κ is selected to be 0. Since a single κ is used in PRG scheme, v_2 can not reach r_2.

To address this shortcoming, another method is proposed below, which combines the PRG theory with the DRG scheme.

4.4.1 PRG Theory with the DRG Scheme

As a quick review, DRG is based on decoupling the input-output dynamics of the system, followed by the application of a bank of SRGs to each decoupled channel. In the solution proposed below, instead of using SRGs to govern the decoupled channels,

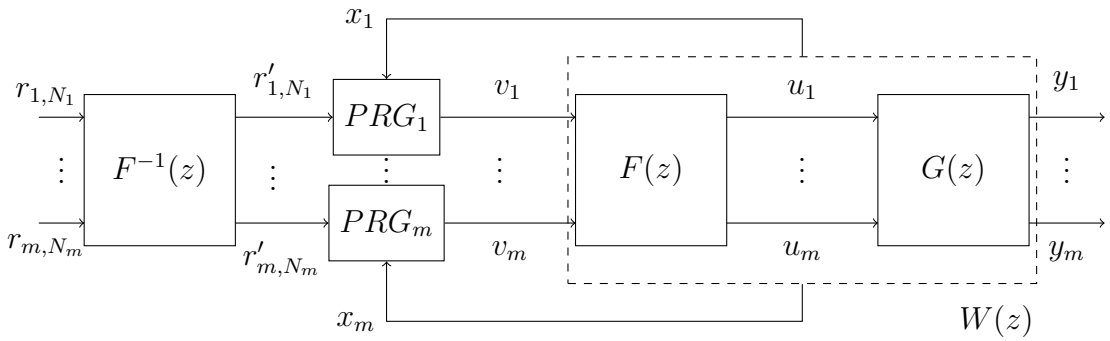

Figure 4.9: PRG block diagram for square MIMO systems. $r_{i,N_i}(t)$ represents the lifted r_i over the preview horizon, i.e., $r_{i,N_i}(t) = (r_i(t), \ldots, r_i(t+N_i))$.

we use the PRG presented in Section 4.1. This leads to the block diagram shown in Figure 4.9. For ease of presentation, it is assumed that the system is square. Non-square case can be handled as well by considering the DRG scheme for non-square systems presented in Section 3.4.

To elaborate, suppose the closed-loop system $G(z)$ is described using (2.16). Similar to DRG, as shown in Figure 4.9, the dynamics of $G(z)$ is decoupled by introducing $F(z)$, leading to a diagonal system: $W(z)$. As discussed in Section 2.5, there are several ways to construct $W(z)$. For the sake of brevity, we only consider the case where $W(z)$ is given by: $W(z) = \text{diag}(G_{11}, G_{22}, \ldots, G_{mm})$. Then, m different PRGs for single-input systems (see Section 4.1) are implemented, one for each W_{ii}. Finally, as discussed in Section 2.5, F^{-1} is introduced to ensure that $v = r$ if r is constraint-admissible. Note that the same process to combine PRG idea and DRG scheme can be applied to other structure of $W(z)$ as well.

To elaborate the DRG-tf idea, the two-link arm robot is transferred to frequency domain. Simulation results of the second approach on the two-link arm robot are shown in Figure 4.10. As can be seen, the constraints are satisfied, as required. Also,

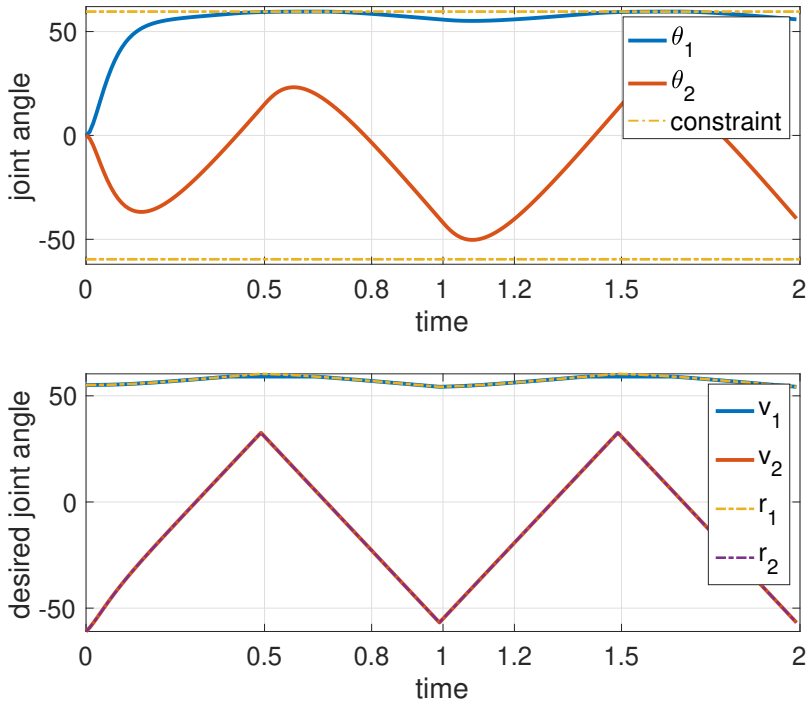

Figure 4.10: Simulation results of PRG for the two-link arm robot. The blue lines represent the response of joint 1 *and the red lines represent the response of joint* 2.

differs from the first approach, when $t \in [0.8, 1.2]$, v_2 reaches r_2. This behavior can be explained as follows. When $t = 0.8$, the PRGs have the future information that r_2 will drop down to -56 at $t = 1$ and then go up towards -20 at $t = 1.2$ (recall that $N_1 = N_2 = 40$). Also, from the top plot of Figure 4.10, it can be seen that when $t \in [0.8, 1.2]$, y_2 (red line in the top plot) does not reach the lower constraint, which leads to $\kappa_2 = 1$. Hence, v_2 reaches r_2.

Finally, it is worthwhile to mention that both extensions of PRG to multi-input systems discussed above can be easily extended to multi-N PRG by combining DRG scheme with Multi-N PRG idea.

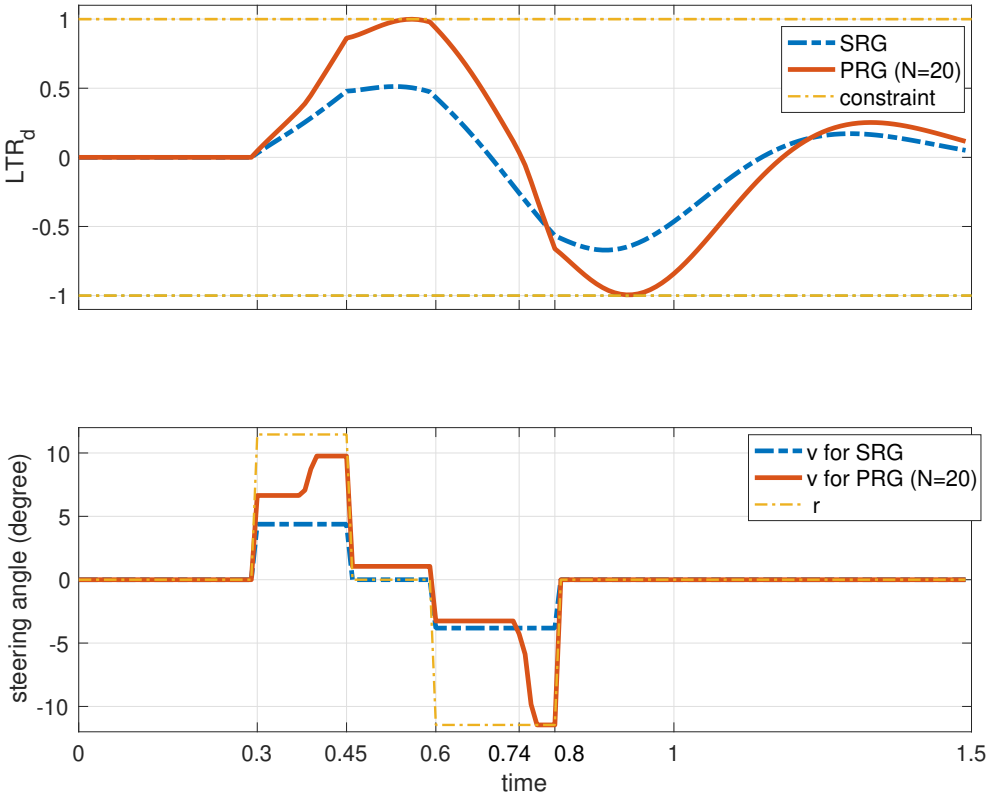

Figure 4.11: Comparison of PRG and SRG. The red lines represent the results of PRG and the blue dash lines refer to the results of SRG.

4.5 Rollover Prevention: PRG

In this section, the implementation of PRG on the rollover prevention example is presented. As reviewed in Section 2.6.2, to prevent the vehicle from rollover, the Load Transfer Ratio (LTR) should be within $[-1, 1]$. The state space matrices A, B, C, and D are obtained from (2.29) combined with Table 2.1.

To implement the PRG scheme, the system dynamics is first discretized at $T_s = 0.01s$. The preview horizon, N, is chosen to be 20 (i.e., 0.2 seconds). This preview

horizon is reasonable according to the work of [169]. The obstacle avoidance maneuver is considered, which takes place at a speed of $v = 144$ km/h. The comparison between the performance of PRG and SRG is shown in Figure 4.11.

It can be seen that, from Figure 4.11, $v(t)$ given by PRG is closer to $r(t)$ (less conservative) when $t \in [0.3, 0.45]$. This is because when $t = 0.3$, the PRG has the future information that the reference would drop down to 0 after 16 time steps so that it allows higher reference (i.e., $v(t)$) than SRG. For the same reason, $v(t)$ given by PRG is less conservative when $t \in [0.74, 0.8]$.

Figure 4.11, however, also shows a limitation of PRG. It can be seen that when $t \in [0.46, 0.59]$, $v(t)$ given by PRG can not reach $r(t) = 0$, even though 0 is an admissible input. The reason can be explained the same as that for one-link arm robot example (explained in Section 4.1). More specifically, note that when $t = 0.46$, the preview information available to the PRG is that $r(t)$ drops to -11.4 at $t = 0.6$ and stays constant afterwards. To enforce the lower constraint for $t > 0.6$, the PRG calculates a κ smaller than 1. However, since κ affects all elements of $v_N(t)$, this leads to a $v(t)$ that is different from $r(t)$ at $t = 0.46$, leading to a conservative solution.

Next, the simulation results of Multi-N PRG method on the vehicle rollover prevention example will be demonstrated. For the sake of illustration, the extreme case where the Multi-N PRG uses all preview horizons between 0 and 20; i.e., $q = 21$ and $N_1 = 0$, $N_2 = 1$, ..., $N_{21} = 20$, is considered.

The simulation results is duplicated in Figure 4.12. For comparison, the results of the implementation of PRG with $N = 20$ is also provided. As Figure 4.12 shows, the outputs for both Multi-N PRG and PRG satisfy the constraints, as expected. However, from the bottom plot of Figure 4.12, it can be seen that when $t \in [0.45, 0.6]$,

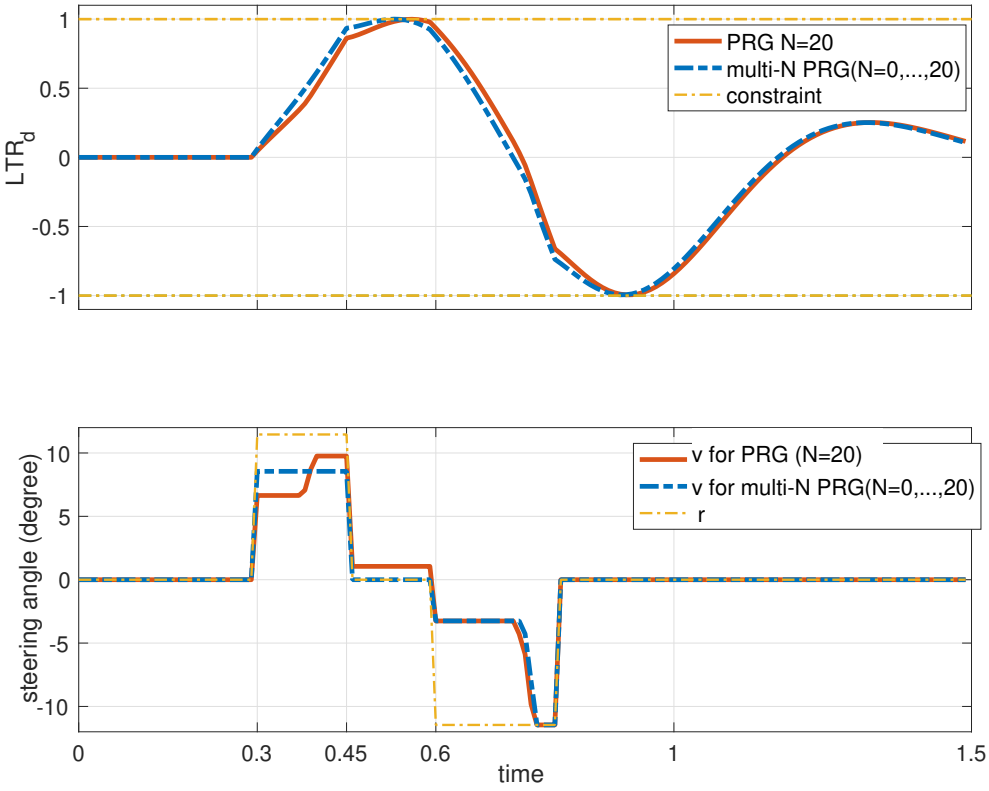

Figure 4.12: Comparison of Multi-N PRG and PRG with $N = 20$. The top plot shows the outputs and the bottom plot shows the setpoint and the governed setpoint.

$v(t)$ given by Multi-N PRG reaches $r(t)$ while $v(t)$ computed by PRG is above $r(t)$. The reason for this behavior is that when $t \in [0.45, 0.6]$, the PRG corresponding to $N_1 = 0$ computes $\kappa = 1$, which leads to $v(t) = r(t)$.

Chapter 5

Nonlinear Reference Governor for MIMO systems

In this chapter, the extensions of Nonlinear Reference Governor as reviewed in Section 2.2.4 and Scalar Reference Governor (SRG) to nonlinear Multi-Input Multi-Output systems (MIMO) will be introduced, especially focusing on quadcopter applications. More specifically, as highlighted in Section 1.4, three different approaches will be presented below, namely Nonlinear Decoupled Reference Governor (NL-DRG), Modified Reference Governor (M-RG), and Neural Network DRG (NN-DRG). In the NL-DRG scheme, a bank of NL-RGs are implemented, where the constraint for each is tightened to account for the worst-case coupling behavior among different channels. However, due to the iterations to find the optimal decision variable and the implementation of multiple NL-RGs, NL-DRG tends to be more computationally expensive than NL-RG and SRG. To address the above limitation of NL-DRG, M-RG is proposed, where multiple SRG are solved and the constraint for each SRG is shrunk to to take the worst-case realization of the coupling behavior as well as the linearization error

into consideration. Obviously, NL-DRG will obtain a less conservative response than M-RG but at a cost of larger computational effort. To address the response issue of M-RG while maintaining a low computational effort. The NN-DRG is proposed, where a well-trained NN is utilized the approximate the input-output mapping of NL-DRG. And a Quadratic Program (QP) is solved to augment the output of NN so that the constraints are satisfied at next time step. For the sake of clarity, in the following discussion, the quadcopter application will be the mainly focus to build the proposed algorithms. However, it is emphasized that the proposed solutions are not necessarily limited to quadcopters and can be applied to constraint management of general nonlinear MIMO systems that are mildly nonlinear, have mild interactions between the different input/output channels, and for which a relatively good dynamical model is available.

5.1 Nonlinear Decoupled Reference Governor (NL-DRG)

The block diagram of NL-DRG for the quadcopter application is shown in Figure 5.1. As can be seen from the diagram, the NL-DRG scheme is comprised of three sequential NL-RGs, one for each input/output pair. Each NL-RG calculates a desired setpoint (either pitch, roll, or yaw rate) to enforce the constraints on the corresponding output. As such, we model the interactions among the various input-output channels by a fictitious bounded exogenous disturbance and robustify each NL-RG against this disturbance such that the constraints are satisfied despite the worst-case possible coupling behavior. Each NL-RG is also robustified against the worst case effect of

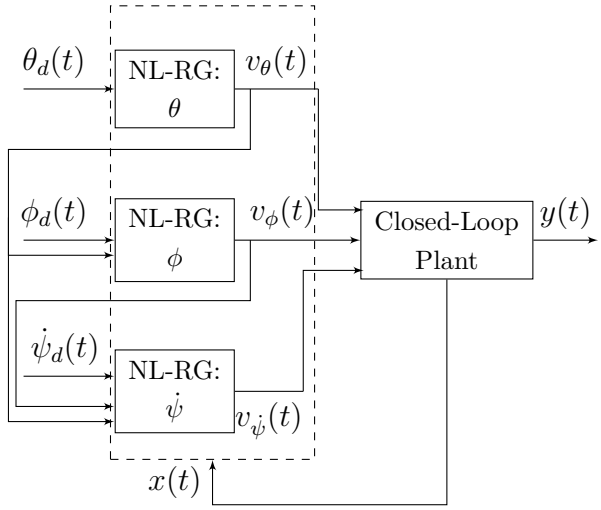

Figure 5.1: Block diagram of NL-DRG for quadcopter constraint management. The dashed box represents the NL-DRG algorithm.

the disturbance d and the measurement noise N_s (shown in Figure 2.7). More details on the approach is provided next.

5.1.1 Quantification of disturbance and sensor noise

Consider the *closed-loop* quadcopter control system shown in Figure 2.7 and described by Eqs. (2.30)-(2.38) and controlled by inner loop controllers shown in Figures 2.8-2.10. This closed-loop system can be discretized (e.g., using forward Euler or the exponential map methods with sufficiently fast sampling rate) and expressed in the following standard form:

$$\begin{aligned} x(t+1) &= f(x(t)) + g(x(t))v(t) + g_w(x(t))w(t), \\ y(t) &= h(x(t)) + h_w(x(t))w(t) \end{aligned} \tag{5.1}$$

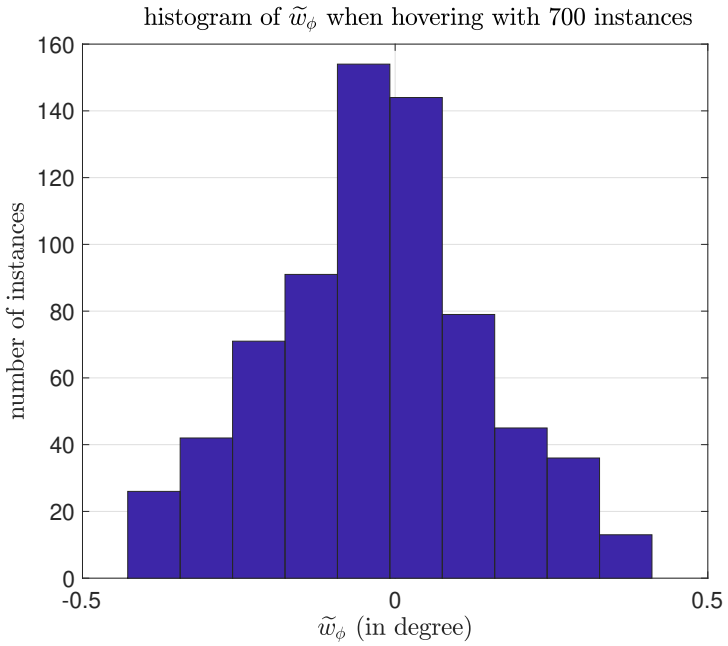

Figure 5.2: This figure shows the histogram of $\widetilde{w}_\phi$ when the Crazyflie is hovering

where x are the states (i.e., the three body angles, the three body angular rates, and the controller integral and derivative filter states), y are the constrained outputs (i.e., y_ϕ, y_θ, and $y_{\dot{\psi}}$), v are the setpoint commands (i.e., v_ϕ, v_θ, and $v_{\dot{\psi}}$), and w is the vector comprised of disturbance d and measurement noise N_s shown in Figure 2.7. For the Crazyflie application, the dimensionality of the various variables is as follows: $x \in \mathbb{R}^{12}$, $y \in \mathbb{R}^3$, $v \in \mathbb{R}^3$, and $w \in \mathbb{R}^{15}$. Since the disturbance and noise are typically bounded in magnitude, it is assumed that $w \in \mathbb{W}$, where $\mathbb{W} \subset \mathbb{R}^d$ is a compact polytope with the origin in its interior. For the remainder of this thesis, the closed-loop system with disturbance and noise is denoted by Σ, and the system without noise and disturbance (i.e., w set equal to zero, which represents the ideal model) is referred by Σ_{disfree}.

In the traditional NL-RG scheme, in order to account for $w(t)$, the bisectional

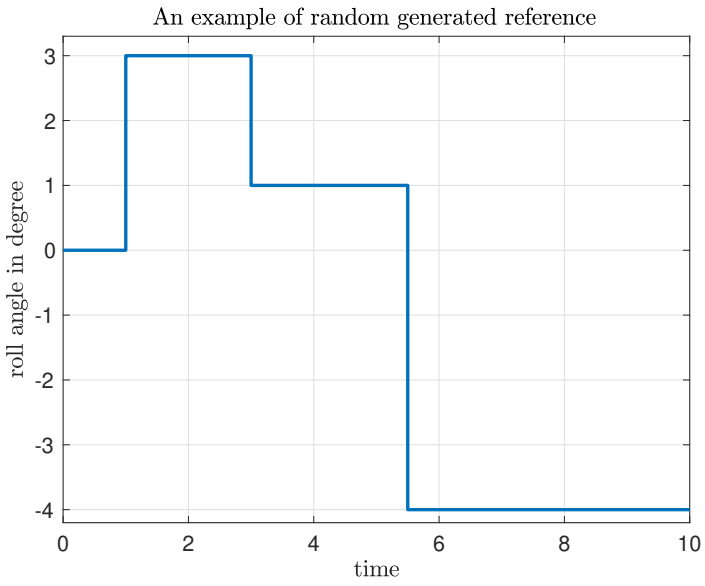

Figure 5.3: This figure shows an example of randomly generated time-series reference for roll angle ($N = 1000$ points and $T_s = 0.01s$)

search algorithm (as shown in Figure 2.1) would be modified in one of two ways. If disturbances affect the states, then in every iteration of NL-RG bisection, multiple simulations with different realization of $w(t)$ must be carried out, which is impractical for real-time implementation. If the disturbance is on the output only, then the constraints can simply be tightened to account for the worst-case realization of the output disturbance, and only one simulation would be run in every iteration of the bisection search, which is more numerically efficient. In this thesis, the latter approach is taken for the sake of computational feasibility despite the fact that the quadcopter system shown in Figure 2.7 has disturbance and noise on both the states and the output. To achieve this, our novel solution is to “convert" the set-bounded state disturbance into a set-bounded output disturbance, which inevitably introduces some degree of conservatism. As will be show later, this conservatism is acceptable for the

quadcopter application. More specifically, (5.1) is rewritten as:

$$\begin{aligned} \widetilde{x}(t+1) &= f(\widetilde{x}(t)) + g(\widetilde{x}(t))v(t), \\ y(t) &= h(\widetilde{x}(t)) + \widetilde{w}(t) \end{aligned} \tag{5.2}$$

where $\widetilde{x}$ is different from x in (5.1) because now $d(t) = 0$. Thanks to the fact that the system (5.1) is pre-stabilized, bounded disturbance input d leads to bounded changes in the output y, which means $\widetilde{w}$ exists and is bounded: $-\widetilde{W} \leq \widetilde{w} \leq \widetilde{W}$. To characterize the bound $\widetilde{W}$ numerically, we carry out numerous experiments on the real Crazyflie 2.0 and also simulate system Σ_{disfree} (i.e., (5.1) with $w = 0$) for the same inputs. The difference between the output of the Crazyflie and that of the simulated system is exactly $\widetilde{w}$ in (5.2). A histogram of $\widetilde{w}$ from numerical study is shown in Figure 5.2 for the roll channel. The maximum value of $\widetilde{w}$ for all time-instants in all the experiments provides an approximation of $\widetilde{W}$.

5.1.2 Quantification of coupling behavior

As mentioned before, to implement three sequential NL-RGs (see Figure 5.1), the interactions among different channels must be modeled by set-bounded exogenous disturbances. By doing so, the constraint set for each NL-RGs can be shrunk to take the worst-case realization of this disturbance into consideration and thus enforce the constraints. Below, we elaborate on how this can be done. For this discussion, we focus on the roll channel (same arguments can be applied to pitch and yaw).

The roll output as presented in (5.2) can be equivalently expressed by:

$$\begin{aligned} \widetilde{x}_\phi(t+1) &= f(\widetilde{x}_\phi(t)) + g(\widetilde{x}_\phi(t)) \begin{bmatrix} v_\phi(t) \\ 0 \\ 0 \end{bmatrix}, \\ y_\phi(t) &= h_\phi(\widetilde{x}_\phi(t)) + \widetilde{w}_\phi(t) + d^c_\phi(t) \end{aligned} \tag{5.3}$$

where d^c_ϕ now captures the perturbations in the roll angle caused by the interactions from the pitch and yaw channels, $\widetilde{x}_\phi$ denotes the states of the system with input as $v(t) = [v_\phi(t), 0, 0]^T$, h_ϕ and $\widetilde{w}_\phi$ represent the first element of vectors h and $\widetilde{w}$ respectively (i.e., the element corresponding to the roll angle), and y_ϕ is the first element of y in (5.2). Clearly, $\widetilde{x}_\phi$ is different from $\widetilde{x}$ in (5.2) because the two systems are driven by different inputs, but the roll angle outputs are the same because any difference between them is captured by the fictitious disturbance d^c_ϕ. Moreover, d^c_ϕ must be set-bounded by the same argument as the previous subsection, i.e., $-\bar{d}^c_\phi \leq d^c_\phi \leq \bar{d}^c_\phi$. The bounds of d^c_ϕ can be quantified either by analytically finding the worse case coupling using (2.32) and (2.30) (in an open-loop setting), or by simulating (5.3) a large number of times with different references and numerically evaluating the worst-case. In this thesis, the latter route is taken because, with the first route, the system performance became too conservative (i.e., $\bar{d}^c_\phi$ was too large) since the extreme value on the angles and angle rates needed to be considered simultaneously, which will not occur in practice.

To numerically find $\bar{d}^c$, we simulate (5.3) with 0 initial condition (i.e., hovering) for a sufficiently large number of times. For each simulation, the setpoints, denote by $\phi_d(t)$, $\theta_d(t)$, and $\dot{\psi}_d(t)$ ($t = 1, \ldots, N$ and N represents a given finite time horizon for

Algorithm 2 Quantify the maximum coupling behavior for the roll angle. J_1 represents the number of simulations with different ϕ_d. J_2 represents the number of simulations with different θ_d and $\dot{\psi}_d$.

1: $\bar{d}_{\phi}^{c} = 0$
2: The initial condition is set to the 0
3: **for** $i \leftarrow 1$ to J_1 **do**
4: The inputs (i.e., ϕ_d, θ_d, and $\dot{\psi}_d$) are generated as three sequences of steps with random heights and times, where the heights are uniformly distributed random numbers with the bounds defined by the constraint sets. The times are integers from the uniform distribution on the set $0 : N$. An example of the random generated time-series reference is shown in Figure 5.3.
5: The disturbance $\widetilde{w}_{\phi}$, whose histogram is shown in Figure 5.2, is generated as one sequence with N timestep and the height for each timestep is normally distributed random number with mean and variance obtained from the empirical distribution, which can be found experimentally using data collected from the Crazyflie.
6: **for** $j \leftarrow 1$ to J_2 **do**
7: Simulate (5.2) with 0 initial condition, ϕ_d, θ_d, $\dot{\psi}_d$, and $\widetilde{w}_{\phi}$ over the finite time horizon N, denoted the output for roll angle as $y_{\phi}(t)$.
8: Choose $d_{\phi}^{c}(t)$ properly such that the simulation output of (5.3) with $\widetilde{w}_{\phi}(t)$ is equal to $y_{\phi}(t)$.
9: Define $d_{tmp} = \max\limits_{t \in \{1,\ldots,N\}} (|d_{\phi}^{c}(t)|)$ and redefine $\bar{d}_{\phi}^{c}$ as $\bar{d}_{\phi}^{c} = \max(d_{tmp}, \bar{d}_{\phi}^{c})$.
10: Repeat Step 4 to get different sequences of θ_d and $\dot{\psi}_d$.
11: **end for**
12: **end for**

the simulation), are randomly generated time-series sequences consisting of different steps at different times. Thanks to the fact that the DC-gain of the three input/output channels (from setpoints to outputs) are all equal to 1, the randomly generated setpoints are selected to be bounded within the corresponding constraint sets. The maximum coupling behavior on the roll angle ($\bar{d}^c_\phi$) is characterized as the maximum difference between the simulated d^c_ϕ with inputs ϕ_d, $\theta_d = 0$, $\dot{\psi}_d = 0$, and $\widetilde{w}_\phi$ and the simulated d^c_ϕ of the same system but with different randomly generated θ_d and $\dot{\psi}_d$. The values of $\bar{d}^c_\theta$ and $\bar{d}^c_{\dot{\psi}}$ can be found similarly. The detailed process is presented in Algorithm 2 for the roll angle. Note that, because we apply steps of different heights at random times within the same simulation, we are effectively taking the system to various states, which means it is not necessary to consider non-zero initial conditions in the algorithm.

5.1.3 NL-DRG

As seen in Figure 5.1, the inputs of the NL-DRG scheme are the states of the quadcopter $x(t)$, the previous governed references given by NL-DRG (i.e., $v_\phi(t-1)$, $v_\theta(t-1)$, and $v_{\dot{\psi}}(t-1)$, though these are omitted from the figure to ensure visual clarity), and the desired setpoints (i.e., $\phi_d(t)$, $\theta_d(t)$, and $\dot{\psi}_d(t)$). The outputs are the governed, constraint-admissible references $v_\phi(t)$, $v_\theta(t)$, and $v_{\dot{\psi}}(t)$. In real time, at each time step t, three NL-RGs are solved sequentially. Each NL-RG leverages the update law (2.13), where the κ in each is obtained by a bisectional algorithm as reviewed in Section 2.2.4. The constraint in each NL-RG is tightened by $\bar{d}$, where $\bar{d} := [\bar{d}_\phi, \bar{d}_\theta, \bar{d}_{\dot{\psi}}]^T = \bar{d}^c + \widetilde{W}$ is quantified offline using Algorithm 2. Furthermore, the input to each NL-RG inside the NL-DRG scheme is either the setpoint from the previous timestep or the set-

point calculated by an upstream NL-RG, with the prioritization order of pitch, roll, then yaw rate. Detailed information is provided in Algorithm 3 and further discussed below.

Algorithm 3 NL-DRG at Timestep t

1: Implement the bisectional search algorithm to find $v_\theta(t)$ such that $-s_\theta + \bar{d}_\theta \leq \theta \leq s_\theta - \bar{d}_\theta$, where s_θ is the constraint value for the pitch angle. For each iteration of the bisection search, simulate Σ_{disfree} with $x(t)$, $v_\phi(t-1)$, and $v_{\dot{\psi}}(t-1)$ over a finite time horizon.
2: Implement the bisectional search algorithm to find $v_\phi(t)$ such that $-s_\phi + \bar{d}_\phi \leq \phi \leq s_\phi - \bar{d}_\phi$. For each iteration of the bisection search, simulate the plant with $x(t)$, $v_\theta(t)$, and $v_{\dot{\psi}}(t-1)$ over a finite time horizon.
3: Implement the bisectional search algorithm to find $v_{\dot{\psi}}(t)$ such that $-s_{\dot{\psi}} + \bar{d}_{\dot{\psi}} \leq \dot{\psi} \leq s_{\dot{\psi}} - \bar{d}_{\dot{\psi}}$. For each iteration of the bisection search, simulate the plant with $x(t)$, $v_\phi(t)$, and $v_\theta(t)$ over a finite time horizon.

Remark 6. *In Algorithm 3, the three sequential NL-RGs are computed based on the order of θ, ϕ, and $\dot{\psi}$, which means that θ is prioritized over ϕ and ϕ takes precedence over $\dot{\psi}$. In other words, NL-DRG computes the governed references so that v_θ is as less conservative as possible. The reason for choosing this prioritization order is that θ and ϕ are more important than $\dot{\psi}$ as they directly affect stability and maneuverability of the drone. Also, the pitch angle, which affects the backward and forward motion of the quadcopter, typically plays a more important role when piloted by a human. Note that depending on different prioritization, the order in Algorithm 3 can be changed.*

Proposition 4. *Asymptotically (i.e., as $J_1 \to \infty$ and $J_2 \to \infty$ in Algorithm 2), NL-DRG guarantees constraint satisfaction, bounded-input bounded-output stability (BIBO), and convergence for constant references.*

Proof. The proof for constraint satisfaction is straightforward. Take the roll angle as an example. Recall the robustification of the constraint set: $-s_\phi + \bar{d}_\phi \leq \phi \leq s_\phi - \bar{d}_\phi$.

Combining this with the fact that the disturbance on the roll angle caused by the coupling behavior, d, and noise, N_s, is bounded by $\bar{d}_\phi$, it is guaranteed that, no matter how v_θ and $v_{\dot{\psi}}$ are varying and/or how d and N_s are changing, if the pair $(x(t), v_\phi(t-1))$ is constraint-admissible with respect to s_ϕ, then, so is $(x(t), v_\phi(t))$ at the computed value of κ. Furthermore, in the iterations of the bisectional algorithm within the NL-RG, the inputs are held constant along the simulations, meaning that $\kappa = 0$ would always be a feasible solution at each timestep, which shows existence of a $\kappa \in [0, 1]$ to enforce constraints. Similar proofs can be obtained for θ and $\dot{\psi}$. As for the BIBO stability and convergence for a constant reference, the proofs are similar as those for RG, which can be found in [58]. Essentially, because the NL-RG uses the update law as shown in (2.13) with $\kappa \in [0, 1]$, each $v(t)$ is a bounded monotonic sequence for a constant reference, which has a limit. □

Obviously, in practice, it is not possible to collect an infinite amount data to quantify the disturbance in Algorithm 2. As such, a small amount of additional safety margin should be introduced (in addition to $\bar{d}^c + \widetilde{W}$) in order to ensure constraint satisfaction in the finite-data regime.

One drawback of NL-DRG is that, since three NL-RGs are computed at every timestep, the computational demand for NL-DRG is higher than that in both NL-RG and RG. Below, an alternative approach (namely M-RG), which is more computationally efficient, will be introduced.

5.2 Modified Reference Governor (M-RG)

Recall that the main idea behinds M-RG is to employ a bank of SRGs, where the constraints in each is tightened to take the worst-case realization of coupling behavior as well as linearization error into consideration. Below, the quantification of the linearization error is introduced followed by the explanation of M-RG.

5.2.1 Quantification of linearization error

Algorithm 4 Quantify the linearization error for the roll angle. J_1 represents the number of simulations with different inputs.

1: $d_\phi^e = 0$
2: x_0 is set to be 0
3: **for** $i \leftarrow 1$ to J_1 **do**
4: The inputs (i.e., ϕ_d, θ_d, and $\dot{\psi}_d$) are generated as three sequences of steps with random heights and times, where the heights are uniformly distributed random numbers with the bounds defined by the constraint sets. The times are integers from the uniform distribution on the set $0 : N$.
5: The disturbance $\widetilde{w}_\phi$, whose histogram is shown in Figure 5.2, is generated as one sequence with N timestep and the height for each timestep is normally distributed random number with mean and variance obtained from the empirical distribution, which can be found experimentally using data collected from the Crazyflie.
6: Simulate both Σ and the linearized closed-loop quadcopter system (i.e., (5.4)) with x_0, ϕ_d, θ_d, and $\dot{\psi}_d$ over the finite time horizon N. The simulation outputs on the roll angle are denoted by $\hat{\phi}(t)$ for the nonlinear system and $\tilde{\phi}(t)$ for the linear system.
7: Define $d_{tmp} = \max_{t\in\{1,\dots,N\}}(|\hat{\phi}(t) - \tilde{\phi}(t)|)$ and set $d_\phi^e = \max(d_{tmp}, d_\phi^e)$.
8: **end for**

To elaborate this idea, the nonlinear dynamics as shown in (5.2) are first linearized

at a equilibrium point:

$$\begin{aligned} x_{lin}(t+1) &= A_{lin}x_{lin}(t) + B_{lin}v(t), \\ y_{lin}(t) &= C_{lin}x_{lin}(t) + \widetilde{w}(t) \end{aligned} \tag{5.4}$$

where x_{lin} denotes the states of the linearized system. The maximum linearization error (d^e) refers to the maximal difference of the output between the linearized and nonlinear system, both driven by the same initial conditions and desired inputs. This error can be calculated as shown in Algorithm 4. Note that another route to quantify d^e is by finding the difference between the outputs of (5.2) and (5.4), and analytically finding the maximal error. However, this approach may lead to a conservative d^e since the extreme value on different states needs to be considered simultaneously.

5.2.2 M-RG

Recall that the problem of implementing separate SRGs on the nonlinear system is that the linearization errors and coupling behaviors are not accounted for. In M-RG, this issue is overcame by shrinking the constraint sets (i.e., (2.2)) for each SRG by $d_{\text{all}} := d^c + d^e + \widetilde{W}$ (see Algorithm 2 and Algorithm 4). Note that the M-RG is applicable if $d_{\text{all}} < s$, otherwise, the MAS's for the three SRGs will be empty, which means no admissible v will be obtained or v will be calculated as 0 at every time step.

In conclusion, M-RG, where sequential SRGs are implemented, may lead to a more conservative system response in exchange for a less computational cost. Below, a new RG-based scheme is pursued that has a superior performance than M-RG while maintaining a highly-attractive computational feature, namely NN-DRG.

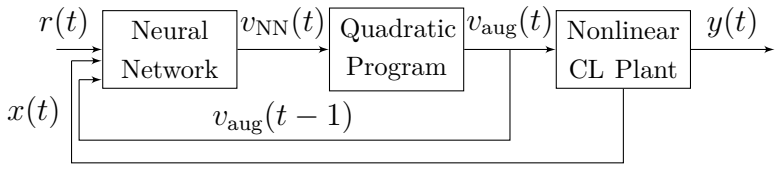

Figure 5.4: Block diagram of NN-DRG.

5.3 Neural Network Decoupled Reference Governor (NN-DRG)

The block diagram of NN-DRG is shown in Figure 5.4. The essential idea behind NN-DRG is to replace NL-DRG (i.e., Algorithm 3) with a well-trained Neural Network representing the input/output mapping of the NL-DRG. To achieve this, during the design stage, numerous data samples are collected offline by simulating the NL-DRG applied to Σ (i.e (5.1)) with randomly generated time-series references and noise a large number of times. For each simulation, the references $\phi_d(t)$, $\theta_d(t)$, and $\dot{\psi}_d(t)$, where $t = 1, \ldots, N$ with $N \in \mathbb{Z}_+$ being a finite simulation window, are generated as discussed in step 4 of Algorithm 2. The input dataset, which will be used to train the NN, consists of the time series vector $[x(t), v(t-1), r(t)]^T$ (i.e., the inputs of the NL-DRG scheme) and the output dataset is the time series data $v(t)$ (i.e., the output of the NL-DRG). The motivation behind choosing Σ as the simulation model as opposed to Σ_{disfree} (i.e (5.1) with $w = 0$) will be discussed in Remark 8). After the data samples are collected, a feedforward neural network with one hidden layer (as reviewed in Section 2.4). The number of neurons and the activation functions will be discussed in Section 6.1. In real-time, the neural network output is calculated using standard forward propagation.

However, due to the training errors caused by the potentially small size of the network and the incomplete training data, running the quadcopter with the NN alone may not guarantee constraint satisfaction. To address this issue, as shown in Figure 5.4, the NN output $v_{NN}(t)$ is further augmented at each timestep by solving a quadratic program (QP), which computes a $v_{aug}(t)$ that is constraint-admissible in the next timestep. To be specific, reconsider the state space representation of the nonlinear system as shown in (2.15). Assume that, at current timestep t, the state $x(t)$ is measured or estimated from the plant. Then, $y(t+1)$ can be expressed as:

$$y(t+1) = h\left(f(x(t)) + g(x(t))v(t)\right) \tag{5.5}$$

The QP aims to find $v_{aug}(t)$ that is as close as possible to $v_{NN}(t)$ while satisfying the constraints at next timestep:

$$\begin{aligned} \underset{v_{aug}(t)}{\text{minimize}} \quad & (v_{NN}(t) - v_{aug}(t))^T Q (v_{NN}(t) - v_{aug}(t)) \\ \text{s.t.} \quad & -s \leq h(f(x(t)) + g(x(t))v_{aug}(t) \leq s \end{aligned} \tag{5.6}$$

where $s = [s_\phi, s_\theta, s_\psi]^T$ and Q is a design parameter.

Remark 7. *From* (5.5)*, it can be seen that, if h is linear, which is the case for quadcopter application (i.e., the output is a subset of the state), then, the mapping from $v_{aug}(t)$ to $y(t+1)$ is also linear. Thus, the optimization problem as shown in* (5.6) *can be solved using a standard QP solver. On the other hand, if $h(x)$ is nonlinear, which is the more general case, then* (5.6) *can be solved either by linearizing the system dynamics around the constraints and applying the QP* (5.6) *on the linearized dynamics or using a nonlinear programming solver, such as an interior-point algorithms [170].*

As we will show in Section 6.1.5, NN-DRG requires significantly lower computational effort than NL-DRG and can enforce the constraints. However, it may cause relatively large, sudden changes in v_{aug} since only one-timestep prediction of the output is involved in the QP. This effect can be minimized if certain conditions are met. Below, an analysis on the performance of NN-DRG will be provided. It will be shown that if: 1) the number of training samples is large and the dataset covers most of the state space; 2) the Lipschitz constant of the NN is small; 3) the worst case training error is small, then, the distance between $v_{NN}(t)$ and $v_{aug}(t)$ (as shown in Figure 5.4) will also be small.

First, several notations need to be introduced. The training dataset of NN (the number of training samples is M) is denoted by:

$$T = \{(x_1, v_{pre,1}, r_1), \ldots, (x_M, v_{pre,M}, r_M)\}$$

where x_i, $v_{pre,i}$, r_i $(i \in 1, \ldots, M)$ represent the states, the previous governed input of NL-DRG, and the desired references, respectively. Then, the maximum training error of the NN is defined as:

$$e_{training} = \max\{\|v_1 - v_{NN,1}\|, \|v_2 - v_{NN,2}\|, \ldots, \|v_M - v_{NN,M}\|\}$$

where v_i and $v_{NN,i}$ represent the output given by NL-DRG and NN with the same input $[x_i, v_{pre,i}, r_i]^T$, respectively. At each timestep t, the measured or estimated state is denoted by $x(t)$ and the desired reference is denoted by $r_d(t)$. The closest point with respect to the Euclidean distance in the dataset T to $(x(t), v_{NN}(t-1), r_d(t))$ is denoted by $(\bar{x}, \bar{v}_{pre}, \bar{r})$. Finally, $\bar{v}$ and $\bar{v}_{NN}$ refer to the output of NL-DRG with input

$[\bar{x}, \bar{v}_{pre}, \bar{r}]^T$ and the output of NN with the same input, respectively.

Theorem 7. *At timestep t, the difference between the output given by Σ with the states $x(t)$ and input $v_{NN}(t)$ (denoted by $y(t+1)$), and the output of the same system but with $\bar{x}$ and $\bar{v}$ (denoted by $\bar{y}$) is bounded by:*

$$\underbrace{\|y(t+1) - \bar{y}\|}_{d_{vio}} \leq \bar{L}\|x(t) - \bar{x}\| + \|g(x(t))\|e_{training} + L_{NN}\|g(x(t))\| \\ \|[\bar{x}, \bar{v}_{pre}, \bar{r}]^T - [x(t), v_{NN}(t-1), r(t)]^T\|$$

where $\bar{L} = L_h L_f + L_g\|\bar{v}\|$, and L_h, L_g, L_f, and L_{NN} represent the Lipschitz constant of h, g, f, and the NN, respectively.

Proof. Using the Lipschitz continuity of the quadcopter dynamics, the following inequalities holds:

$$\begin{aligned} d_{\text{vio}} &= \|h(f(\bar{x}) + g(\bar{x})\bar{v}) - h(f(x(t)) - g(x(t))v_{NN}(t)\| \\ &\leq L_h\|f(\bar{x}) - f(x(t))\| + \|g(\bar{x})\bar{v} - g(x(t))v_{NN}(t)\| \\ &\leq L_h L_f\|\bar{x} - x(t)\| + \|g(\bar{x})\bar{v} - g(x(t))v_{NN}(t)\| \\ &= L_h L_f\|x(t) - \bar{x}\| + \|g(\bar{x})\bar{v} - g(x(t))\bar{v} + g(x(t))\bar{v} - g(x(t))v_{NN}(t)\| \\ &\leq L_h L_f\|x(t) - \bar{x}\| + L_g\|\bar{v}\|\|x(t) - \bar{x}\| + \|g(x(t))\|\|v_{NN}(t) - \bar{v}\| \end{aligned} \tag{5.7}$$

Also, $\|v_{NN}(t) - \bar{v}\|$ can be bounded by:

$$\begin{aligned} \|v_{NN}(t) - \bar{v}\| &= \|v_{NN}(t) - \bar{v}_{NN} + \bar{v}_{NN} - \bar{v}\| \\ &\leq \|v_{NN}(t) - \bar{v}_{NN}\| + \|\bar{v}_{NN} - \bar{v}\| \end{aligned}$$

Recall that the training error is bounded by $e_{training}$, which implies that $\|\bar{v}_{NN} - \bar{v}\|$ is

bounded by $e_{training}$. Then, the above inequality can be rewritten as:

$$\begin{aligned} \|v_{NN}(t) - \bar{v}\| &\leq \|v_{NN}(t) - \bar{v}_{NN}\| + e_{training} \\ &\leq L_{NN}\|[\bar{x}, \bar{v}_{pre}, \bar{r}]^T - [x(t), v_{NN}(t-1), r(t)]^T\| + e_{training} \end{aligned} \tag{5.8}$$

By combining (5.7) and (5.8), the result follows □

From Theorem 7, it can be seen that if the training dataset is large and covers the whole state space (i.e., $\|[x(t) - \bar{x}\|$ is small), L_{NN} is small, and the maximum training error is small, then $\|y(t+1) - \bar{y}\|$ is also small, which implies that the maximal constraint-violation of $y(t+1)$ is small (recall that $\bar{y}$ must satisfy the constraint since the input $\bar{v}$ is computed by NL-DRG). This means that the distance between $v_{NN}(t)$ and $v_{aug}(t)$ will also be small. Note that L_{NN}, which can be quantified using [171], exists since the activation function (denoted by σ) of the NN is Lipschitz continuous and, from (2.23), the output of the NN is just a rotation and stretch of σ, which is also Lipschitz continuous. Additionally, from Theorem 7, it can be seen that, to ensure d_{vio} is small, the multiplication between L_{NN} and $\|[x(t), v_{NN}(t-1), r(t)]^T - [\bar{x}, \bar{v}_{pre}, \bar{r}]^T\|$ should also be small. Note that the distance between $[x(t), v_{NN}(t-1), r(t)]^T$ and $[\bar{x}, \bar{v}_{pre}, \bar{r}]^T$ is related to how well the NN is trained (i.e., the NN can or can not approximate the input-output map of NL-DRG accurately). More specifically, if the NN is not trained well (i.e., $[x(t), v_{NN}(t-1), r(t)]^T$ is far from from $[\bar{x}, \bar{v}_{pre}, \bar{r}]^T$), then, L_{NN} is required to be small so that large difference in the input will not cause large difference in the output. However, small L_{NN} may lead to a slow system response.

Chapter 6

Constraint Management for Quadcopter Drones

With the increasing utilization of quadcopter drones, more and more studies have been focused on the constraint management of quadcopter (as explained in Section 1.2.4). Motivated by the necessity of constraints in the quadcopter control design, below, the proposed nonlinear RG-based solutions (NL-DRG, M-RG, and NN-DRG) will be implemented on the Crazyflie 2.0 [62] (as reviewed in Section 2.6.3). Both simulation and experimental results will be presented.

6.1 Simulation results of Proposed Methods

In section, the simulation results of NL-DRG, M-RG, and NN-DRG on the model of Crazyflie will be provided. Recall from Section 2.6.3, the constraints are imposed

on the roll angle: $\phi \in [-5^\circ, 5^\circ]$; pitch angle: $\theta \in [-5^\circ, 5^\circ]$; and yaw rate: $\dot{\psi} \in [-10^\circ/sec, 10^\circ/sec]$. Also, for a better demonstration, the simulation results of NL-RG (explained in Section 2.2.4) will also be presented.

By implementing Algorithm 2 with $J_1 = J_2 = 5000$ and $N = 1000$ (corresponding to 10 second simulation), we obtain $\widetilde{W} = [0.65^\circ, 0.65^\circ, 5.3^\circ/sec]^T$, $\bar{d}_\phi = 0.95^\circ$, $\bar{d}_\theta = 0.95^\circ$, and $\bar{d}_{\dot{\psi}} = 6.7^\circ/sec$. The simulation results of NL-DRG are shown in Figure 6.2. It can be seen that the outputs satisfy the constraints as required. We will provide details on the computation times after we show simulation results for NN-DRG, which is provided next.

6.1.1 Nonlinear Reference Governor (NL-RG) on Quadcopter

In this section, the NL-RG method, as reviewed in Section 2.2.4, is implemented on the nonlinear quadcopter dynamics. To ensure the constraint satisfaction of the nonlinear system affected by disturbance and noise, the constraint set as shown in Figure 2.1 is modified from $y \in [-s, s]$ to $y \in [-s + \bar{W}, s - \bar{W}]$. The simulation results of NL-RG on the quadcopter dynamics are duplicated in Figure 6.1. It can be seen that, when $t \in [1, 3]$, the roll angle and pitch angle can not track the setpoints even no constraint violation is detected. To explain the root cause of this behavior, note that a single κ is used in NL-RG scheme to govern ϕ, θ, and $\dot{\psi}$ simultaneously. Also note, from Figure 6.1, $\dot{\psi}$ already reaches the constraint when $t \in [1, 3]$, implying $\kappa = 0$. As a result, the governed references given by NL-RG for the roll angle and pitch angle could not arise above 00, which lead $\phi = \theta = 0$.

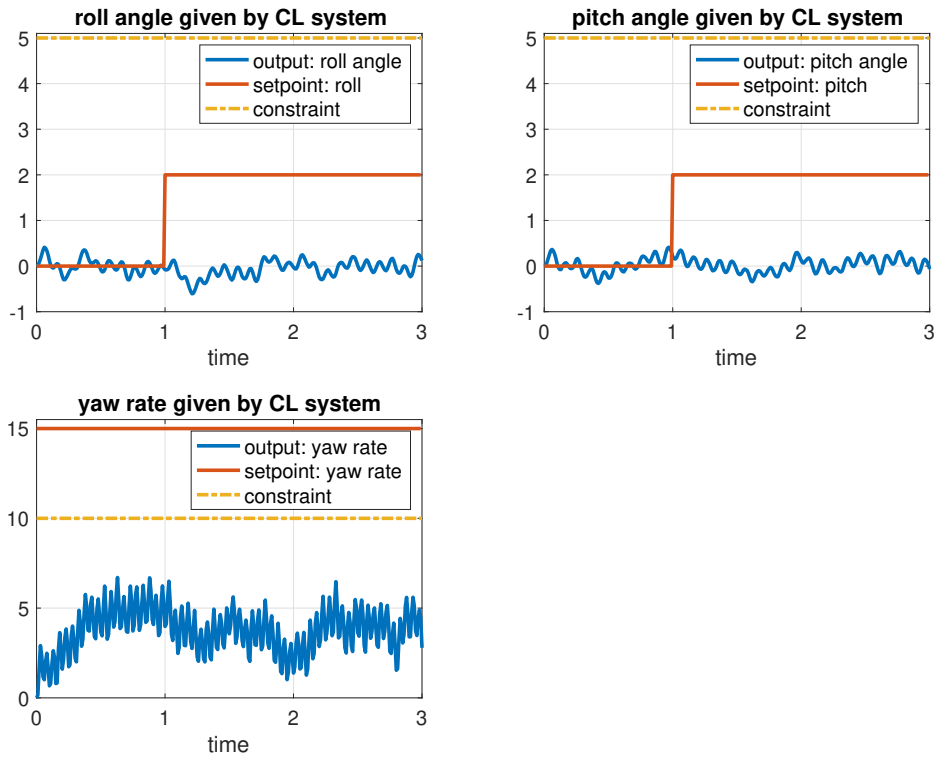

Figure 6.1: Simulation results of NL-RG on the nonlinear quadcopter dynamics.

6.1.2 Nonlinear Decoupled Reference Governor (NL-DRG) on Quadcopter

The simulation results of NL-DRG on the quadcopter dynamics are shown in Figure 6.2. The desired pitch angle is set to be 0. As the figure shows, the constraints are satisfied, as required. Note that one advantage of NL-DRG compared to NN-DRG (as will be shown later) is that, thanks to the fact that v given by NL-DRG is forced to be monotonic increasing or decreasing (see (2.13)) for a constant setpoint, thus, v will not respond to the noise and disturbance (in other words, the sensor noise

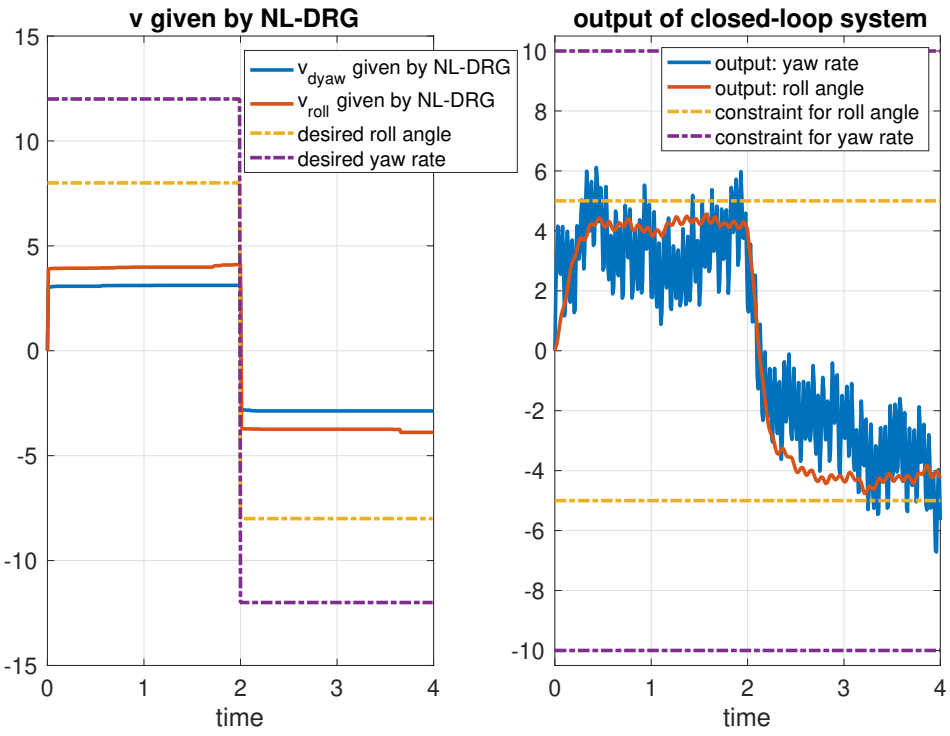

Figure 6.2: Simulation response of NL-DRG

and/or disturbance will not transmit to v), which is not the case for NN-DRG. More specifically, as (2.23) shows, the output and input of NN are connected via the weight matrices (w_1 and w_2) and bias (b_1 and b_2). Thus, depending on the training results, the output of NN may be heavily or slightly affected by the disturbance in the input.

Below, the simulation of M-RG on the quadcopter dynamics is presented to address the computational issue of NL-DRG

6.1.3 Modified Reference Governor (M-RG) on Quadcopter

In this section, the simulation results of M-RG on the quadcopter dynamics are delivered. Recall from Section 5.2, first, the linearization error (denoted by d^e) between the linearized model (Eqs. (2.34) and (2.35)) and nonlinear model (Eqs. (2.30) and (2.32)) needs to be quantified offline. Denoted the linearized quadcopter dynamics as Σ_{lin}. As Algorithm 4 shows, the d^e is quantified by simulating Σ_{lin} and Σ with same time-series setpints and numerically finding the maximum difference between the output given by Σ_{lin} and the output given by Σ. From simulation, with $J1 = 5000$ and $N = 1000$, the maximum linearization error on roll angle, denoted by d^e_ϕ, equals to $0.8°$, the maximum linearization error on pitch angle (d^e_θ), equal to $0.8°$, and the maximum linearization error on yaw rate ($d^e_{\dot{\psi}}$), equivalent to $0.9°$.

In real-time implementation, for each SRG, the constraint set is shrunk to take the worst-case realization of linearization error, coupling behavior, as well as disturbance and noise into consideration. The simulation results of M-RG on quadcopter modeling are illustrated in Figure 6.3. It can be seen, from Figure 6.3, the constraints are satisfied, as expected. Meanwhile, the performance on yaw rate is more conservative than that given by NL-DRG (as shown in Figure 6.2) since M-RG shrinks the constraints further than that for NL-DRG.

Below, the simulation results of NN-DRG, which has an improved performance than M-RG while maintaining a lower computational cost than NL-DRG, on the quadcopter dynamics will be delivered.

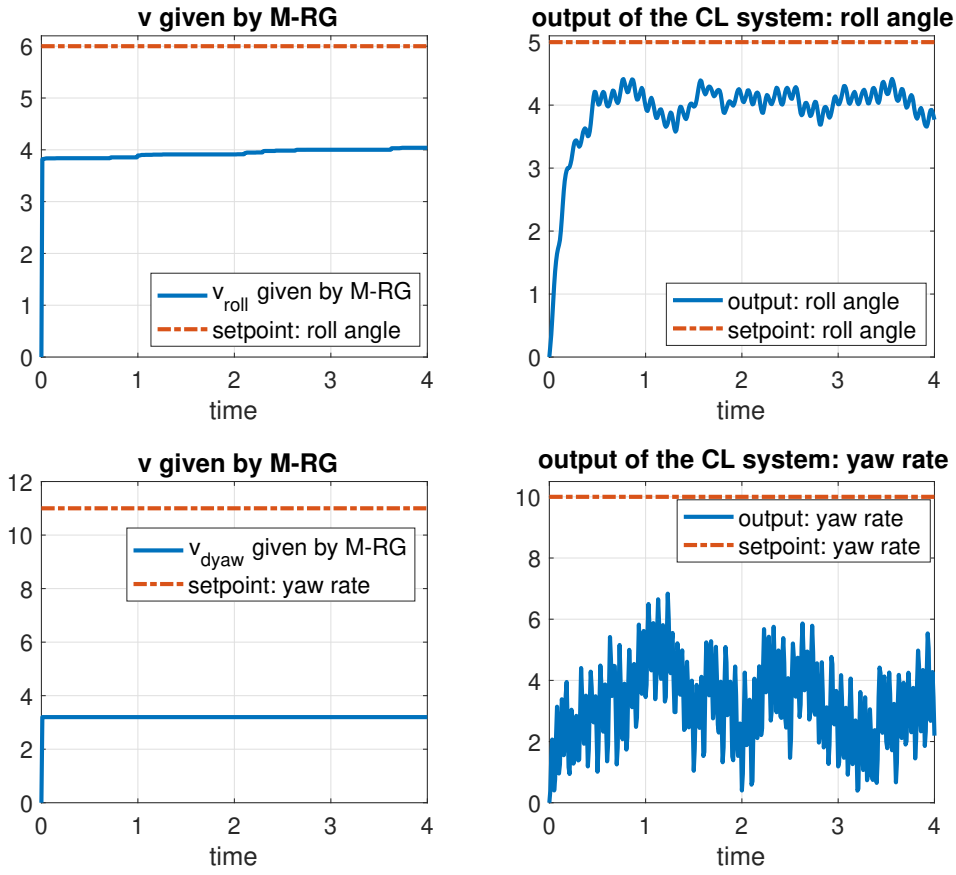

Figure 6.3: Simulation results of M-RG on the quadcopter dynamics

6.1.4 Neural Network Decoupled Reference Governor (NN-DRG) on Quadcopter

To implement NN-DRG, first, 100000 input-output pairs (where the input is $[x(t), v(t-1), r(t)]$ and output is $[v(t)]$) are collected off-line by simulating the NL-DRG on the Σ with randomly generated time-series inputs as described previously. Second, a simple-structure feedfoward Neural Network (NN) with 1 hidden layer is trained by

Table 6.1: Performance of the Neural Network Model

# of hidden neurons	Mean square error (in degree^2)	Largest training error (in degree)	training Epochs
2	0.0498	6.5776	33
5	0.0054	6.4973	269
10	2.842×10^{-3}	6.8755	165
30	1.233×10^{-3}	6.9729	409

the backpropagation method with the Levenberg-Marquardt, which needs fewer training iterations. The activation function is chosen to be the Sigmoid in the hidden layer and linear in the output layer. The motivation behind using a single hidden layer is that, according to the NN universal approximation theory, one layer is sufficient to represent the input-output mapping of NL-DRG, because its outputs are continuous functions of its inputs. The training errors of NNs with different number of neurons are listed in Table 6.1. It can be seen that, with the increasing of the number of neurons, the MSE is decreasing while the maximum training error maintains almost the same. Note that, by modifying the architecture of the NN model (e.g., using recurrent NN), the maximum training error may be reduced. Further investigation on this topic will be studied in our future work. For the illustration purpose, feedforward NN with 10 neurons in the hidden layer is sufficient to show the performance of the proposed methods because, as shown in Figure 6.4 and Figure 6.5, most training samples have training error less than 1° and the system performance of NN-DRG on the closed-loop Crazyflie dynamics with 10 and 30 neurons perform better than that with 2 and 5 neurons. The setpoints in Figure 6.5 are $\phi_d = 10^\circ$, $\theta_d = 0$, and $\dot{\psi}_d = 20^\circ/sec$, and the constraints are $\phi, \theta \in [-5^\circ, 5^\circ]$ and $\dot{\psi} \in [-10^\circ/sec, 10^\circ/sec]$.

Additionally, Figure 6.6 visually shows, using a shadow region, the cross-sections

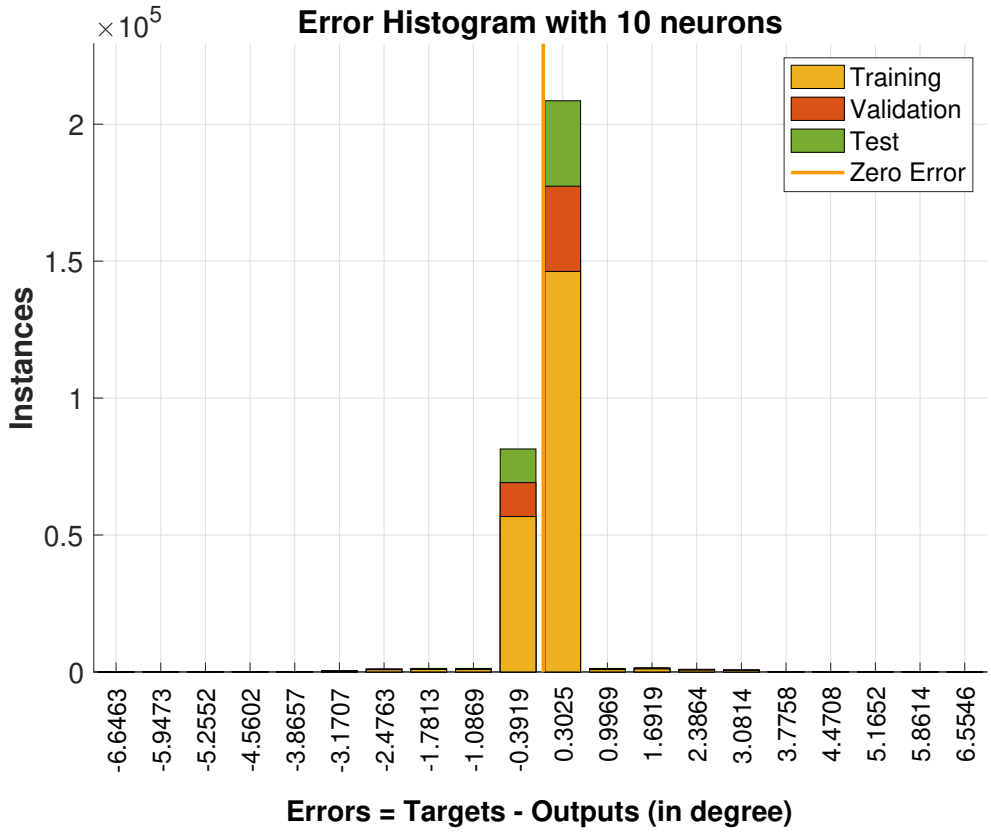

Figure 6.4: This figure represent the histograms, where the x-axis refers to the training error and the y-axis represent the number of instances where the training error lies in the corresponding range.

of the maximal admissible set along the p (the pitch rate as introduced in (2.30)), ϕ (the roll), and θ (the pitch), with all other states set to 0. These sets indicate the regions in the state-input planes where constraints will be satisfied if the inputs are held constant. Since the quadcopter system is nonlinear, these sets are numerically obtained by simulating Σ_{disfree} with the NL-DRG with many randomly generated references. As comparison, the approximation of this set, as obtained numerically through similar simulations but with the trained NN, is shown using dots. It can

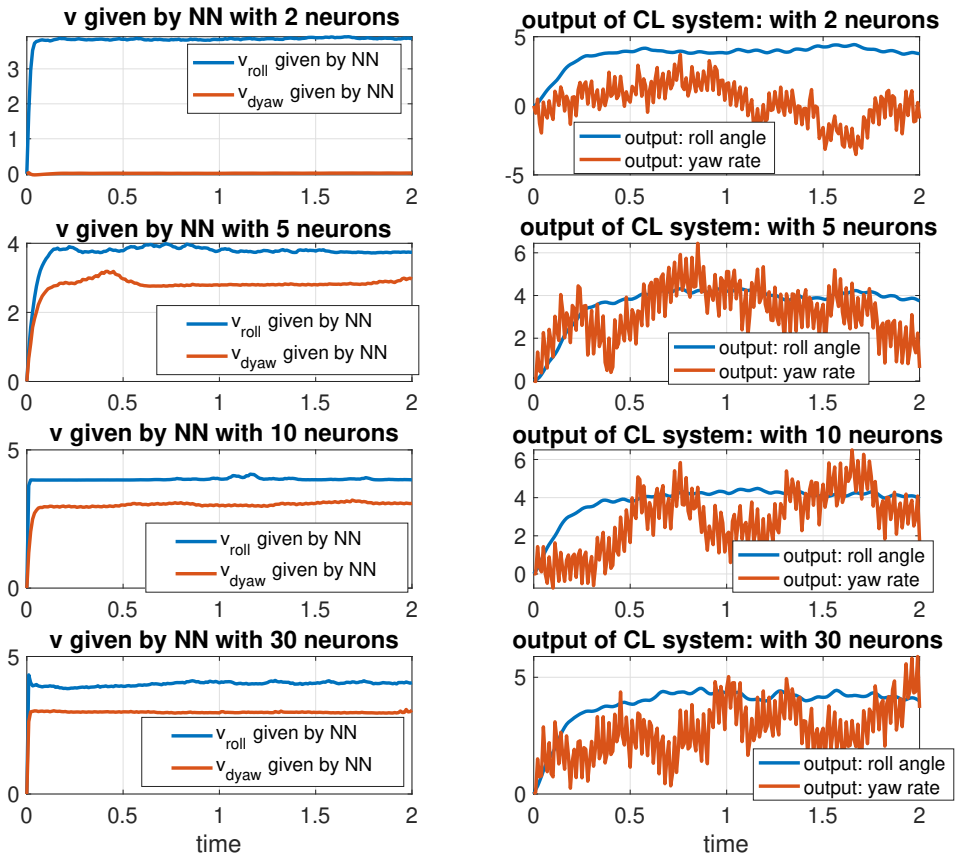

Figure 6.5: Comparison of neurons number

be seen that the NN can accurately approximate the mapping from v to ϕ and θ. However, the mapping from v to angular velocity p is not accurately captured.

Remark 8. *An obvious difference between the collection of the training data based on Σ and $\Sigma_{disfree}$ is that the dataset constructed based on Σ covers states with noise and disturbance. For simplicity, in this discussion only, let us call the NN-DRG as "std NN-DRG" for the case where the NN is trained based on $\Sigma_{disfree}$ and "robust NN-DRG" for the case where the NN is trained based on Σ. With such minor modification to the dataset, the system performance varies massively. The comparison between the*

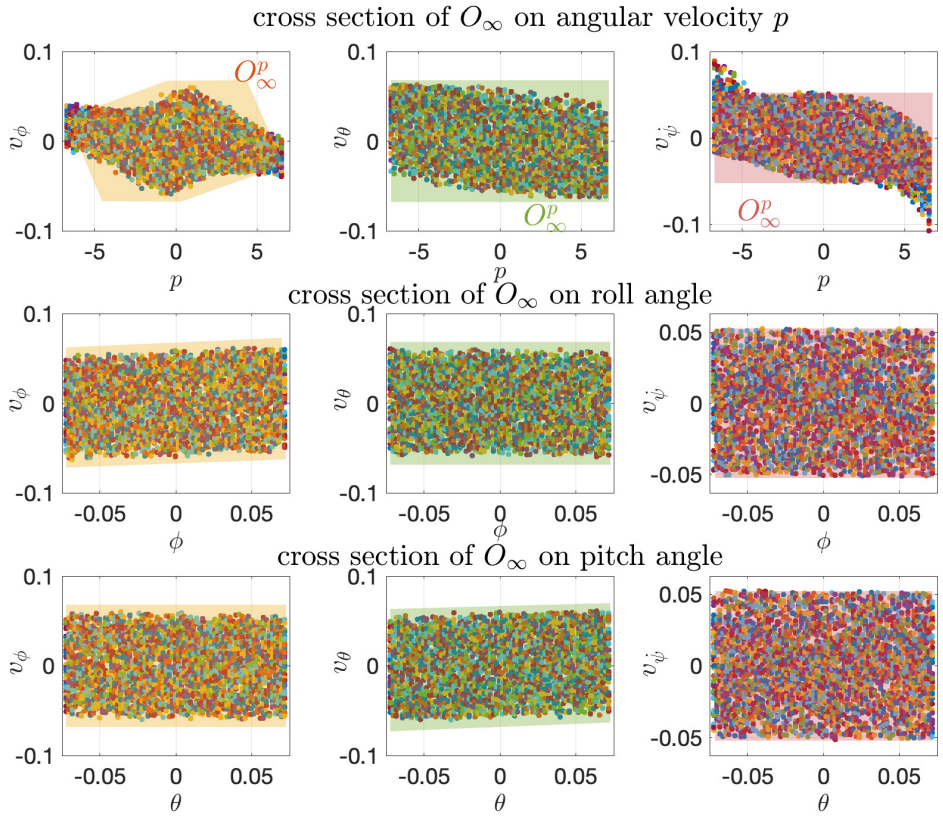

Figure 6.6: The cross sections of the MAS

simulation results of robust NN-DRG and std NN-DRG is shown in Figure 6.7 (the desired roll angle and yaw rate are 10° *and* $20^\circ/sec$*, respectively). It can be seen that the* v *given by robust NN-DRG remains almost a constant while the* v *given by std NN-DRG oscillates widely, which is caused by the noise and disturbance in the input of the NN. The reason for this is that the dataset for robust NN-DRG is more similar to the input-output pairs that are observed during the real-time operation of the Crazyflie (with disturbance and noise). Moreover, because of the disturbance and noise, the training dataset for the robust NN-DRG contains more regions of the state-space where the Crazyflie might actually operate.*

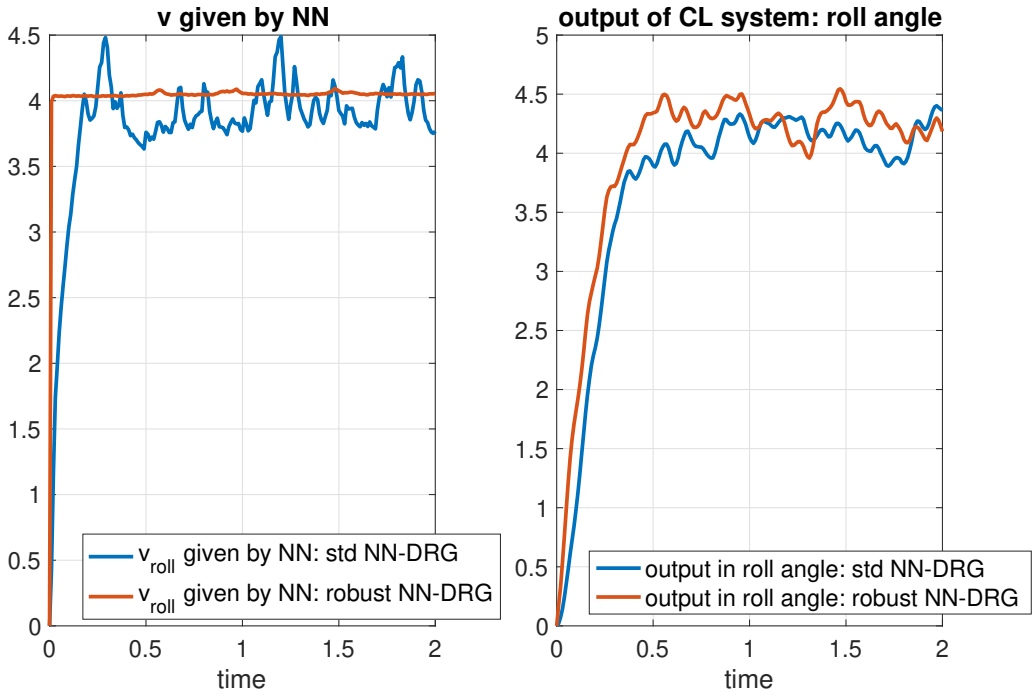

Figure 6.7: The comparison on the simulation response of std NN-DRG and robust NN-DRG

Next, we show the simulation results of the entire NN-DRG scheme (i.e., the NN augmented with the QP in (5.6)) in Figure 6.8. In this simulation, we set Q to be the identity matrix (i.e., I_3). The impact of Q on the system performance is explained in Remark 9. The desired roll angle and yaw rate are 15° and 20°/*sec*, respectively. It can be seen that, from Figure 6.8, the constraints are satisfied in the outputs. Note that $v_{\dot{\psi}}$ drops down to nearly 4.8° when t is around 1.5sec while the output on yaw rate is still far away from the constraint. The reason for this can be explained as follows. First, recall that to ensure constraint satisfaction for the nonlinear Crazyflie system with disturbance and noise, the constraint set is shrunk by $\bar{d}_{\dot{\psi}}$, which represents the effects of the worst-case coupling behavior, disturbance, and sensor noise. This is the reason why the output appears to be far from the constraint (i.e., the “conservatism"

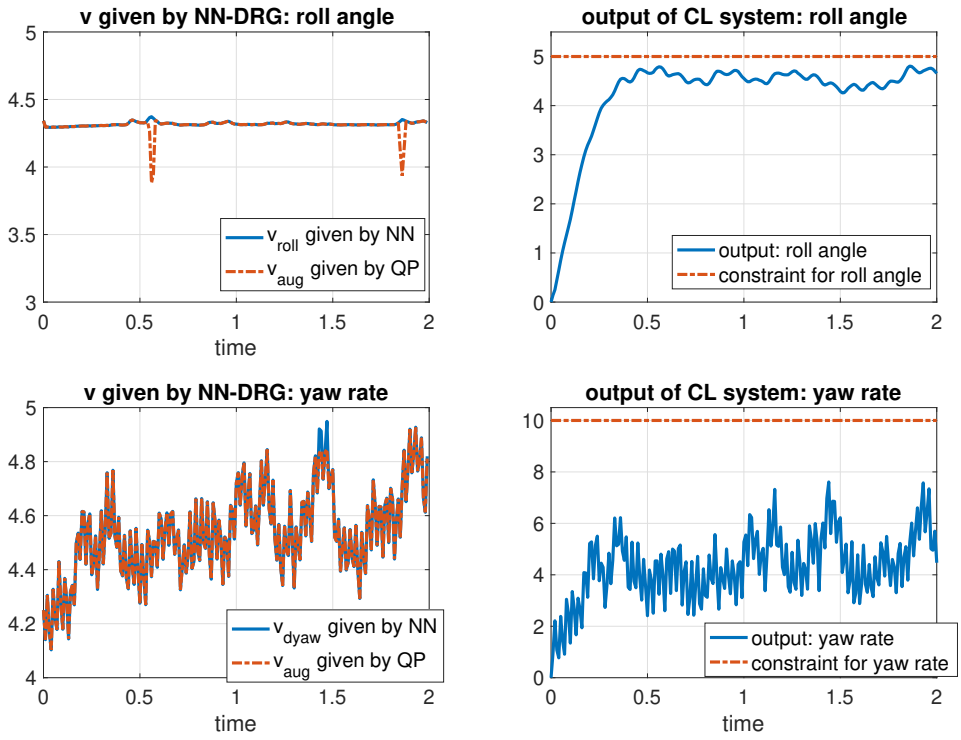

Figure 6.8: The simulation results of NN-DRG on the closed-loop nonlinear Crazyflie model. The oscillations are due to the stochastic sensor noise and the deterministic sinusoidal disturbance introduced in the simulation.

seen in the response). The reason for the drop around 1.5sec is that the QP in the NN-DRG formulation detects a possible violation of the tightened constraint (i.e., $s - \bar{d}$ shown in Algorithm 3), so it lowers its computed output to avoid what it deems to be a potential constraint violation on the yaw rate.

Remark 9. *The diagonal matrix, Q, is used in the optimization problem shown in* (5.6) *to penalize the deviations from the NN output in the roll, pitch, and yaw rate channels. The larger the Q_{ii} is, the more the corresponding angle is prioritized over the other two angles/angle rates to be as close as possible to the output given the NN.*

Table 6.2: Comparison of the computation time among the proposed nonlinear RG-based methods

	NL-RG	**NL-DRG**	**M-RG**	**NN-DRG**
avg (sec)	0.0203	0.0717	2.24×10^{-5}	8.79×10^{-3}
max (sec)	0.0518	0.12	6.36×10^{-5}	0.0261

6.1.5 Computational Comparison

In this section, we will provide a computational comparison between NL-RG, M-RG, NL-DRG, and NN-DRG on the discretized nonlinear quadcopter dynamics with $T_s = 0.01$ (the discretization is achieved by using the Forward Euler method). The references used in this comparison (ϕ_d, θ_d, and $\dot{\psi}_d$) are shown in Figure 6.3. Recall that M-RG requires the solution to three LPs at each time step, while NL-RG and NL-DRG require simulating the plant over a finite time horizon (which is chosen to be 10sec in this thesis) and implementing the bisectional search algorithm to find the solution. Note that instead of using generic LP solvers to solve M-RG, we use the algorithm presented in [149] to solve them. The NN-DRG requires the solution to one quadratic program (QP) at each time step. The QP is solved using Multi-Parametric Quadratic Programming (MPQP), which is introduced in [75].

We simulated the nonlinear Crazyflie dynamics (no disturbance involved) with all four methods: NL-RG as well as the proposed M-RG, NL-DRG, and NN-DRG. All simulations were performed for 1000 time steps in Matlab on an Apple Macbook Pro with M1 chip and 8 GB memory. In order to eliminate the effects of background processes running on the computer, each of the above experiments were run 10 times and the averages were calculated. We calculate the per-time step averages and maximums of each of the three methods. The results are shown in Table 6.2. As can be

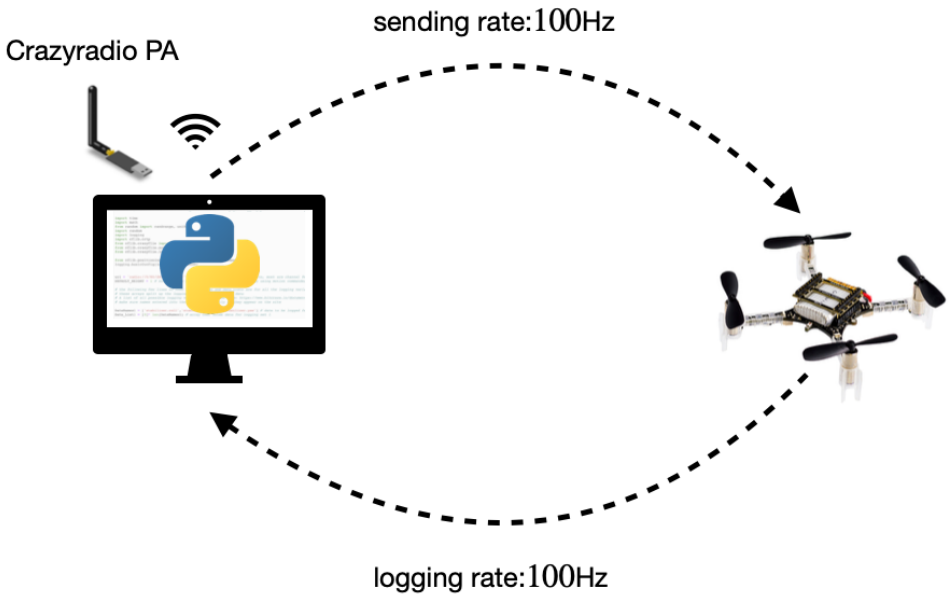

Figure 6.9: Communication between client and Crazyflie

seen, M-RG runs two orders of magnitude faster than NN-DRG because NN-DRG needs to solve a QP problem. The NN-DRG is almost 2 times faster than NL-RG since NL-RG needs to simulate the plant over 10sec and implement the bisectional search algorithm. The NL-RG terminates almost 3 times faster than NL-DRG since in NL-DRG scheme, at every time step, three NL-RGs are needed to be implemented.

6.2 Experimental Results of Proposed Methods

In this section, the simulation results of NL-DRG, M-RG, and NN-DRG on the real Crazyflie are presented. At first, a brief background on how to send commands

to Crazyflie will be introduced. The communication between the client (desktop computer) and the Crazyflie is shown in Figure 6.9. More specifically, after running NL-DRG. M-RG, or NN-DRG on a desktop computer using the Python programming language, the computed setpoints are transmitted to the Crazyflie via a USB radio dongle. The communication rate is 100Hz (i.e., commands are sent every 10 ms) to ensure a smooth flying. The Crazyflie, then, will communicate its states back to the desktop computer. The states will be logged and also used for the next computation. The logging rate is also 100Hz to ensure that the proposed methods have the accurate information on the state of the Crazyflie. Additionally, to have a better performance, the states of the Crazyflie are first mildly filtered using FIR filters before using them in the proposed algorithms. More specifically, a 1^{st} order FIR filter (averaging among two timesteps) is used on p, q, ϕ, and θ. The r is filtered using a 2^{nd} order FIR filter (averaging among three timesteps).

6.2.1 Experimental Results of NL-DRG

The experimental results of NL-DRG on the real Crazyflie are shown in Figure 6.10. The setpoints are $\phi_d = 10°$, $\theta_d = 0°$, and $\dot{\psi}_d = 15°/sec.$ As can be seen, the constraints are satisfied. However, due to the very large computation time of NL-DRG, the sending and logging rate of Crazyflie had to be changed from 0.01sec to 0.2sec, which is why the plots look jagged. This is not desirable as it leads to poor performance. This shortcoming is addressed by the M-RG, whose experimental results are provided next.

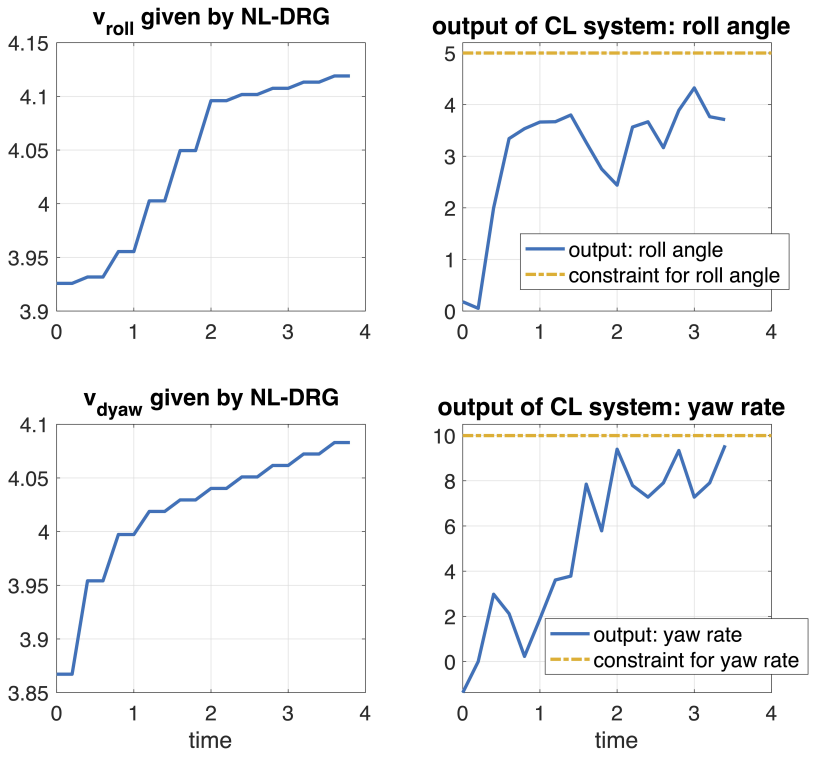

Figure 6.10: The response of NL-DRG on the real Crazyflie. The sending and logging rate is $0.2sec$

6.2.2 Experimental Results of M-RG

The experimental results of M-RG on the real Crazyflie is shown in Figure 6.11. The desired stepoints are: $\phi_d = 6°$ and $\dot{\psi}_d = 12°/sec$. It can be seen that the outputs satisfy the constraints, as expected.

Next, we will show the experimental results of NN-DRG to improve the performance of M-RG while maintaining a low computational cost.

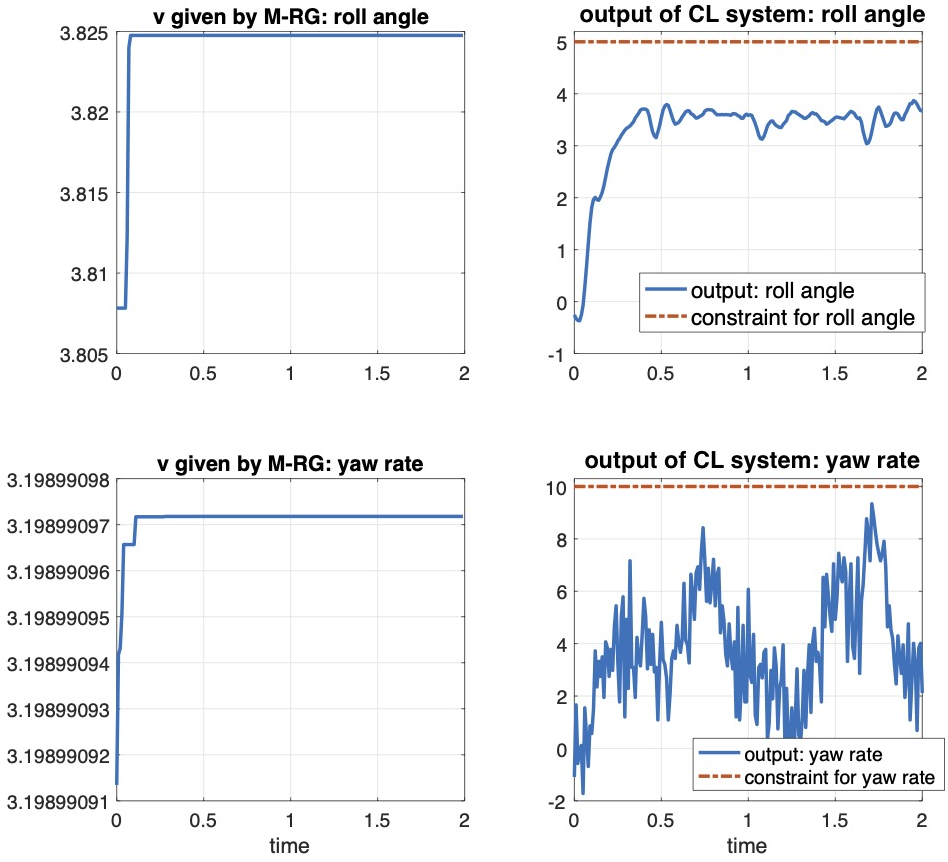

Figure 6.11: The experiment results of NN-DRG on the Crazyflie

6.2.3 Experimental results of NN-DRG

For a more clear illustration, first, the experimental results of only NN (i.e, without QP implementation) on the real Crazyflie are presented. From Figure 6.12, it can be seen that the output for yaw rate violates the constraints when $t \in [0.5, 1.5]$ due to the training error and/or incomplete data collection.

Next, we will show that the experimental results of NN-DRG (i.e., the NN followed by the QP). In this experiment, the QP is solved using python-embedded CVX

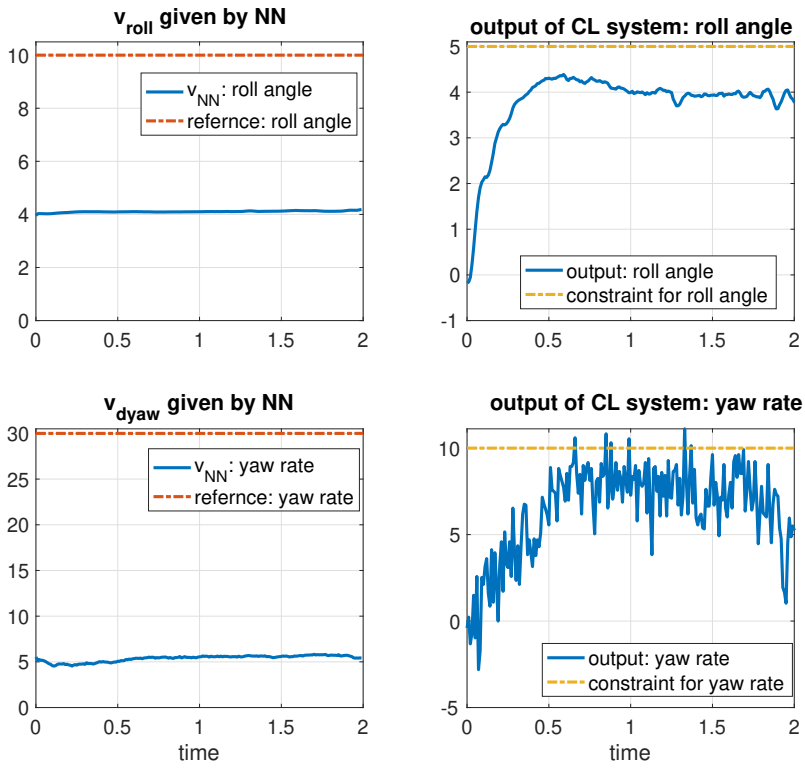

Figure 6.12: The experiment results of NN on the Crazyflie

toolbox, which has average computation time around 0.02s. To ensure that the NN-DRG works as expected, the command sending rate and logging rate are modified from 0.01s to 0.02s.

The experimental results of NN-DRG on the Crazyflie is shown in Figure 6.13. It can be seen that the outputs satisfy the constraints, as required. Also, to avoid constraint violation, v_{aug} for the yaw rate drops down from 5.7° to 5.2° when $t \in [1, 1.5]$. Note that, as Theorem 7 shows, if larger and more comprehensive training dataset is used, the distance between $v_{NN}(t)$ and $v_{aug}(t)$ will decrease.

Remark 10. *The results presented in this thesis (i.e., NL-DRG, M-RG, and NN-*

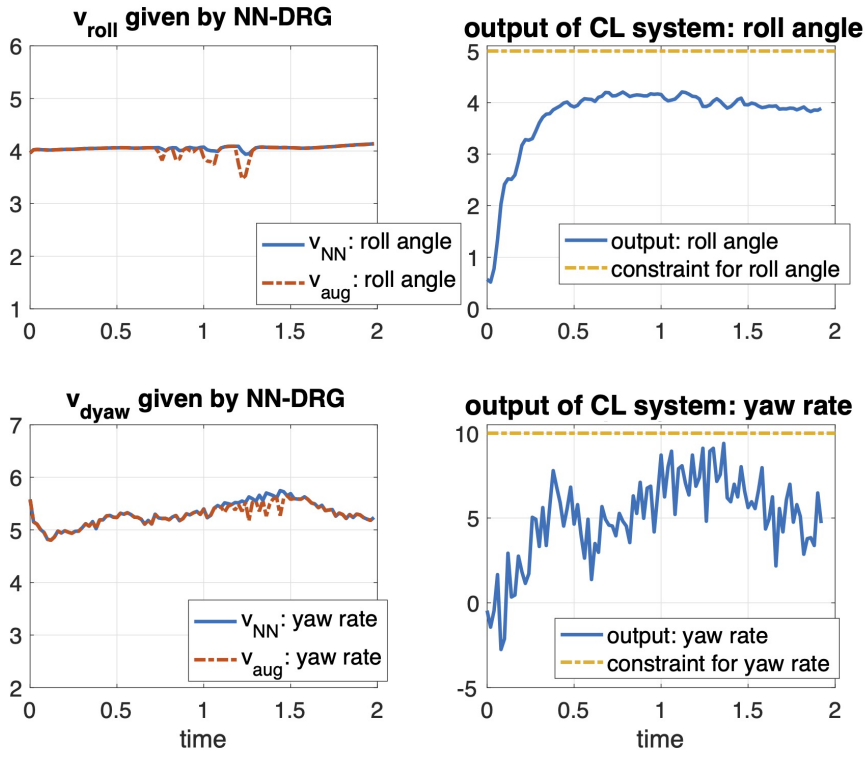

Figure 6.13: Experiment results of NN-DRG on the real Crazyflie

DRG) can be applied to constraint management of nonlinear MIMO systems that can be described by equations of the form (5.1)*. The shortcoming is that the response may be conservative in some cases. Generally, if the interactions between the different channels in the nonlinear system are mild, the sensor noise and disturbances are relatively small, and a prioritization order can be defined between the channels, then,* $\bar{d}$ *would be small and, thus, conservatism would be small. Otherwise, the NL-DRG might lead to a overly conservative response. In the most extreme case, if* $\bar{d}$ *is larger than the constraint* s*, then, no admissible* v *can be obtained, i.e.,* v *will be calculated as* 0 *at every time step.*

Chapter 7

Conclusions and Future Works

This dissertation focused on the theoretical extensions and practical applications of RGs. More specifically, the RG theory has been extended to different types of systems, namely MIMO systems, systems incorporated with preview information, and nonlinear systems. The schemes presented in this dissertation were supported by systematical analysis. As for the application aspect of RG, the real quadcopter done, namely Crazyflie 2.0, was implemented as the platform to exam our proposed nonlinear RG-based solutions. The main developments and results are summarized below for each of the above developments.

7.1 Decoupled Reference Governor

A method for constraint management of coupled linear MIMO systems was studied in Chapter 3. The method is referred to as the Decoupled Reference Governor (DRG) and is based on decoupling the input-output dynamics, followed by application of scalar reference governors to each decoupled channel. This idea was first developed

in my previous work [26], namely DRG-tf. This work improved the design of DRG-tf, analyzed the transient performance, and studied the observer design for DRG-tf. Also, the DRG formula was extended to state space decoupling method, namely DRG-ss. Unknown disturbances and parametric uncertainties for DRG scheme were addressed. Finally, DRG was extended to non-square MIMO systems.

7.2 Preview Reference Governor

In Chapter 4, a reference governor-based method for constraint management of linear systems was proposed. The method is referred to as Preview Reference Governor (PRG) and can systematically account for the preview information of reference and disturbance signals. The method is based on lifting the input of the system to a space of higher dimension and designing maximal admissible sets based on the system with lifted input. We showed a limitation of PRG and proposed an alternative method, which we referred to as Multi-horizon PRG (multi-N PRG), to overcome the limitation. Disturbance previews, parametric uncertainties, and inaccurate preview reference information were also addressed. We also showed that the PRG for multi-input systems using the lifting idea (i.e., first solution in Section 4.4) might cause conservative response. Thus, we proposed another method, which combines the Decoupled Reference Governor scheme and PRG, to overcome this limitation.

7.3 Nonlinear Reference Governor on MIMO systems

Three reference governor-based solutions to enforce constraints on nonlinear MIMO systems were presented in Chapter 5. The first solution, referred to as the Nonlinear Decoupled Reference Governor, was proposed to address the limitation of NL-RG (e.g. overly conservative on MIMO systems), an existing RG-based method applicable to nonlinear systems. In NL-DRG scheme, the sequential NL-RGs are computed, where the constraint for each NL-RG is shrunk to account the worst-case coupling dynamics. As a result, it performs better than NL-RG, but at the expense of a larger computational cost. The second approach, namely M-RG, was proposed to address computational issue of NL-DRG. The M-RG scheme is based on the standard SRG, which can guarantees constraint satisfaction of the nonlinear systems but leads to a more conservative response as compared to NL-DRG. To lower the computation time of NL-DRG while maintaining a superior performance than M-RG, the third method, namely NN-DRG, was presented, where a well-trained Neural Network is used to replace the functionality of NL-DRG and a QP is solved to ensure that the outputs in the next time step satisfy the constraints. The limitation of NN-DRG is that, since only one-timestep prediction of the output is involved in the QP design, a large drop in the governed command (i.e., $v(t)$ sent to the closed-loop quadcopter dynamics) may be obtained, which can be alleviated by training the NN using a larger and comprehensive dataset, and choosing a NN with a smaller Lipschitz constant and training error.

7.4 Constraint Management for Quadcopter Drones

Chapter 6 presented the simulation and experimental results of NL-DRG, M-RG, and NN-DRG on a real quadcopter drones, namely Crazyflie 2.0. From simulation results, it can be seen that NL-DRG performed better than M-RG and NN-DRG. Moreover, M-RG led to a more conservative response than NL-DRG since the constraints are shrunk further to prevent the constraint violation. To address the shortcoming of M-RG while maintaining low computational effort, NN-DRG was implemented. The NN-DRG scheme guaranteed constraint satisfaction. However, due to the fact that only one-timestep prediction of the output is involved in the QP, NN-DRG may cause the commanded reference to drop upon detection of a constraint violation. We showed that such a drop can be made small by training the NN using a larger and more comprehensive dataset and reduced training error.

From experimental results, due to the large computational time of NL-DRG, it may lead to a undesirable system performance. Meanwhile, M-RG can guarantee constraint satisfaction and has a relatively small computational effort, which inevitably introduces some degree of conservatism. Finally, NN-DRG improves the performance of M-RG and has a smaller computational cost compared to NL-DRG.

7.5 Future Works

There are still numerous open questions in the constraint management, such as how to recover from constraint violation and how to effectively handle the internal and external disturbance. As for the practical applications, the RG still lacks an impact on the industry and, thus, the implementation of RG-based methods on real systems should be explored broadly. For each of the RG schemes proposed in this dissertation, the future works are listed below:

- DRG: Future work will explore modifications to DRG to ensure that the inputs to the closed-loop system (i.e., u in Figure 4.9) remain below the references (i.e., r). We will also explore DRG formulations that have the ability to recover from constraint violation, should unknown disturbances or observer errors push the system outside of the maximal admissible sets.

- PRG: Future work will explore preview control in the context of Vector Reference Governors, as well as finding the optimal set of λs that gives the best performance in our robust PRG formulation. We will also investigate the extension of PRG to nonlinear systems.

- Nonlinear RG-based solutions: Future work will remove the QP from the NN-DRG formulation and quantify the probability that the NN will satisfy the constraints despite training errors to further simplify the computations. I will also explore other Neural Network models on the approximation of NL-DRG to improve the performance of NN-DRG. The extensions of the proposed methods to systems with parametric uncertainties will be studied. Finally, the stability

and convergence analysis of NN-DRG will also be explored.

Bibliography

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

[2] 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.

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

[4] 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, 1991.

[5] Ayad Al-Mahturi and Herman Wahid. Optimal tuning of linear quadratic regulator controller using a particle swarm optimization for two-rotor aerodynamical system. *Int. J. Electr. Comput. Energ. Electron. Commun. Eng*, 11(2):184–190, 2017.

[6] Muhammad Ali Masood Cheema, John Edward Fletcher, Dan Xiao, and Muhammad Faz Rahman. A linear quadratic regulator-based optimal direct thrust force control of linear permanent-magnet synchronous motor. *IEEE Transactions on Industrial Electronics*, 63(5):2722–2733, 2016.

[7] Prachi Barsaiyan and Shubhi Purwar. Comparison of state feedback controller design methods for mimo systems. In *2010 International Conference on Power, Control and Embedded Systems*, pages 1–6. IEEE, 2010.

[8] Luc Le Tien, Alin Albu Schaffer, and Gerd Hirzinger. Mimo state feedback controller for a flexible joint robot with strong joint coupling. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3824–3830. IEEE, 2007.

[9] L Merazka, F Zouari, and A Boulkroune. Fuzzy state-feedback control of uncertain nonlinear mimo systems. In *2017 6th International Conference on Systems and Control (ICSC)*, pages 103–108. IEEE, 2017.

[10] Man Zhihong and Xing Huo Yu. Terminal sliding mode control of mimo linear systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 44(11):1065–1070, 1997.

[11] Man Zhihong, Andrew P Paplinski, and Hong Ren Wu. A robust mimo terminal sliding mode control scheme for rigid robotic manipulators. *IEEE transactions on automatic control*, 39(12):2464–2469, 1994.

[12] Angelika Bunse-Gerstner, Dorota Kubalińska, Georg Vossen, and Diane Wilczek. h2-norm optimal model reduction for large scale discrete dynamical mimo systems. *Journal of computational and applied mathematics*, 233(5):1202–1216, 2010.

[13] Paul Van Dooren, Kyle A Gallivan, and P-A Absil. H2-optimal model reduction of mimo systems. *Applied Mathematics Letters*, 21(12):1267–1273, 2008.

[14] P. Falb and W. Wolovich. Decoupling in the design and synthesis of multivariable control systems. *IEEE Transactions on Automatic Control*, 12(6):651–659, 1967.

[15] Elmer G Gilbert. The decoupling of multivariable systems by state feedback. *SIAM Journal on Control*, 7(1):50–63, 1969.

[16] W Murray Wonham and A Stephen Morse. Decoupling and pole assignment in linear multivariable systems: a geometric approach. *SIAM Journal on Control*, 8(1):1–18, 1970.

[17] Lu Liu, Siyuan Tian, Dingyu Xue, Tao Zhang, YangQuan Chen, and Shuo Zhang. A review of industrial mimo decoupling control. *International Journal of Control, Automation and Systems*, 17(5):1246–1254, 2019.

[18] Gaurang Shah and Sebastian Engell. Tuning mpc for desired closed-loop performance for mimo systems. In *Proceedings of the 2011 American Control Conference*, pages 4404–4409. IEEE, 2011.

[19] 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.

[20] 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.

[21] JS McDonald and JB Pearson. l_1-optimal control of multivariable systems with output norm constraints. *Automatica*, 27(2):317–329, 1991.

[22] Jian-Xin Xu and Xu Jin. State-constrained iterative learning control for a class of mimo systems. *IEEE Transactions on Automatic Control*, 58(5):1322–1327, 2012.

[23] Xu Jin. Adaptive fault tolerant control for a class of input and state constrained mimo nonlinear systems. *International Journal of Robust and Nonlinear Control*, 26(2):286–302, 2016.

[24] P.O.M. Scokaert and J.B. Rawlings. Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169, 1998.

[25] 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, 1995.

[26] Yudan Liu. Decoupled reference governors for multi-input multi-output systems. In *Graduate College Dissertations and Theses*, page 970, 2018.

[27] Ahmet Arda Ozdemir, Peter Seiler, and Gary J Balas. Design tradeoffs of wind turbine preview control. *IEEE Transactions on Control Systems Technology*, 21(4):1143–1154, 2013.

[28] Arne Koerber and Rudibert King. Combined feedback–feedforward control of wind turbines using state-constrained model predictive control. *IEEE Transactions on Control Systems Technology*, 21(4):1117–1128, 2013.

[29] C. Gohrle, A. Schindler, A. Wagner, and O. Sawodny. Design and vehicle implementation of preview active suspension controllers. *IEEE Transactions on Control Systems Technology*, 22(3):1135–1142, May 2014.

[30] Shuhei Shimmyo, Tomoya Sato, and Kouhei Ohnishi. Biped walking pattern generation by using preview control based on three-mass model. *IEEE transactions on industrial electronics*, 60(11):5137–5147, 2012.

[31] Kiyotsugu Takaba. A tutorial on preview control systems. In *SICE 2003 Annual Conference (IEEE Cat. No. 03TH8734)*, volume 2, pages 1388–1393. IEEE, 2003.

[32] Asif Farooq and David JN Limebeer. Path following of optimal trajectories using preview control. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2787–2792, 2005.

[33] Nidhika Birla and Akhilesh Swarup. Optimal preview control: A review. *Optimal Control Applications and Methods*, 36(2):241–268, 2015.

[34] MY El Ghoumari, H-J Tantau, and J Serrano. Non-linear constrained mpc: Real-time implementation of greenhouse air temperature control. *Computers and electronics in agriculture*, 49(3):345–356, 2005.

[35] A Rahideh and MH Shaheed. Constrained output feedback model predictive control for nonlinear systems. *Control engineering practice*, 20(4):431–443, 2012.

[36] Jinpeng Yu, Lin Zhao, Haisheng Yu, and Chong Lin. Barrier lyapunov functions-based command filtered output feedback control for full-state constrained nonlinear systems. *Automatica*, 105:71–79, 2019.

[37] Dong-Juan Li, Jing Li, and Shu Li. Adaptive control of nonlinear systems with full state constraints using integral barrier lyapunov functionals. *Neurocomputing*, 186:90–96, 2016.

[38] Alberto Bemporad. Reference governor for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 43(3):415–419, 1998.

[39] Elmer Gilbert and Ilya Kolmanovsky. Nonlinear tracking control in the presence of state and control constraints: a generalized reference governor. *Automatica*, 38(12):2063–2073, 2002.

[40] Emanuele Garone and Marco M Nicotra. Explicit reference governor for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 61(5):1379–1384, 2015.

[41] I. Kolmanovsky and Jing Sun. Parameter governors for discrete-time nonlinear systems with pointwise-in-time state and control constraints. In *Proceedings of the 2004 American Control Conference*, volume 4, pages 3075–3081 vol.4, 2004.

[42] Jinay S Gadda and Rajaram D Patil. Quadcopter (uavs) for border security with gui system. *International Journal of Engineering Research and Technology*, 2(12):620–624, 2013.

[43] Disha Amrutlal Gandhi and Munmun Ghosal. Novel low cost quadcopter for surveillance application. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 412–414, 2018.

[44] Víctor H Andaluz, Edison López, David Manobanda, Franklin Guamushig, Fernando Chicaiza, Jorge S Sánchez, David Rivas, Fabricio Pérez, Carlos Sánchez, and Vicente Morales. Nonlinear controller of quadcopters for agricultural monitoring. In *International Symposium on Visual Computing*, pages 476–487. Springer, 2015.

[45] Beibei Xu, Wensheng Wang, Greg Falzon, Paul Kwan, Leifeng Guo, Zhiguo Sun, and Chunlei Li. Livestock classification and counting in quadcopter aerial images using mask r-cnn. *International Journal of Remote Sensing*, 41(21):8121–8142, 2020.

[46] Muhammad Fadhil Abdullah, Inung Wijayanto, and Angga Rusdinar. Position estimation and fire detection based on digital video color space for autonomous quadcopter using odroid xu4. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*, pages 169–173. IEEE, 2016.

[47] Pengcheng Wang, Zhihong Man, Zhenwei Cao, Jinchuan Zheng, and Yong Zhao. Dynamics modelling and linear control of quadcopter. In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 498–503. IEEE, 2016.

[48] Ian D. Cowling, Oleg A. Yakimenko, James F. Whidborne, and Alastair K. Cooke. A prototype of an autonomous controller for a quadrotor uav. In *2007 European Control Conference (ECC)*, pages 4001–4008, 2007.

[49] Rong Xu and Umit Ozguner. Sliding mode control of a quadrotor helicopter. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 4957–4962, 2006.

[50] Arshad Mahmood and Yoonsoo Kim. Decentrailized formation flight control of quadcopters using robust feedback linearization. *Journal of the Franklin Institute*, 354(2):852–871, 2017.

[51] H Merabti, I Bouchachi, and K Belarbi. Nonlinear model predictive control of quadcopter. In *2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 208–211. IEEE, 2015.

[52] Andrew Zulu and Samuel John. A review of control algorithms for autonomous quadrotors. *arXiv preprint arXiv:1602.02622*, 2016.

[53] G Ganga and Meher Madhu Dharmana. Mpc controller for trajectory tracking control of quadcopter. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–6. IEEE, 2017.

[54] Weihua Zhao and Tiauw Hiong Go. Quadcopter formation flight control combining mpc and robust feedback linearization. *Journal of the Franklin Institute*, 351(3):1335–1355, 2014.

[55] Nur Uddin, Hendra G Harno, and Rianto Adhy Sasongko. Altitude control system design of bicopter using lyapunov stability approach. In *2021 International Symposium on Electronics and Smart Devices (ISESD)*, pages 1–6. IEEE, 2021.

[56] M Navabi, Ali Davoodi, and Hamidreza Mirzaei. Trajectory tracking of under-actuated quadcopter using lyapunov-based optimum adaptive controller. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, page 09544100211010852, 2022.

[57] Ilya V Kolmanovsky and Jing Sun. Parameter governors for discrete-time nonlinear systems with pointwise-in-time state and control constraints. *Automatica*, 42(5):841–848, 2006.

[58] Oliver De Quadros. *Implementation of an Explicit Reference Governor in a Nonlinear Quadcopter System*. Northeastern University, 2015.

[59] Elie Hermand, Tam W. Nguyen, Mehdi Hosseinzadeh, and Emanuele Garone. Constrained control of uavs in geofencing applications. In *2018 26th Mediterranean Conference on Control and Automation (MED)*, pages 217–222, 2018.

[60] Walter Lucia, Giuseppe FranzĂ¨, and Mario Sznaier. A hybrid command governor scheme for rotary wings unmanned aerial vehicles. *IEEE Transactions on Control Systems Technology*, 28(2):361–375, 2020.

[61] Bryan Convens, Kelly Merckaert, Bram Vanderborght, and Marco M. Nicotra. Invariant set distributed explicit reference governors for provably safe on-board control of nano-quadrotor swarms. *Frontiers in Robotics and AI*, 8, 2021.

[62] Julian Förster. System identification of the crazyflie 2.0 nano quadrocopter. 2015.

[63] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

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

[65] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*, volume 26. Birkhäuser, 2012.

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

[67] Panagiotis D Christofides, Riccardo Scattolini, David Munoz de la Pena, and Jinfeng Liu. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41, 2013.

[68] Dong Jia and Bruce H Krogh. Distributed model predictive control. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 4, pages 2767–2772. IEEE, 2001.

[69] Brett T Stewart, Aswin N Venkat, James B Rawlings, Stephen J Wright, and Gabriele Pannocchia. Cooperative distributed model predictive control. *Systems & Control Letters*, 59(8):460–469, 2010.

[70] David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

[71] Daniel Limón, Ignacio Alvarado, TEFC Alamo, and Eduardo F Camacho. Robust tube-based mpc for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010.

[72] D.L. Marruedo, T. Alamo, and E.F. Camacho. Input-to-state stable mpc for constrained discrete-time nonlinear systems with bounded additive uncertainties. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 4619–4624 vol.4, 2002.

[73] Gilberto Pin, Davide M Raimondo, Lalo Magni, and Thomas Parisini. Robust model predictive control of nonlinear systems with bounded and state-dependent uncertainties. *IEEE Transactions on automatic control*, 54(7):1681–1687, 2009.

[74] Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. In *Nonlinear model predictive control*, pages 345–369. Springer, 2009.

[75] 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.

[76] Hans Joachim Ferreau, Hans Georg Bock, and Moritz Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(8):816–830, 2008.

[77] James A Primbs and Chang Hwan Sung. Stochastic receding horizon control of constrained linear systems with state and control multiplicative noise. *IEEE transactions on Automatic Control*, 54(2):221–230, 2009.

[78] Debasish Chatterjee, Peter Hokayem, and John Lygeros. Stochastic receding horizon control with bounded control inputs: A vector space approach. *IEEE Transactions on Automatic Control*, 56(11):2704–2710, 2011.

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

[80] Keng Peng Tee, Shuzhi Sam Ge, and Eng Hock Tay. Barrier lyapunov functions for the control of output-constrained nonlinear systems. *Automatica*, 45(4):918–927, 2009.

[81] Ben Niu and Jun Zhao. Barrier lyapunov functions for the output tracking control of constrained nonlinear switched systems. *Systems & Control Letters*, 62(10):963–971, 2013.

[82] Keng Peng Tee, Beibei Ren, and Shuzhi Sam Ge. Control of nonlinear systems with time-varying output constraints. *Automatica*, 47(11):2511–2516, 2011.

[83] Wei Sun, Shun-Feng Su, Yuqiang Wu, Jianwei Xia, and Van-Truong Nguyen. Adaptive fuzzy control with high-order barrier lyapunov functions for high-order uncertain nonlinear systems with full-state constraints. *IEEE Transactions on Cybernetics*, 50(8):3424–3432, 2019.

[84] Jason Choi, Fernando Castaneda, Claire J Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. *arXiv preprint arXiv:2004.07584*, 2020.

[85] Keng Peng Tee and Shuzhi Sam Ge. Control of nonlinear systems with partial state constraints using a barrier lyapunov function. *International Journal of Control*, 84(12):2008–2023, 2011.

[86] Yuan Yuan, Zheng Wang, Lei Guo, and Huaping Liu. Barrier lyapunov functions-based adaptive fault tolerant control for flexible hypersonic flight vehicles with full state constraints. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(9):3391–3400, 2018.

[87] Yan-Jun Liu, Shumin Lu, Shaocheng Tong, Xinkai Chen, CL Philip Chen, and Dong-Juan Li. Adaptive control-based barrier lyapunov functions for a class of stochastic nonlinear systems with full state constraints. *Automatica*, 87:83–93, 2018.

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

[89] E.G. Gilbert, I. 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 vol.1, 1994.

[90] A. Bemporad. Reference governor for constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 43(3):415–419, 1998.

[91] A. Bemporad, A. Casavola, and E. Mosca. Nonlinear control of constrained linear systems via predictive reference management. *IEEE Transactions on Automatic Control*, 42(3):340–349, 1997.

[92] 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.

[93] Joycer Osorio, Mario Santillo, Julia Buckland Seeds, Mrdjan Jankovic, and Hamid R. Ossareh. A reference governor approach towards recovery from constraint violation. In *2019 American Control Conference (ACC)*, pages 1779–1785, 2019.

[94] Uroš Kalabić and Ilya Kolmanovsky. Reference and command governors for systems with slowly time-varying references and time-dependent constraints. In *53rd IEEE conference on decision and control*, pages 6701–6706. IEEE, 2014.

[95] Joycer Osorio and Hamid R. Ossareh. A stochastic approach to maximal output admissible sets and reference governors. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 704–709, 2018.

[96] Marco M Nicotra and Emanuele Garone. The explicit reference governor: A general framework for the closed-form control of constrained nonlinear systems. *IEEE Control Systems Magazine*, 38(4):89–107, 2018.

[97] Takeshi Hatanaka and Kiyotsugu Takaba. Output feedback reference governor for nonlinear systems. *Transactions of the Society of Instrument and Control Engineers*, 41(10):803–812, 2005.

[98] Joycer Osorio. *Reference Governors: From Theory to Practice*. The University of Vermont and State Agricultural College, 2020.

[99] Steven W Chen, Tianyu Wang, Nikolay Atanasov, Vijay Kumar, and Manfred Morari. Large scale model predictive control with neural networks and primal active sets. *Automatica*, 135:109947, 2022.

[100] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.

[101] Vladimír Kučera. The discrete riccati equation of optimal control. *Kybernetika*, 8(5):430–447, 1972.

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

[103] T.A. Johansen, I. Petersen, and O. Slupphaug. On explicit suboptimal lqr with state and input constraints. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 1, pages 662–667 vol.1, 2000.

[104] Jeff B Burl. *Linear optimal control: H (2) and H (Infinity) methods*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[105] Miaomiao Ma and Hong Chen. Constrained h2 control of active suspensions using lmi optimization. In *2006 Chinese control conference*, pages 702–707. IEEE, 2006.

[106] C. Scherer, P. Gahinet, and M. Chilali. Multiobjective output-feedback control via lmi optimization. *IEEE Transactions on Automatic Control*, 42(7):896–911, 1997.

[107] Wei He, Amoateng Ofosu David, Zhao Yin, and Changyin Sun. Neural network control of a robotic manipulator with input deadzone and output constraint. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):759–770, 2015.

[108] Wei He, Yuhao Chen, and Zhao Yin. Adaptive neural network control of an uncertain robot with full-state constraints. *IEEE Transactions on Cybernetics*, 46(3):620–629, 2016.

[109] John J Hopfield and David W Tank. "neural" computation of decisions in optimization problems. *Biological cybernetics*, 52(3):141–152, 1985.

[110] Qiaoming Han, Li-Zhi Liao, Houduo Qi, and Liqun Qi. Stability analysis of gradient-based neural networks for optimization problems. *Journal of Global Optimization*, 19(4):363–381, 2001.

[111] Zeng-Guang Hou, Cang-Pu Wu, and Paul Bao. A neural network for hierarchical optimization of nonlinear large-scale systems. *International Journal of Systems Science*, 29(2):159–166, 1998.

[112] Xue-Bin Liang and Jun Wang. A recurrent neural network for nonlinear optimization with a continuously differentiable objective function and bound constraints. *IEEE Transactions on Neural Networks*, 11(6):1251–1262, 2000.

[113] Xin-Yu Wu, You-Shen Xia, Jianmin Li, and Wai-Kai Chen. A high-performance neural network for solving linear and quadratic programming problems. *IEEE transactions on neural networks*, 7(3):643–651, 1996.

[114] Nicolas Lanzetti, Ying Zhao Lian, Andrea Cortinovis, Luis Dominguez, Mehmet Mercangöz, and Colin Jones. Recurrent neural network based mpc for process industries. In *2019 18th European Control Conference (ECC)*, pages 1005–1010. IEEE, 2019.

[115] Li-Xin Wang and Feng Wan. Structured neural networks for constrained model predictive control. *Automatica*, 37(8):1235–1243, 2001.

[116] Julian Nubert, Johannes Köhler, Vincent Berenz, Frank Allgöwer, and Sebastian Trimpe. Safe and fast tracking on a robot manipulator: Robust mpc and neural network control. *IEEE Robotics and Automation Letters*, 5(2):3050–3057, 2020.

[117] Xiaojing Zhang, Monimoy Bujarbaruah, and Francesco Borrelli. Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework. *IEEE Transactions on Control Systems Technology*, 2020.

[118] Kenneth H Loewenthal and Steven M Bryant. Neural network optical character recognition system and method for classifying characters in a moving web, January 27 1998. US Patent 5,712,922.

[119] Boukaye Boubacar Traore, Bernard Kamsu-Foguem, and Fana Tangara. Deep convolution neural network for image recognition. *Ecological Informatics*, 48:257–268, 2018.

[120] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 5209–5217, 2017.

[121] S Reynold Chu, Rahmat Shoureshi, and Manoel Tenorio. Neural networks for system identification. *IEEE Control systems magazine*, 10(3):31–35, 1990.

[122] Paul J Werbos. Neural networks for control and system identification. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 260–265. IEEE, 1989.

[123] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):1–74, 2021.

[124] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.

[125] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[126] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[127] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[128] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[129] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010.

[130] Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation. *IEEE transactions on neural networks*, 16(1):57–67, 2005.

[131] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. *International Scholarly Research Notices*, 2012, 2012.

[132] Guang-Bin Huang and Chee-Kheong Siew. Extreme learning machine: Rbf network case. In *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, volume 2, pages 1029–1036. IEEE, 2004.

[133] Hikmet Kerem Cigizoglu and Murat Alp. Generalized regression neural network in modelling river sediment yield. *Advances in Engineering Software*, 37(2):63–68, 2006.

[134] Donald F Specht et al. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[135] Hong-ze Li, Sen Guo, Chun-jie Li, and Jing-qi Sun. A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. *Knowledge-Based Systems*, 37:378–387, 2013.

[136] Shiow-Shung Yang and Ching-Shiow Tseng. An orthogonal neural network for function approximation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(5):779–785, 1996.

[137] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

[138] Joydeep Ghosh and Yoan Shin. Efficient higher-order neural networks for classification and function approximation. *International Journal of Neural Systems*, 3(04):323–350, 1992.

[139] Ernesto A Paiva, Juan C Soto, Julio A Salinas, and William Ipanaqué. Modeling and pid cascade control of a quadcopter for trajectory tracking. In *2015 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, pages 809–815. IEEE, 2015.

[140] Nengsheng Bao, Xie Ran, Zhanfu Wu, Yanfen Xue, and Keyan Wang. Research on attitude controller of quadcopter based on cascade pid control algorithm. In *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1493–1497. IEEE, 2017.

[141] Endrowednes Kuantama, Ioan Tarca, and Radu Tarca. Feedback linearization lqr control for quadcopter position tracking. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 204–209, 2018.

[142] Faraz Ahmad, Pushpendra Kumar, Anamika Bhandari, and Pravin P Patil. Simulation of the quadcopter dynamics with lqr based control. *Materials Today: Proceedings*, 24:326–332, 2020.

[143] Alberto Bemporad, Carlo A Pascucci, and Claudio Rocchi. Hierarchical and hybrid model predictive control of quadcopter air vehicles. *IFAC Proceedings Volumes*, 42(17):14–19, 2009.

[144] Endrowednes Kuantama, Ioan Tarca, and Radu Tarca. Feedback linearization lqr control for quadcopter position tracking. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 204–209. IEEE, 2018.

[145] Michael Achtelik, Thomas Bierling, Jian Wang, Leonhard Höcht, and Florian Holzapfel. Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties. In *Infotech@ Aerospace 2011*, page 1485. 2011.

[146] Diego Domingos, Guilherme Camargo, and Fernando Gomide. Autonomous fuzzy control and navigation of quadcopters. *IFAC-PapersOnLine*, 49(5):73–78, 2016. 4th IFAC Conference on Intelligent Control and Automation SciencesI-CONS 2016.

[147] Yudan Liu, Joycer Osorio, and Hamid R Ossareh. Decoupled reference governors: a constraint management technique for mimo systems. *International Journal of Control*, pages 1–20, 2021.

[148] Yudan Liu and Hamid R Ossareh. Preview reference governors: A constraint management technique for systems with preview information. *Systems & Control Letters*, 152:104932, 2021.

[149] Yudan Liu, Joycer Osorio, and Hamid Ossareh. Decoupled reference governors for multi-input multi-output systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1839–1846. IEEE, 2018.

[150] 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.

[151] 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.

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

[153] Peter L Falb and William A Wolovich. Decoupling in the design and synthesis of multivariable control systems. *IEEE Transactions on Automatic Control*, 1967.

[154] S Lloyd. Decoupling a multivariable discrete-time system. *Electronics Letters*, 6(26):831, 1970.

[155] L Silverman. Decoupling with state feedback and precompensation. *IEEE Transactions on Automatic Control*, 15(4):487–489, 1970.

[156] Ronald DeVore, Boris Hanin, and Guergana Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

[157] Subir Kumar Saha. *Introduction to robotics*. Tata McGraw-Hill Education, 2014.

[158] Selim Solmaz, Martin Corless, and Robert Shorten. A methodology for the design of robust rollover prevention controllers for automotive vehicles: Part 2-active steering. In *2007 American Control Conference*, pages 1606–1611, 2007.

[159] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 22:22, 2011.

[160] Carlos Luis and JĂŠrĂ´me Le Ny. Design of a trajectory tracking controller for a nanoquadcopter, 2016.

[161] Guanrong Chen. Stability of nonlinear systems. *Encyclopedia of RF and Microwave Engineering*, pages 4881–4896, 2004.

[162] Christopher John Harris and JME Valenca. *The stability of input-output dynamical systems*, volume 168.

[163] Uros Kalabic. Reference governors: Theoretical extensions and practical applications. 2015.

[164] Eric Colin Kerrigan. *Robust constraint satisfaction: Invariant sets and predictive control*. PhD thesis, University of Cambridge, 2001.

[165] B Pluymers, JA Rossiter, JAK Suykens, and Bart 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. IEEE, 2005.

[166] Tzon-Tzer Lu and Sheng-Hua Shiou. Inverses of 2×2 block matrices. *Computers & Mathematics with Applications*, 43(1-2):119–129, 2002.

[167] ShiNung Ching, Yongsoon Eun, Eric Gross, Eric Hamby, Pierre Kabamba, Semyon Meerkov, and Amor Menezes. Modeling and control of cyclic systems in xerography. In *Proceedings of the 2010 American Control Conference*, pages 4283–4288, 2010.

[168] Samia M Mahil and Ahmed Al-Durra. Modeling analysis and simulation of 2-dof robotic manipulator. In *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4. IEEE, 2016.

[169] Ali Y Ungoren and Huei Peng. An adaptive lateral preview driver model. *Vehicle system dynamics*, 43(4):245–259, 2005.

[170] Richard H Byrd, Mary E Hribar, and Jorge Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[171] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Appendix A

Crazyflie Code and Common Questions

The Crazyflie can be controlled through mobile app or computer. More specifically, the mobile app can be downloaded through Bitcraze website. In the app, the desired roll angle, pitch angle, and yaw angle can be sent to the Crazyflie and the latest version of the firmware can be updated. To control the Crazyflie via computer, Crazyradio PA is necessary to be used. Detailed information on this topic and corresponding python code will be presented below.

A.1 Logging

Logging configurations are used for logging variables from the firmware. Each log configuration contains a number of variables that should be logged as well as a time period (in millisecond) of how often the data should be sent back to the host. Once the log configuration is added to the firmware, the firmware will automatically send back the data at every period. These configurations are implemented in the following ways:

1. Connect to the Crazyflie.
2. Create a log configuration that contains the variables that are required to be logged
3. Add the log configuration and define the log period
4. Set up callbacks for the data and start the log configuration
5. Each time the firmware sends data back to the host, the callback function will the called with a time-stamp and the data

Detailed python code is demonstrated below:

```
import numpy as np
import logging
import cflib.crtp
from cflib.crazyflie import Crazyflie
from cflib.crazyflie.syncCrazyflie import SyncCrazyflie
from cflib.crazyflie.log import LogConfig
from cflib.positioning.motion_commander import MotionCommander

# URI of the Crazyflie to connect to, most are channel 80
uri = 'radio://0/80/2M/E7E7E7E7E7'

# DataNames and Data_Lists are for all the logging variables to be logged
DataNames = ['stabilizer.roll','stabilizer.pitch', 'stabilizer.yaw']
Data_List = [0]* len(DataNames1)

# ensure the Crazyflie has the flow deck attached
is_deck_attached = False
def param_deck_flow(name, value_str):
    value = int(value_str)
    global is_deck_attached
    if value:
        is_deck_attached = True
        print('Deck is attached')
    else:
        is_deck_attached = False
        print('Deck is NOT attached')

#logging callbacks for each of the logging groups
def log_stab_callback(timestamp, data, logconf):

    for i in range(0,len(DataNames)):
        Data_List[i] = data[DataNames[i]]
    with open("Data.txt", "a") as file:
        file1.write('%d' % timestamp)
        for i in range(0,len(DataNames)):
            file.write('\t%s' % Data_List[i])
        file.write('\n')
# this is the method that the drone runs first
if __name__ == '__main__':
    cflib.crtp.init_drivers()

    # initalizing logg for data set. The logging rate is 10ms
    Log_Config1= LogConfig(name='Attitude', period_in_ms=10)

    # add variables to the logging configurations
    for i in range(0,len(DataNames)):Log_Config.add_variable(DataNames[i], 'float')

```

```
    # 'run_method' is executed to send the commands
    if is_deck_attached:
        run_method(scf, Log_Config)
```

A.2 Command Sending

Below, for simplicity, the python code for M-RG will only be presented. And the structures of the code for NL-DRG and NN-DRG are similar to that for M-RG. The Crazyflie executes the "run_method" to send the commands:

```
from math import pi
# code to compute the decision variable: kappa
# the inputs are the MAS (Hx, Hv, and h), reference: r, the states: x,
# and the previous governed command: v
def kappa_compute(Hx,Hv,h,r,v,x):
    kappa=1.0;
    a=np.multiply(Hv,(r-v));
    b=np.array(h)-np.matmul(Hx,x)-np.multiply(Hv,v);
    for j in np.arange(0, len(Hx)):
        if b[j] > 0.0:
            if a[j] > 0.0:
                kappa=min(kappa,b[j]/a[j])
        if b[j] <= 0.0:
            kappa=0
    kappa = np.amax(kappa,0);
    return kappa

# Log_Config1 represents the roll angle and pitch angle
# Log_Config2 refers to the angular velocity: p, q, and r
def run_method (scf, Log_Config1, Log_Config2):
    # starting all logging configurations
    Log_Config1.start()
    Log_Config2.start()

    # must give 0,0,0,0 command first
    scf.cf.commander.send_zdistance_setpoint(0,0,0,0)
    time.sleep(.1)

    # using motion commander to control the initial height when take off
    # when this is called, drone will rise to DEFAULT_HEIGHT (m)
    DEFAULT_HEIGHT = 1
    with MotionCommander(scf, default_height = DEFAULT_HEIGHT) as mc:
        time.sleep(1)
        # the sending rate is 10ms and the code runs 2s
        for i in range (0,200):
            # set the setpoints: roll angle is 6 degree and yaw rate is 12 degree/s
```

```
            r_roll,r_pitch,r_dyaw=6*pi/180,0,12*pi/180

            # get the current states
            roll=float(Data_List1[0])*pi/180
            pitch=float(Data_List1[1])*pi/180
            q=float(Data_List2[1])*pi/180
            r=-float(Data_List2[2])*pi/180

            # compute the current governed command for roll angle
            x_temp1=[[x_angle1], [x_rate1], [p],[roll]
            # The MAS for roll angle is represented by Hx1, Hv1, and h1
            kappa1=kappa_compute(Hx1,Hv1,h1,r_roll,v0,x_temp1)
            v0=float(v0+kappa1*(r_roll-v0))

            # compute the current governed command for pitch angle
            x_temp1=[[x_angle2], [x_rate2], [q],[pitch]
            # The MAS for pitch angle is represented by Hx2, Hv2, and h2
            kappa2=kappa_compute(Hx2,Hv2,h2,r_pitch,v1,x_temp2)
            v1=float(v1+kappa2*(r_pitch-v1))

             # compute the current governed command for yaw rate
            x_temp3=[[x_rate3],[r]]
            # The MAS for yaw rate is represented by Hx3, Hv3, and h3
            kappa3=kappa_compute(Hx3,Hv3,h3,r_dyaw,v2,x_temp3)
            v2=float(v2+kappa3*(r_dyaw-v2))

            # send the command with the unit in degree
            scf.cf.commander.send_zdistance_setpoint(v0*180/pi,v1*180/pi,v2*180/pi,
            DEFAULT_HEIGHT)

            # the computation time of M-RG is around 0.006s.
            # to ensure the sending rate is 0.01s, the code will pause for 0.004s.
            time.sleep(.004)
        mc.start_back()
        time.sleep(1)

    time.sleep(1)

    Log_Config1.stop()
    Log_Config2.stop()
```

A.3 Communication

The communication from computer to Crazyflie is achieved via Crazyraio PA[1], which is a long range open USB radio dongle. After executing "run_method", the results will be sent to the quadcopter via Crazyraio PA.

A.4 Common Questions for Crazyflie

- **Q**:Wwhy the Crazyflie can not take off even the flow deck is attached and command is sent successfully?
  A: There are several things we can check:
  - are the propellers assembled correctly? More specifically, the propellers with "A" labeled should attached to the arms with M_2 and M_4 labeled. The propellers with "B" should attached to the the arms with M_1 and M_3.
  - does the motor work properly?
  - are the wires connected the motor mounts to the wings broken?
- **Q**: What happen if the code returns "Deck is NOT attached"?
  A: This is highly because the deck is not attached correctly, More specifically, the forward direction is labeled in both the flow deck and Crazyflie. And they should be pointed in the same direction
- **Q**: Why the Crazyflie can not fly as expected when above the dark floor?
  A: This is because the sensor in the Crazyflie can not recognize the altitude distance from a dark floor.
- **Q**: why the blue LED in M_2 arm lights up solidly when turning on the Crazyflie?
  A: This may happen when the firmware of the Crazyflie is updated while the Crazyflie has a low battery. It is recommended to charge the power when updating the Crazyflie. But if this situation happens, we can resolve it using following steps:
  1. open the Crazyflie PC client[2]
  2. insert the Crazyradio to the computer and, in the PC client, scan the Crazyflie around and connect the one we want

[1] https://www.bitcraze.io/products/crazyradio-pa/

[2] https://www.bitcraze.io/documentation/repository/crazyflie-clients-python/master/

3. open the "bootloader", programming the old version of the firmware into the Crazyflie, such as "2016.11-Crazyflie"
4. after the programming finished, update the Crazyflie firmware with the latest version. Be carefully that we should select the version with "CF2" instead the ones with "Bolt" or "Tag".

- **Q**: why the code returns "Too many packets lost"?
  A: This might because of the wrong number of the channel in the URL of the Crazyflie. For most cases, the channel number should be 80. But if is isn't, the channel number can be found in the Crazyflie PC client