NEW PARADIGMS FOR DESIGN AND CONTROL OF DYNAMICAL NETWORKS

By

YAN WAN

A thesis submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering & Computer Science

May 2009

To the Faculty of Washington State University

The members of the Committee appointed to examine the dissertation of YAN WAN find it satisfactory and recommend that it be accepted.

Sandip Roy, Chair

Ali Saberi

Anton Stoorvogel

Bernard Lesieutre

ACKNOWLEDGEMENT

Looking back on my path over the past four-and-a-half years of my Ph.D studies, I could not be more grateful for all the experiences that I have had and all the people that I have met. I established an important body of research, worked with the most brilliant researchers, had enjoyable time with a lot of friends, and was reborn in the name of Jesus Christ.

First of all, I am grateful to my advisor, Dr. Sandip Roy, the best advisor that a student could ever imagine—brilliant, responsible, encouraging, and caring. Four years ago, I did not fully recognize my research potential, until he and Dr. Ali Saberi excited me with a class research project on decentralized partitioning. Sandip guided me very closely in the project and displayed to me this picture that doing research is so much fun. I was amazed by his way of doing research—he seems to me like a brave but experienced explorer of an unknown world and also a little kid playing with his favorite toys, very happy, focused, skilled, enthusiastic, and energetic. After I joined his group, he provided me systematic training in every aspects of research, including e.g., coming up with ideas, working out complete results, and writing. When I was stuck in research, he could always point out directions that broke the ice. He is also incredibly good and insightful in finding connections among topics. I am especially grateful for all the discussions with him on figuring out new directions to work on. Other than research, he also set the model for me as a successful educator. His class is full of fun, and he has the ability to explain complicated concepts in an easy-to-understand way. Moreover, he is willing to give huge time after class to his students. Sandip and I are also close (life-time) friends, and he had always offered me tremendous help in aspects other than research.

I am also grateful to Dr. Saberi, who also guided me closely in research, and from whom I gained significant knowledge in the area of Systems and Control. He impressed me as a master of Systems and Control with deep and precise understandings of the heart of the field. His conversations with me were always very precious and beneficial, since I could obtain deep insights of the field from him that are hard to find elsewhere, and get clear pictures of which topics are worth exploring. I am indebted to him for being so supportive of me and his effort in training me to become a researcher who is enthusiastic, insightful, highly focused, and dedicated, as a reflection of himself.

I also want to express my gratefulness to Dr. Anton Stoorvogel from the University of Twente, the Netherlands. I really appreciate him flying to Pullman every year to guide the students in our group in research, and also to come to my defense in this harsh winter season. I am deeply impressed by his incredible ability in solving tough mathematical problems. Working and talking with him is very inspiring and a lot of fun. I would also like to thank Dr. Bernard Lesieutre from the University of Wisconsin-Madison for serving on my committee, and for the various insightful discussions on several research topics.

Outside my research committee, I would like to give my thanks to other faculty and staff both in my department and other schools. Dr. Zhe Dang, thanks to whom I started to build a Computer Science background, is a well-respected researcher and a terrific friend. I will always remember many insightful talks he gave in class and in person, concerning the interesting topics in the field of Algorithmics, and also the philosophies of research and life. He is always there for help when I come up to him. I really appreciate his caring and support. I am also grateful to Dr. Shira Broschat and Dr. Douglas Call for their training in biology and for all their helps; and to Dr. James Krueger, Dr. David Rector, and Dr. Chris Davis for the collaboration on the interesting project on sleep regulation. My thanks also goes to Ms. Ruby Young, who is retired but showed her caring for all the students in the department for many years, Ms. Sidra Gleason for all the help and being so nice, and also Ms. Ning Hsu for all the enjoyable conversations and encouragement.

At this point, I could not forget to give thanks to the educators who impacted my life and paved my path piece by piece up to the point that I am now: my elementary school mathematics teachers Ms. Yan Xu and Ms. Xiangyang Tong, Mr. Honggen Wang, who taught me middle school Physics, my wonderful undergraduate instructors Dr. Jianjiang Zhou (EE) and Dr. Baiqing He (Math), and my Master's advisor Dr. Larry Wurtz.

My research is supported by National Science Foundation grants ECS-0528882 and ECCS-0725589, and National Aeronautics and Space Administration grant NNA06CN26A. I also obtained multiple travel grants from the School of Engineering and Computer Science and also the Graduate School at Washington State University. I am grateful for the support from these agencies.

I am also grateful for the fellowship of students in our university, Mora Xue for her smile and accompaniment, Linmin Yang and Huan Zhao for the fun and laughs we had together, Tao Yang and Kevin Chang for the help and kindness, and also Zheng Wen, Jasmin Minteer, Xu Wang, Babak Malek, and Da Meng for the conversions on research and the wonderful time we had together.

Thanks to my friends in Pullman and Moscow, who made my life here much more fun. I am indebted to Dr. Tianxi Zhang and Ms. Yirong Sun, who cared for me as their kid (though I am much older than their kids), and invited me over so many times to have a delicious dinner. I am also grateful for the support from and the wonderful time I had with my friends in the local Chinese church, Paster Yeh and his family, Lei Wang, Fei Pan, Tracy, Hongkun, Jingyi, Yan Zhuang, Winnie, Edison, Yun Chen, Minhua, and many others.

No words can even come close to expressing my thanks to my sweet family, to whom I own so much and who are the the most precious treasure of my life. I am indebted to my parents for their tremendous love, and the trust and support for whatever decisions that I made; my brother, sister-in-law and my grandpa for their love and care; and my uncle whom I will miss forever, for his encouragement at my early age. And specially thanks to my beloved husband Haibo Zhu for being my best friend, being there always for help and support, and making every day of my life a happy day.

Finally, I would like to give my thanks to the Lord, for all the rejoicing and the tears. Thank you, Lord, for your tremendous mercy and love. Thank you for all the blessings and forgiveness that you gave me, and for everything in my life!

NEW PARADIGMS FOR DESIGN AND CONTROL OF DYNAMICAL NETWORKS

Abstract

by Yan Wan, Ph.D.
Washington State University
May 2009

Chair: Sandip Roy

My Ph.D. research is focused on devising novel paradigms for the design and control problems in dynamical networks that arise in various infrastructure- and agent- network applications. While network structure is ubiquitous in many modern systems and so network dynamics have been systematically analyzed, works on design and control of networks (i.e., shaping the dynamics of a network through applying static/dynamic feedback controllers) are sparse. Despite the lack of works in network design and control, solutions in this field are badly needed in many network applications such as traffic management, epidemic control, and sensor/vehicle networking.

In this research, we provide systematic solutions to design and control problems that are common to many dynamical network applications. To develop these solutions, we are engaged in effort in two major aspects: 1) modeling and identifying design and control problems in network applications; 2) developing novel tools for network design and control. In the development, our philosophy is that network design and control must exploit network structure to be effective, due to the significant role played by networks' structures in their dynamics. With this philosophy in mind, we have systematically addressed multiple important and typical network design and control tasks, including designing (static) network node properties and edge properties (at all or a subset of the nodes/edges), and also designing decentralized dynamical controllers, to meet eigenvalue-related performance requirements. We have also addressed novel controller design for networks that are present to constraints such as input saturation and delays.

Part of the work is concerned with studies that support the above development of network design and control methods. We provide some interesting results on infinite-dimensional systems, stabilization of systems with saturation, and the maneuvering of system zeros. Moreover, we have also addressed network-related numerical tasks, such as distributed network partitioning and effective network simulation.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

To my dear parents Xinming Wan and Sulan Fu,

my husband Haibo Zhu,

my brother Li Wan and his family Xia Yang and Yuhao Wan,

my grandpa Xiuliang Wan,

and the Lord all mighty, my savior and my support.

In memory of my uncle Professor Xinguang Wan.

1. INTRODUCTION

Large-scale networks are fascinating: a group of agents with relatively simple local dynamics can coordinate to complete very complicated network tasks. Networks are ubiquitous and widely analyzed. However, surprisingly little is known concerning the design and control of the dynamics of **modern large-scale networks**, which exhibit two common characteristics: 1) network structure is critical to the network dynamics and 2) control must be achieved in the face of severe constraints and variations.

We aim to contribute to the challenging and rich field of network theory, by developing general tools for the crucial tasks of design and control in modern networks. To develop these tools, we are engaged in research in two deeply-coupled directions: 1) modeling and abstraction of critical dynamical design/control problems in various **autonomous-agent** and **infrastructure networks**; 2) development of general new control-theoretic methods for solving these problems ([1–21]). Throughout this development, we take the perspective that a network's structure (the interactions between network agents or components) is a key factor determining its dynamics and must be exploited in control/design [5]. Our research has several applications, ranging from micro- and macro-biological/ecological systems [1, 4, 7, 21], to air traffic, power, and sensor networks [2, 3, 7, 19, 21]. Our tools for control/design contribute to these applications, and also to fundamental development in such areas as decentralized control, algebraic graph theory, and large-scale simulation.

Here, we will give a brief introduction of our research in the following four aspects. First, we will motivate and formulate the core dynamical network control/design problems using four applications: air traffic management, virus spread control, sleep regulation, and sensor/vehicle networking (Section 1.1). Second, we will introduce the two exciting new design methodologies (one concerned with static control, the other with dynamical or memoried control) that address these core design problems (Section 1.2 and Section 1.3). Third, we will briefly describe some other interesting network-related numerical problems in Section 1.4.

1.1 Modeling: Network Control and Design Problem Formulations

A primary aspect of my research is to identify design and regulation (control) problems in modern networks. In comparison to the classical centralized system applications, modern network applications have a major challenge: multiple agents must interact in a decentralized fashion to complete cooperative tasks. This interaction significantly increases the difficulty of the control problem itself, as well as its tractability. Thus, network modeling requires abstractions that capture only the most intrinsic properties of both local dynamics and network interactions to be tractable. We stress here that because network structure highly influences network dynamics, network structure is a key aspect to consider in modeling. However, a bulk of network design/control works either do not start with a tractable model or totally ignore network structure in the modeling, and hence are far from effective in obtaining practical design and controls. Here, we describe our modeling and problem formulation, which serve as a basis for design and control, in four network applications: air traffic management, virus spread control, sleep regulation, and sensor/vehicle networking.

Air Traffic Management Air traffic flow management actions in the United States National Airspace System (NAS) are complicatedly interdependent. We have taken the perspective that good flow management strategies must be designed at a network level, by taking into consideration traffic in multiple Centers in the presence of uncertainty (e.g., severe weather, airborne, taxi-in and taxi-out delays). In order to design optimal network-level flow management strategies, we emphasize the full understanding of **boundary restrictions**' impact on generic traffic flows, with the aim of developing restriction abstractions for tractable network evaluation/optimization. In [2], we examined several abstractions (e.g., the detailed queueing model, the discrete-time saturation model, and the dynamic linear model) for boundary restrictions. Based on the essential dynamics of these models, we introduced a highly-abstracted network model: the **algebraic linear model**. In this model, each restriction trades off the variance of the **outflow** with **backlog** in a linear fashion, and each flow merges or splits at a boundary. Using this model, we posed the network-level optimal restriction design problem as a *graph edge design problem* with the objective of minimizing an outflow variance plus mean backlog cost. We also studied the modeling and computation of sensitivities of NAS performance to disturbances, and the use of sensitivities in air traffic flow management [3].

Virus Spread Control The spread of viruses (e.g., computer viruses or biological epidemics like SARS or influenza) also demonstrates typical modern large-scale network behavior. The global virus spread pattern is determined by both local dynamics (e.g. an individual or a group of closely contacted individuals to become infected, recovered, or removed, etc.) and network interactions (e.g. the transmission of virus from/to its contacting agents). Each **agent** (an individual or a group of closely contacted individuals) has local dynamics (e.g., to become infected, recovered, or removed, etc.), and is also involved in the virus transmission from/to its contacting agents. Together, the local and network dynamics determine the global virus spread pattern in the population. Hence in a practical virus spread model, whether a disease will spread throughout a population or vanish depends on both intra-agent parameters for each agent (e.g., **local transmission rate**, **infectious period**, and **recovery rate**), and inter-agent parameters between each pair of contacting agents (e.g., **inter-agent transmission rate**). Although population structure is believed to be crucial for epidemic spread, little work is concerned with designing control strategies (e.g., isolation, quarantine, fast hospitalization, etc.) that exploit the network structure. It is our perspective that designing *heterogeneous* control strategies can stop virus spread faster with less **control resources** used. In [1], we incorporated the control parameters that exploit population structure in a computer virus spread model and a biological-epidemic model, and examined the problem of allocating limited control resources to the network to minimize the speed of virus spread in these two models. Through modeling and analysis, the control resource allocation problems reduce to the problem of designing a diagonal matrix D/K to minimize the dominant eigenvalue of $D + KG$ subject to constraints on D and K, where the matrix G represents the topology of spread.

Sleep Regulation Network Sleep is a fundamental biological process. The experimental research on sleep mechanisms suggested that sleep is activity-dependent and is not a centralized phenomenon. For instance, moving a single whisker can cause parts of rat's brain, and in turn the entire organism, to go to sleep faster or enter a deeper sleep. However, there is no work that models the translation of local activity (activity at one or a small number of functional units known as cortical columns) into a global sleep state, i.e., the regulation of sleep through external activity. We view organism sleep as emerging from the local sleep states of the functional units; these local sleep states evolve through integration of local activity inputs, loose couplings with neighboring cortical columns, and global regulation (e.g. by the circadian clock). In [4], we constructed a network model that captures the impact of the network connections. In the model, these cortical columns—represented as coupled or networked activity integrators—transit between sleep and waking states based on thresholds on the total activity. The model dynamics for the several canonical experiments (which we have studied both through simulation and system-theoretic analysis) gives a qualitative explanation for sleep evolution and emphasizes the role of network couplings in sleep regulation. The model also motivates the study of and allows us to verify the network's **external stability** (disturbance rejection), i.e., the coordination in the presence of small persistent variations in the activity inputs.

Sensor Networks and Vehicle Control Autonomous vehicle teams and distributed sensor networks are predicted to be important parts of future transportation and communication systems, due to the following advantages: 1) security; 2) the ability to work in human-unsuitable environments; 3) ease in implementation (e.g., centralized communication/control is unnecessary). We describe a sensor/vehicle network using the following two structures: 1) the local dynamics for a agent (e.g., a double-integrator model is typical for a vehicle) and 2) the **sensing architecture**, which characterizes the set of information that each agent can observe in the network. Having a decentralized sensor/vehicle network complete certain tasks is challenging, especially under some practical considerations, such as the presence of input saturation and actuation delays. Let me summarize three control/design problems that we have worked on. First, we have sought to minimize the **settling time** of a distributed sensor fusion algorithm through designing the sensing communication structures. The problem is formulated as selecting the edge weights of a graph subject to an upper bound on their total, so as to maximize a spectral measure [6]. Second, we are interested in various tasks performed by autonomous vehicle teams, like **formation** tasks. In [10–12, 14], we have sought novel decentralized controls that allow autonomous vehicle teams to accomplish these various tasks in the face of delay and saturation. Third, in [19, 20], we have motivated and developed an algorithm for decentralized network partitioning.

1.2 Dynamical Network Design

A number of the critical dynamical network-related tasks that we have identified are fundamentally concerned with network design, i.e., the design of network interconnections to shape dynamics. For instance, the air traffic management problem is to set boundary restrictions to minimize a network-wide cost, and the epidemic control problem is to allocate control resources across the network to minimize the speed of virus spread. Given the special characteristics of modern dynamical networks, we take the perspective that dynamical network design must take account of topological structure to achieve desired performance. However, dynamical network designs that exploit network structure are essentially nonexistent. The major difficulties of structural dynamical network design are the following: 1) the design is intrinsically decentralized; 2) practical issues like the constraints on design variables, implementability, and the robustness of design need to be considered. We address the dynamical network design problem by introducing new tools that resolve these difficulties.

Although the applications that we address vary widely, the structures of all these networks can be abstracted to graphs, with different interpretation of nodes and edges. Based on this abstraction, we classify network design problems into two categories: **graph node design** and **graph edge design**. Specifically, graph node design is concerned with designing node properties (e.g., affecting all branches coming/leaving a node, or affecting local dynamics) to shape the dynamics of the system subject to certain constraints. Graph edge design, instead, aims at designing edges for the same goal. Typical node design problems are the air traffic management and virus spread control problems that we formulated above. Edge design applications are also abundant: e.g., the problem of minimizing settling time of a distributed sensor fusion algorithm. We have developed new methods to solve these two design problems, which mesh optimization machinery together with eigenvalue sensitivity and algebraic graph theory. The results contribute significantly to the field of static decentralized controller design and its applications. Here we discuss these two graph design tasks and the methods addressing them in some depth.

Graph Node Design A representative graph node property design problem can be abstracted to the following linear algebra problem: design a *diagonal* matrix D/K to minimize a measure (e.g., the dominant eigenvalue) of $D + KG$ subject to the constraints that $0 \leq D/K \leq L$ and $\sum K \leq \Gamma$, where G captures the network topology [1, 8]. This problem is equivalent to a static *decentralized control* problem: the control actions at an agent only directly impacts the dynamics locally and are constrained (mathematically, only alter the system matrix through multiplication/addition of a constrained diagonal matrix) and yet must be used in coordination to shape the dynamics globally.

By using tools from algebraic graph theory and optimization theory, e.g., the Courant-Fischer theorem, eigenvalue sensitivity, and Lagrange multipliers, we have characterized the eigenstructure of $D + KG$ associated with the optimal solution, and hence devised finite-search algorithms for the optimal design. For certain common classes of graph topologies, further structural results are obtained, which have led to the development of algorithms for network designs with highly reduced complexity. Rather than only providing the optimal solution, our novel design approach also gives us insights about the optimal design. For instance, it suggests that the optimal D matrix is the one that equalizes the row sums of $D + KG$, as best as possible within the constraint on D. In the case the individual constraints prevent such a design, the entries in neighboring nodes are adjusted to obtain the constrained optimum.

Graph Edge Design We have developed similar tools to address graph edge design [6]. Although the methods are similar, we note that graph edge design problems have a property distinct from that of the graph node design problems: edges in a cyclic graph have *redundant* contributions to the associated network dynamics; this fact suggests that graph edge designs usually have multiple optimal solutions. Our design method finds the common structural characteristics of the optimal solutions, and is capable of obtaining multiple optima.

In many applications concerned with graph edge design, it is common that only a subset of edges in a graph are eligible for design. In this **partial graph edge design**, the subgraph with the fixed graph edges poses a crucial structure that limits the performance of designs. We bring tools from the structural study of linear systems, namely, the **Special Coordinate Basis** (SCB), to obtain the explicit relationship of this fixed edge graph and the *zero structure* of a linear control system [7]. This characterization facilitates using time-scale ideas in dynamic analysis and performance design.

In summary, our techniques give optimal solutions for graph design problems, which are in essence constrained decentralized control problems. These structure-exploiting designs significantly supersede homogeneous designs in shaping the network dynamics. As we showed in a case study of SARS transmission in Hong Kong, our design provides significant advantage over a homogeneous control strategy, with only 79% of the resources needed. Two particular benefits of our method are 1) the approach provides us with the structural characterization of the eigenvalue/eigenvectors of the optimal solution; 2) the algorithms are quite efficient for certain common network topologies.

1.3 Decentralized Network Control

Decentralized controller design has long been of interest to the controls community, and has been re-invigorated by the extensive need for controlling modern network in such fields as autonomous vehicle control/sensor networking, and systems biology. Research on these modern networks' dynamics has made clear that new decentralized controller designs are badly needed. Also, the network design problems posed in Section 1.2 are in essence *static* decentralized controller design problems, giving further motivation for studying network control. However, in order to obtain better performance, *dynamic* decentralized controller designs need to be explored from the network design viewpoint. However, very little is known about designing high-performance decentralized controllers that are applicable for modern network applications (see [11] for more discussion).

We have been engaged in a major effort to design stabilizing and high-performance yet practical controllers for decentralized systems that is fundamentally based on 1) locally using feedback of multiple derivatives of the observation and 2) using multiple-delay or lead-compensator control schemes to implement these multiple-derivative controllers [10–14]. The philosophy is that by taking derivatives of the observations up to the order of integrator chain, the channels are presented with statistics of the global state, and hence can achieve control. Rather than concentrating the complexity and extent of actuation/observation at a single agent, this approach coordinates actuation/sensing capabilities throughout the network, and hence is suitable for modern applications. Moreover, this new methodology is capable of addressing many complexities common to modern decentralized systems, including very general observation topologies, saturation nonlinearities, and inherent network delays. In [10–12], we introduced the multiple-derivative methodology with delay and lead-compensator implementation using a canonical double-integrator network. We have also examined multiple-delay implementations of multiple-derivative output-feedback controllers in [13–15].

Our efforts show that decentralized controller design requires new tools that deal with structural limitations of decentralization. For instance, system zeros can pose significant difficulties in aspects of decentralized controller design, e.g., for high performance design or in the presence of saturation. Motivated by these needs, we have been engaged in an effort to build new tools. We have devised smart pre-compensators/feed-forward compensators to modify structure/dynamics of Multiple-Input-Multiple-Output (MIMO) Linear Time-Invariant (LTI) systems for zero cancellation and zero relocation [16, 17], and introduced a novel alternative controller design for stabilizing non-minimum phase saturating LTI plants [18].

1.4 Network Numerical Tasks

Given the complexity of modern networks, high-performance designs/controls may sometimes need to be meshed with or verified using numerical analysis and simulation tools, so as to reduce the complexity of control/design. To this end, we have been developing numerical algorithms/tools that are specifically geared toward network applications [19–21]. Here, we introduce two topics that we have worked on: a distributed and flexible network partition algorithm, and a tool for efficient simulation under uncertainty.

Decentralized Network Partition Graph partitioning becomes important in essentially distributed applications—i.e., for swarms of autonomous vehicles, in ad hoc wireless and sensor networks, and for independent/competing players in the power market. Motivated by these applications, we have developed a distributed algorithm for self-partitioning a network that uses a quasi-linear stochastic automaton known as the **influence model** [19]. The automaton-based network partitioning algorithm requires only local computation, and is capable of finding the optimal k-way partition with respect to a broad range of cost functions and given various constraints, in directed and weighted graphs. The decentralized partitioning is very useful for accomplishing complicated network control tasks, by producing *reduced-order* systems for further decentralized control.

Efficient Simulation Many large-scale systems (e.g., power systems, air traffic networks, VLSI circuits, and biological regulatory networks) have complicated parameter-dependent dynamics that can only be examined in detail using time-consuming simulations. When these parameters are uncertain, it is often critical to find the dependence of specific simulation outputs on the parameters. However, because these large-system simulations are usually computationally costly, running exhaustive simulations over the range of potential parameter values is not practical, especially for real-time applications. We have studied polynomial representation of the mapping between uncertain parameters—whose statistics may in general need to be inferred from data or may only be partially known—and outputs from time-intensive simulations, using **probabilistic collocation method** (PCM), which permits polynomial approximation of the mapping between the parameter and output over the *likely* parameter values using only a few simulations [21]. We have given several new analyses concerning the ability of PCM to predict the mapping structure as well as output statistics. We have also developed a holistic methodology for the typical case that the uncertain parameter's probability distribution is unknown, and instead only depictive moments or sample data (which possibly depend on known regressors) are available. Moreover, we studied the application of PCM to weather-uncertainty evaluation in air traffic flow management.

1.5 Thesis Outline

The rest of the thesis is arranged as follows. In Part I (Chapter 1—Chapter 5), we motivate and formulate the design and control problems in several applications, and also give preliminary illustrations of our design and control methods. In Part II (Chapter 6—Chapter 10) and Part III (Chapter 11—13), we throughly develop network design and control methods that exploit network structure. Part IV (Chapter 14-19) is concerned with new studies and tools that support the development of network design and control methods. Finally, Part V (Chapter 20—Chapter 21) studies some network-related numerical tasks.

PART I: MODELING

The primary aim of this thesis is to introduce entirely new methods for design and control in dynamical networks. Given the long history of research in both decentralized controls and network analysis, it is important that we carefully motivate why radically different tools are needed for modern dynamical networks. This first part of the thesis is focused on motivating core design and control problems in modern dynamical networks from several application perspectives. Along with problem formulations in these various application areas, this section gives preliminary illustrations of the design/control methods to be developed systematically later, and highlights the achievements in the application areas that stem from the design/control tools. In this way, we hope to whet the readers' interest in the core methods for dynamical networks introduced later.

Part I is organized as follows. Chapter 2 introduces resource allocation tasks for virus-spreading control in both human-population and computer networks. Chapter 3 and 4 are concerned with modeling network flow management design in air traffic networks. Finally, Chapter 5 develops a network model for distributed sleep regulation in the brain's cortex. We note that sensor networking, autonomous-vehicle coordination, and numerical-methods applications are also pursued in the thesis. However, study of these applications is deeply integrated with the core tool development, and so they are not included in this Part.

2. DESIGNING SPATIALLY-HETEROGENEOUS STRATEGIES FOR CONTROL OF VIRUS SPREAD

The spread of a virus—whether in a human population, computer network, or cell-to-cell—is closely tied to the spatial (graph) topology of the interactions among the possible infectives. In this chapter, we study the problem of allocating limited control resources (e.g., quarantine or recovery resources) in these networks in a way that exploits the topological structure, so as to maximize the speed at which the virus is eliminated. For both multi-group and contact-network models for spread, these problems can be abstracted to a particular decentralized control problem for which the goal is to minimize the dominant eigenvalue of a system matrix. We give explicit solutions to these problems, using eigenvalue sensitivity ideas together with constrained optimization methods employing Lagrange multipliers. Our design method shows that the optimal strategy is to allocate resources so as to equalize the propagation impact of each network component, as best as possible within the constraints on the resource. Finally, we show that this decentralized control approach can provide significant advantage over a homogeneous control strategy, in the context of a model for SARS transmission in Hong Kong.

Tab. 2.1: Notations

Epidemic Model Parameters	
R_0	Basic reproduction ratio
n	No. of groups in a multi-group model; No. of individuals in contact network model
β_{ji}	For multi-group model, the transmission coefficient from District i to j; For contact network model, the probability that virus spreads from node i to j during one time step
$\bar{N}_i$	Population in District i in the multi-group model
$\lambda_i(t)$	Infectiousness of District i
$\bar{\beta}$	Reference transmission coefficient
f_{ji}	Corrective factor specifying transmission coefficient relative to reference
$\bar{f}_{ji}$	Nominal value for f_{ji} (i.e., before control is applied)
r_i	Control variable: scales the transmission coefficients from District i
c_i	Control variable: scales the transmission coefficients into District i
T_i	The average infectiousness duration of an individual in District i
$\bar{T}$	Nominal average infectiousness duration without control
t_i	Control variable: scales the infectious duration in District i
$p_i(t)$	Probability that node i is infected in the contact network model
δ_i	Recovery rate of a note i

Key Design Problem Parameters	
G	Topology matrix
K	Gain matrix
D	Additive gain matrix
λ_{max}, v_{max}, w_{max}	Dominant eigenvalue and its associated right and left eigenvectors

We use * to denote the optimized parameter values.

2.1 Problem Formulation and Motivation

The significant impacts of epidemics in recent years highlight the need for controlling virus spread with limited resources [22,23]. Here, we put forth the perspective that spatially-heterogeneous control strategies enable mitigation of virus spread with sparse resources. Thus, we pose the virus-spreading control problem as a constrained decentralized design task for a dynamic network model, and give an analytical methodology for completing the design task. Our design method leads us to the insight that resources (whether vaccination, rapid detection capability, quarantines, or other resources) should be allocated so as to equalize the propagation impact of each network component, as best as possible within the constraints on the resource. We apply our method to design spatially-heterogeneous controls for Hong Kong's 2003 SARS outbreak (see [24]), which outperform the existing homogeneous controls considered in the literature.

While our primary focus is on virus-spreading control, this work also constitutes a significant contribution to our ongoing efforts in decentralized controller *design* and its applications. Recently, researchers in such fields as autonomous vehicle control and sensor networking have recognized the need for decentralized algorithms/controllers that exploit a network's topological structure (see e.g. [25], see also the review article [26]). Concurrently, systems biologists have recognized the significant role played by a network's graph structure on associated dynamics, in both molecular and population biology, see the review article [27]. Further, in a subset of these efforts, control and/or identification tasks that are based on the network structure are of interest.

From a control-theory standpoint, the *existence* of stabilizing decentralized controllers for networks can be checked using the seminal work of Wang and wang-davison-decentralized-1978 [28], but the *design* of practical but high performance controllers remains difficult (whether by numerical methods or explicit analysis). In [29], we posed an optimal decentralized design problem for a simple class of network dynamics, and developed a design tool using optimization machinery together with eigenvalue sensitivity and graph-algebra notions. Our efforts here show that similar tools can be developed for a family of decentralized design problems including some constrained ones. Our results also highlight that particularly simple and structurally-insightful design tools can be obtained for certain special classes of network topologies, such as ones with non-negative weights.

The chapter is organized as follows. In the remainder of this section, we review two models for computer-virus and biological-virus spread, namely a multi-group model for spatially inhomogeneous populations (Section 2.1.2), and a contact network model for interactions of individuals (Section 2.1.3). In turn, we formulate several virus-spread control problems as decentralized design problems. In Section 2.1.1, we develop a methodology for solving these decentralized design problems in detail. Finally, in Section 2.3, we apply our method to synthesize spatially-heterogeneous controllers for the SARS outbreak in Hong Kong.

2.1.1 Epidemic Control: Brief Review

Mathematical epidemiology has a history of more than two centuries. One major focus of the mathematical work on epidemics is characterization of the basic reproduction ratio R_0 (defined as the average number of secondary infections produced during an infected individual's infectious period, when the individual is introduced into a population where everyone is suspectible [30]). It is well known that for $R_0 > 1$ a disease can spread throughout the population and may eventually persist in equilibrium, while for $R_0 < 1$ the epidemic eventually terminates. The basic reproduction ratio can be computed from models and also found experimentally, see e.g. [24, 30–33].

Control can be viewed as reducing R_0, and hence stopping the spread of a virus. Common control methods include: 1) vaccination; 2) reduction of local contact rates; 3) shortening of the time between symptom appearance and hospitalization including through improved virus detection; 4) restriction on long-range movement; 5) isolation of symptomatic people and those in contact with them (quarantine) [24, 32, 33]. All these control schemes change one or more parameters of the epidemic model of interest. Hence, by analyzing R_0 or simulating dynamics over a parameter range [24, 32], we can analyze the impact of different control schemes. For example, the strategy of vaccinating newborns in a heterogeneous population (assuming contact rates at only two levels) was studied in [30, 34], where an age-related model was used.

In the remainder of this section, we give explicit formulations of control-design tasks in the context of two widely-studied models.

2.1.2 Spatial Control in a Multi-Group Model: Formulation

Spatial interaction structure is critical in epidemics (e.g., SARS [24,35]). However, there is little work in the literature on designing controls (e.g. isolation and quarantine) that are specialized to the network structure, to optimally mitigate epidemic spread with limited resources. We believe such a systematic design of control parameters at different points in the network can provide guidance for effective epidemic control. Here we review spatially inhomogeneous models for epidemics, and so pose the optimal spread control problem.

Early epidemic modeling assumes homogeneous mixing, i.e. any pair of individuals are assumed equally likely to interact (or equivalently the strengths of the interactions are the same). In reality, populations are spatially heterogeneous, and in fact this spatial structure (social interaction topology) of the population plays an important role in epidemic spread. Usually, **multi-group models**, also called **meta-population models** (models in which the population is composed of multiple interacting groups, which internally have homogeneous mixing) are used to represent the spatially inhomogeneous dynamics. Abundant work exists in the literature on constructing and analyzing these multi-group models [24, 32, 36–47], e.g., article [39] establishes the connection of coupling between groups with explicit movement patterns, articles [46, 47] study the effect of aviation traffic on global epidemics, and [24, 32, 41–43] provide case studies of various diseases such as smallpox, measles, SARS and foot-and-mouth disease. Of particular interest to us, references [48–50] show

how to calculate the epidemic threshold in a heterogeneous population. Article [48] calculates the threshold from a reaction-diffusion point of view. In references [49,50], the basic reproduction ratio R_0[*] is shown to be the dominant eigenvalue of the **next generation operator**, the elements of which are defined as the expected numbers of new infections within a group that are produced by one infective with another group during its infectious period. The next generation matrix has been used to calculate R_0 for various applications, see e.g. [24] and [32]. Often, stochastic models for epidemic spread are also used because chance fluctuations can be large, especially in the early stages of an epidemic [24,33].

Here, we study inhomogeneous (distributed) virus-spread control in a multi-group model for spatial propagation, in particular designing controls that optimally reduce R_0. Spatially inhomogeneous models have already been used to study epidemic control, however little effort has been devoted to inhomogeneous control strategies. For instance, the outbreak of severe acute respiratory syndrome (SARS) in 2003 aroused a lot of interest in spatial modeling and control [24,33,51], because of the geographical patterns observed in the virus spread [24,33,35]. Of interest to us, the article [24] models SARS in Hong Kong using a stochastic multiple-group model, where each group corresponds to a (spatial) district in Hong Kong. The authors identify the basic reproduction ratio R_0 for the model, and show how homogeneous (identical network-wide) control can be used to reduce R_0 to 1. We notice that the early work [38] studies an inhomogeneous control strategy for steady-state behavior of persistent epidemics in an open population. Our work builds on [38] and introduces an optimal inhomogeneous control of the full epidemic dynamics. In Section 2.3, we show that our optimal control which exploits network structure can reduce R_0 further with the

[*] R_0 is usually real since an interaction network structure is most often non-negative and irreducible. We limit ourselves to the real maximum eigenvalue case in our later design.

same amount of resource, or equivalently achieve $R_0 = 1$ with less resource.

Let us now formulate the multi-group epidemic model and associated control design problem. Specifically, we will first describe the model in generality, and then consider the particular dependence of model parameters on control actions. We note that the model without controls is very similar to the one in [24], and can be viewed as a multi-group SIS model. We consider a multi-group model with n groups, which we refer to as *districts* since we are primarily interested in the spatial spread of epidemics. We use the notation $\overline{N}_i$ for the population of each district i. The individuals in each district are modeled as transmitting the disease through homogeneous mixing within the district, as well as interaction with individuals in other districts; we model this transmission across groups in an aggregate fashion using an effective coupling (rather than through explicit modeling of individuals' trajectories), see [40, 52–55] for justification. As in the existing literature [24, 52], we find it convenient to define a **reference transmission coefficient** and then to define particular transmission coefficients within and between groups relative to this reference. Specifically, we use the notation $\overline{\beta}$ for the reference transmission coefficient, which identifies a typical rate at which secondary infections would be produced if all contact were from infectives[†] (and incorporates both the effective contact rate and the average infectiousness of an infected individual, see e.g. [24]). Usually, the reference transmission coefficient is chosen as the transmission rate internal to a prototypical district and assuming that no control actions have yet been taken, although other references can equivalently be used. The actual **transmission coefficient** β_{ji} between District i and District j—i.e., the average rate at which infections in District i would be produced by individuals in District j if all contact were from infectives—is given by $\beta_{ji} = f_{ji}\overline{\beta}$, where f_{ji} is a corrective factor

[†] We note that the transmission rate describes secondary-infection production capability for the whole population, rather than per individual.

that specifies the transmission coefficient relative to the reference, and is a parameter amenable to control in our model (as we shall specify in the next paragraph). We stress that f_{ji} and β_{ji} are also defined within a district, i.e. for $i = j$; we note that f_{ii} may differ from 1 in our model, either because of intrinsic differences between the regions or because control actions have been taken. For two different districts ($i \neq j$), f_{ij} captures the relative rate of inhomogeneous mixing as well as any controls that further reduce contact. Finally, the average duration of the infectious period is denoted as T_i, and also is assumed amenable to control. This model can be analyzed by tracking the *infectiousness* $\lambda_i(t)$ of each region i, i.e. the average total infectiousness of the individuals in the region i (see [24] for details). Briefly, it can be shown that the average rate at which each individual in region i becomes infected is given by $\sum_{j=1}^{n} \beta_{ji} \frac{\lambda_j(t)}{\bar{N}_j}$, and in turn that rate of change of $\lambda_i(t)$ is given by $\bar{N}_i T_i \sum_{j=1}^{n} \beta_{ji} \frac{\lambda_j(t)}{\bar{N}_j}$ (assuming that the infected population is small compared to the total population). One thus recovers that the next-generation matrix (see [24, 49, 50]) is:

$$A = \bar{\beta}\ diag(T_i \bar{N}_i) \times \begin{bmatrix} f_{11} & f_{21} & \dots & f_{n1} \\ f_{12} & f_{22} & \dots & f_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ f_{1n} & f_{2n} & \dots & f_{nn} \end{bmatrix} [(diag(\bar{N}_i)]^{-1}. \tag{2.1}$$

Now let us consider applying controls in the context of this model. Specifically, we assume that the corrective factors and infectious period durations have nominal values, namely $\bar{f}_{ji}$ and $\bar{T}$. We consider three sorts of control that deviate from this nominal (see e.g. [56] for motivation of these controls in stopping virus spread): 1) We allow the change of the contact rate of individuals in District i by a factor of $r_i \in [0, 1]$ which decreases the spread of virus both locally and to spatial neighbors, and hence is modeled as scaling the transmission coefficient or equivalently the corrective factor: $f_{ji} = r_i \bar{f}_{ji}$, for all j (including $j = i$). Note that r_i can be reduced by isolation of closely connected and fairly isolated groups such as a school/college, or restriction on public assemblage. 2) we allow change of the contact rate of an individual from outside districts to a district i by a factor of c_i. In this case, $f_{ji} = c_i \bar{f}_{ji}$, for all i, j such that $i \neq j$. The external contact rate factor $c_i \in [0, 1]$ can be reduced by prohibition of travel from another district to District i, or similarly isolation of arriving travelers for some days (i.e. for a period longer than the incubation period). 3) We allow the average duration of the infectious period of each District i or T_i, to deviate from $\bar{T}$ by a factor of t_i. Hence $T_i = t_i \bar{T}$. The factor t_i can be reduced by shortening the time between symptom appearance and hospitalization in District i, e.g.,through faster detection of infected individuals. Control measures such as isolation of people who may have contacted an infected individual (i.e. isolation of a neighborhood with infected people), or isolation of symptomatic people (possibly including some individuals that have false symptoms and are not infected), reduce both T_i and r_i. The next generation matrix upon specification of the control parameters is:

$$A = \bar{\beta}\bar{T}\ diag(t_i r_i \bar{N}_i) \times \left(diag(\bar{f}_{ii}) + diag(c_i) \begin{bmatrix} 0 & \bar{f}_{21} & \dots & \bar{f}_{n1} \\ \bar{f}_{12} & 0 & \dots & \bar{f}_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{f}_{1n} & \bar{f}_{2n} & \dots & 0 \end{bmatrix} \right) [(diag(\bar{N}_i)]^{-1}. \quad (2.2)$$

Let us now formally pose the controller design problem. In doing so, note that one very reasonable performance measure is the dominant eigenvalue of the next generation matrix, which represents the spread rate of the epidemic. Let's say we are interested in designing t_i and/or r_i. Noting that we can write the next generation matrix as $A = KG$, where the **topology matrix** is

$$G = \bar{\beta}\bar{T}\ diag(\bar{f}_{ii}) + \beta\bar{T}\ diag(\bar{N}_i c_i) \begin{bmatrix} 0 & \bar{f}_{21} & \dots & \bar{f}_{n1} \\ \bar{f}_{12} & 0 & \dots & \bar{f}_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{f}_{1n} & \bar{f}_{2n} & \dots & 0 \end{bmatrix} [(diag(\bar{N}_i)]^{-1},$$

and the **gain matrix** is $K = diag(t_i r_i)$, we can view the design problem as that of finding a diagonal matrix K so as to minimize $\lambda_{max}(KG)$ (where $\lambda_{max}()$ denotes the dominant eigenvalue of a matrix), subject to constraints that $0 \le K_i \le 1$ and that the K_i in total exceeds a **lower bound** Γ (since much resource is needed to make K_i small). Here is a formal statement:

Problem 1. Design diagonal matrix K such that $\lambda_{max}(KG)$ is minimized, where K is subject to the following constraints:

1) $tr(K) = \sum K_i \ge \Gamma$

2) $0 \le K_i \le 1$ for all i.

In the case that we restrict long-distance movement (i.e. movement between districts) and so design c_i, we can write the next generation matrix as $A = D + KG$, where

$$G = \bar{\beta}\bar{T}\ diag(t_i r_i \bar{N}_i) \begin{bmatrix} 0 & \bar{f}_{21} & \dots & \bar{f}_{n1} \\ \bar{f}_{12} & 0 & \dots & \bar{f}_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{f}_{1n} & \bar{f}_{2n} & \dots & 0 \end{bmatrix} [(diag(\bar{N}_i)]^{-1},$$

$K = diag(c_i)$ and $D = \bar{\beta}\bar{T}\ diag(t_i r_i \bar{f}_{ii})$.

In this case, we can formulate the design problem as follows:

Problem 2. Design diagonal matrix K such that $\lambda_{max}(D+KG)$ is minimized, where K is subject to the constraints that

1) $tr(K) \geq \Gamma$

2) $0 \leq K_i \leq 1$ for all i.

Remark 1: We may alternately consider other constraints on K. Two realistic ones are listed below. In this chapter, we do not explore these cases in any depth, though our approach of finding an optimal solution can easily be generalized for these cases:

- We may consider a constraint based on resource cost increasing inversely with K_i, i.e. $\sum \frac{1}{K_i} \leq \Gamma$. More generally, any cost that is concave with respect to the K_i can be assumed.
- Often, it is natural that the resource cost also scales with N_i, or otherwise differs from one district to another. Thus we may wish to consider constraints of the form $\sum \alpha_i K_i \leq \Gamma$, where α_i is a district-specific scaling factor.

Remark 2: Our focus here, on minimization of the basic reproductive ratio, is of course only one aspect of epidemic control. One might also consider design of total epidemic size or duration, or pursue design of the steady-state in an open system. Some of these problems are amenable to similar analysis, see e.g. our efforts in [2, 5].

2.1.3 Inhomogeneous Control in a Contact Network Model

Contact network models (also known as **agent based models** or **automaton models**) for virus spread—those in which *individuals'* infection states (or state probabilities) are tracked—have been used to model cell-to-cell spread of influenza [57], SARS propagation [58], and computer virus spread [59], among other applications. Contact network models are motivated by the observation that homogeneity usually does not exist in real populations, perhaps not even within small groups.

Contact network models are thus appealing in that they can capture the specific network interactions among individuals. It is worth noting that, within the general framework of contact network models, considerable research is focused on special classes of network topologies (e.g., scale-free, small-world, correlated, mesh) [22, 60–64].

A contact network model defined for a general network topology was proposed in [65] (which is motivated by the computer virus application). This chapter approximated the epidemic threshold (a threshold on the infection rate to curing rate ratio, above which epidemic occurs, i.e. such that $R_0 = 1$) as the inverse of the dominant eigenvalue of the network's *adjacency matrix*, assuming that (at each discrete time step) an infected node infects its adjacent node with a common probability and is cured with a different common probability. However, because the interaction probabilities are identical, this chapter does not provide us with insight into topology-based network design, which can potentially lower the network's vulnerability to virus spread. Our work here seeks topology-exploiting designs, e.g. problems of where in the contact network to place limited control resources.

Few works have studied heterogeneous network resource allocation for stopping virus-spread. The article [22] proposed a targeted immunization strategy (a few nodes with the highest connectivity are immunized) for power-law networks, and evaluated its performance using simulation. The article [23] concluded that selective immunization (e.g. immunizing the upper-level nodes in a tree-like topology, or nodes with high connectivity) significantly reduces a network's vulnerability to virus attack compared to random immunization. However, this work is also built on simulation, and hence does not provide us with an immunization strategy that meets a performance requirement, or that must operate under particular rigid constraints.

We develop network resource allocation strategies that optimize spread-based performance requirements (e.g., epidemic diminishing rate, number of nodes affected, and the total duration of the epidemic), with the motivation that such design will aid in defending networks against virus attacks. In our effort, the network parameters (e.g., the local curing rates and infection rates) from [65] have the flexibility of design. For example, providing a selected set of individuals/nodes with faster detection capabilities and treatment (or, in the case of computer viruses, better virus scan softwares) can increase these nodes' local curing rates. Similarly, providing antibiotics to individuals (equivalently, providing computers with strong firewalls) can safeguard these nodes from common viruses, or at least lower the rate of infection from their neighboring nodes. Each of these control actions is associated with a cost (e.g., financial cost, productivity loss). Thus, it is not realistic to immunize or provide real-time repair to every individual in a network. Instead, we must assign limited control resources to achieve the best performance. Our study indicates how resources can be allocated in a way that appropriately uses the network topology.

To pursue control, we build on the contact network model proposed in [65], with the motivation that this model has already been of interest in studying resource allocation. Our model is a generalization of [65], in that we allow variation in local curing rates and infection rates throughout the network. As in [65], we model virus spread as a discrete-time dynamics defined on a directed graph. Each node ($i \in 1, ..., n$) represents an individual (node) in the network, which may either be infected or susceptible. Each directed edge represents a path along which a virus can spread from one node to another. The branch weight β_{ij} represents the probability that virus originating from node i spreads to node j during a time step. Notice that β_{ij} increases with the transmission rate from node i to j and the infectiousness of the virus, and decreases as protection at node j is increased (e.g., through giving antibiotics to humans, or providing firewalls to computers). We set $\beta_{ij} = 0$ if node j is not a neighbor to i (node i cannot transmit an infection to j) or node j's protection prevents any infection. An infected node has probability δ_i to recover at a discrete time step.

We consider two possible control actions in our model. 1) We allow for control that makes a node j less susceptible to any virus spread. In this case, we assume that the nominal weights are scaled by a constant for all entering branches, i.e. the weights become $K_j\beta_{ij}$, $K_j \in [0,1]$. We note that decreasing K_j from 1 is costly. 2) We allow control of the recovery rate δ_i. We note that increasing the recovery rate is expensive, in that more medicine or quicker hospitalization is needed (better virus removal programs or quicker human intervention, respectively, for computer network applications).

Now let us analyze the network's dynamics. Denoting the probability that each node i is infected at time t as $p_i[t]$, we find that the probability the node is infected at time $t+1$ is

$$p_i[t+1] = \left(1 - \prod_{\forall j}(1 - K_i\beta_{ji}p_j[t])\right) + (1-\delta_i)p_i[t]. \tag{2.3}$$

Assuming $K_i\beta_{ji}p_j[t] << 1 \ \forall i,j,t$ (which is accurate for small time steps), the quantity $1 - \prod_{\forall j}(1 - K_i\beta_{ji}p_j[t])$ can be well approximated by $\sum_{\forall j} K_i\beta_{ji}p_j[t]$, and thus we can linearize (2.3) to obtain the network dynamics

$$P[t+1] = (D + KG)P[t] \tag{2.4}$$

where topology matrix $G = \begin{bmatrix} 0 & \beta_{21} & \dots & \beta_{n1} \\ \beta_{12} & 0 & \dots & \beta_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{1n} & \beta_{2n} & \dots & 0 \end{bmatrix}$, additive gain matrix $D = diag(1-\delta_i)$, gain matrix $K = diag(K_i)$, and $P[t] = [p_1[t] \ p_2[t] \ \dots \ p_n[t]]^T$. In the case where only δ_i are being designed, we find it convenient to use the notation that $P[t+1] = (D+G)P[t]$, since the K_i are fixed.

The diagonal matrices D and K are the ones we have ability to design. Our aim is to design D and K so that the network structure best inhibits virus spread, and hence minimizes epidemic size. This goal leads us to consider optimization with respect to a performance measure. The one we consider here is the dominant eigenvalue of $D+KG$ (denoted as $\lambda_{max}(D+KG)$), as we know that the dominant eigenvalue governs the growth/decay rate of infection.

The matrix D contains the local recovery rate of each node. Increasing δ_i (or equivalently decreasing D_i) can speed up the elimination of a virus, but at higher resource cost. Therefore, we aim to design D so that the cost of control (i.e., sum of δ_i) is under a limit, while the performance of the design is optimized. Similarly, matrix K represents the virus protection strength for each node. Decreasing K_i for more nodes can also speed up the elimination of a virus. We thus design K so that the cost of control is less than a threshold (i.e., sum of K_i is over a limit), while the performance of the design is optimized.

Let us pose these network design problems formally:

Problem 3. Design diagonal matrix D, such that $\lambda_{max}(D+G)$ is minimized, while D satisfies the constraints:

1) $tr(D) \geq \Gamma$

2) $0 \leq D_i \leq 1$ for all i.

Problem 4. Design diagonal matrix K, such that $\lambda_{max}(D+KG)$ is minimized, while K is satisfies the constraints:

1) $tr(K) \geq \Gamma$

2) $0 \leq K_i \leq 1$ for all i.

2.2 Network Design

In this section, we address the design problems formulated in Section 2.1, namely to design diagonal D to minimize $\lambda_{max}(D+G)$, and to design diagonal K to minimize $\lambda_{max}(KG)$ and $\lambda_{max}(D+KG)$, subject to the described constraints. We give methods for finding the optimal resource allocations both for general topologies G, and for specific classes of topologies that are common in virus-spreading applications. Our methods turn out to be deeply connected to on-going research on the design of high-performance decentralized controllers for modern networks, so we begin by briefly discussing the connection. We then describe the solution to the $D+G$ case, since the full suite of results is easier to describe/interpret in this case. We further present the design for the design of $\lambda_{max}(KG)$ and $\lambda_{max}(D+KG)$, and finally briefly highlight some computational and implementation-related concerns.

2.2.1 Connection to Decentralized Control

Let us begin by explaining why the three design problems listed above are canonical decentralized controller design problems. Decentralized control—i.e., the task of controlling a system with many components each of which has only partial ability to regulate the global dynamics—has been of interest for many years (e.g., [28, 66, 67]), and has application in such diverse fields as electric power system control and robotics. The decentralized control of systems comprising simple but highly limited agents that cooperate through sensing/communication has gained especial prominence in recent years, as networks have become ubiquitous and increasingly coordination of the network components is critical to achieving desired tasks [25, 26, 28, 29, 68, 69]. Our problems fit within this paradigm for control of modern networks: the control actions in a region or for an individual only directly impact the epidemic spread locally and are constrained (mathematically, only alter the system matrix through multiplication/addition of a constrained *diagonal matrix*) and yet must be used in coordination to stop spread globally. We stress that we have not taken a decentralized-control perspective simply for the sake of convenience of analysis or implementation: the deisgn problems that we address are in their essence decentralied, in that control actions (e.g., vaccination) in regions only react to and impact local infected populations; this fundamental constraint is reflected in the diagonal structure of the designable matrices D and K.

For modern network tasks such as ours, understanding the role played by the graph topology in permitting stabilization and high-performance control has been of particular interest (e.g., [25, 26, 29]). This graph-theoretic viewpoint has highlighted that, although the existence of *stabilizing* decentralized controllers can be checked (see the seminal work of Davison and Wang [28]), the problem of designing high-performance decentralized controllers remains difficult by any means. Our efforts here contribute to understanding of high-performance decentralized controller design (see also [29, 69]).

More formally, the decentralized controller design problem is the following: for a network with n channels or agents or components, controllers (rules for determining channel inputs or actuations from observations) must be developed, so that the entire network's evolution meets performance and/or robustness requirements. While a variety of controller forms and performance requirements are of interest, linear control schemes subject to constraints are especially common, and many performance measures are based on the modes (eigenvalues) governing the dynamics. In such cases, the controller design problem can be abstracted to a linear-algebraic design problem in which—due to decentralization—the design parameters are contained in diagonal or block-diagonal matrices. We notice that the virus-spreading control problems introduced in Section 2.2 take this form, in that a linear but constrained decentralized mechanism is used to optimize an eigenvalue-based cost measure; as expected, this problem abstracts to a linear-algebraic one concerned with designing a diagonal gain matrix. It is worth noting that the particular linear algebraic design problems posed here arise in a range of applications, including in autonomous-vehicle control and numerical computation, among others [25, 26, 29, 68].

2.2.2 Designing $\lambda_{max}(D+G)$

We address the problem of designing diagonal D such that $D+G$ has minimum dominant eigenvalue, subject to the constraints that $0 \leq D_i \leq L$ and $tr(D)$ is lower-bounded. For convenience, we refer to an optimum D as D^*, the dominant eigenvalue of D^*+G as λ^*_{max}, and the corresponding left- and right- eigenvectors as w^*_{max} and v^*_{max}. Our design method is founded on the observation that an optimized topology D^*+G has a very special eigenstructure, based on which we can compute D^* and find the optimal performance.

We begin with the structural result:

Theorem 2.1. *Consider a matrix $D+G$, where D is diagonal and G is an $n \times n$ matrix. Consider any $D = D^*$ that minimizes the dominant eigenvalue of $D+G$ subject to the constraints 1) $\sum D_i \geq \Gamma$ and 2) $D_i \in [0, L]$, and assuming the dominant eigenvalue of $D+G$ is real and non-repeated*[‡]*. The optimizing D^* and corresponding eigenvalue/eigenvectors λ^*_{max}, w^*_{max} and v^*_{max} (properly normalized) satisfy one of the following two conditions:*

[‡] The theorem can be easily generalized to the case that $D+G$ has real, simple dominant eigenvalues.

1) $\sum D_i^* = \Gamma$. *In this case, for each* i *we either have* $0 < D_i^* < L$ *and* $w^*_{max_i} v^*_{max_i} = 1$*, or we have* $D_i^* = L$ *or* $D_i^* = 0$.

2) $\sum D_i^* > \Gamma$. *In this case, for each* i *we either have* $0 < D_i^* < L$ *and* $w^*_{max_i} v^*_{max_i} = 0$*, or we have* $D_i^* = L$ *or* $D_i^* = 0$.

Proof: This result follows from standard theorems on eigenvalue sensitivity [70], as well as theorems on constrained optimization using Lagrange multipliers [71]. Let us denote $D_i = d_i^2$, since we require $D_i \geq 0$ for all i. The procedure for finding an optimum D^* under constraint is to form the Lagrangian $L = \lambda_{max}(D + G) + \sum a_i(d_i^2 + m_i^2 - L) - C(\sum d_i^2 - n^2 - \Gamma)$ and set the derivatives of it with respect to all variables (namely d_i, a_i, m_i, C and n_i) to 0. (Here, m_i and n are slack variables to transform inequality constraints to equality constraints). This procedure leads to the equations below:

$$\begin{aligned} & d_i^*\left(\frac{w^*_{max_i} v^*_{max_i}}{w^T v} + a_i^* - C^*\right) = 0 \qquad (2.5) \\ & d_i^{*2} + m_i^{*2} = L \\ & m_i^* a_i^* = 0 \\ & n^* C^* = 0 \\ & \Gamma + n^{*2} = \sum d_i^{*2} \end{aligned}$$

Note that the first equation above follows from the eigenvalue sensitivity formula. The two cases in the theorem thus follow automatically from consideration of the variables n^* and C^*, one of which must be 0. Specifically, the case where $\sum D_i^* = \Gamma$ follows from stetting n^* to 0, while the case where $\sum D_i^* > \Gamma$ follows from setting C^* to 0. □

Theorem 2.1 tells us that an optimum D^* can be either at or inside the constraint boundaries.

When D^* is at the boundary $\sum D_i^* = \Gamma$, each D_i^* falls into one of the three categories: 1) $D_i^* = L$; 2) $D_i^* = 0$; and 3) $w_{max,i}^* v_{max,i}^* = 1$. When D^* is not at the boundary $\sum D_i^* = \Gamma$, each D_i^* again is at 0 or L, or $w_{max,i}^* v_{max,i}^* = 0$. We notice that, for any condition in one of the above forms, the number of equations and variables are equal, and so we can get a possible optimal solution from these equations. Thus, we see that a finite search can in theory be used to find the optimal solution, among these possibilities. However, the number of possible optimal solutions (number of solutions which satisfy one set of equations of this type) grows exponentially with the dimension of the matrix G, and so the calculation will be very complicated for even moderate-sized G. In the rest of this section, we will show that when G is specially structured—e.g., non-negative or symmetric, we can develop more explicit (and hence easier-to-evaluate and interpret) expressions for an optimal solution.

In the following Theorem 2.2, we show that for a very broad class of topology matrices G, an optimizing D is one that uses maximum total resource, i.e. one for which $\sum_i D_i = \Gamma$.

Theorem 2.2. *Consider $D + G$, where D is diagonal, and G is an $n \times n$ matrix. Assume that the largest eigenvalue of $D + G$ is real and non-repeated for all D such that 1) $\sum D_i \geq \Gamma$ and 2) $D_i \in [0, L]$. Any matrix D^* that minimizes the dominant eigenvalue of $D + G$ subject to these constraints satisfies $\sum D_i^* = \Gamma$, if the left and right eigenvectors of $D + G$ corresponding to the dominant eigenvalue have the same sign patterns for all D. Classes of matrices satisfying this condition include 1) irreducible[§] non-negative matrices and 2) diagonally symmetrizable matrices*

[§] To ease the understanding of those readers not familiar with matrix theory, we note that irreducible matrix is one that can not be transformed into block upper-triangular matrix by simultaneous row/column permutations. The associated digraph of an irreducible matrix is strongly connected, e.g., a path exists between any two nodes in the digraph.

(matrices for which there exists diagonal Q such that $Q^{-1}GQ$ is symmetric).

Proof: The condition that the left and right eigenvectors of the dominant eigenvalue have the same sign pattern implies the relationship that, for all i, $w_{max,i}v_{max,i} > 0$. Thus, according to the eigenvalue sensitivity theorem, the dominant eigenvalue decreases monotonically with the decrease of D_i for all i, since $\frac{\partial \lambda_{max}(D+G)}{\partial D_i} = w_{max,i}v_{max,i}$ is positive. Therefore, an optimum D^* is on the boundary $\sum D_i^* = \Gamma$. From the Perron Frobenius Theorem, the dominant eigenvalue of any non-negative and irreducible matrix is real and non-repeated, and the left and right eigenvectors associated with the dominant eigenvalue are positive [72], and hence have the same sign pattern. For a diagonally symmetrizable G, it is easy to check through a similarity transform that the eigenvalues are real and the left and right eigenvectors associated with any eigenvalue are identical related by a positive diagonal scaling and hence have the same sign pattern. Hence, the theorem is proved. $\square$

Theorem 2.2 guarantees an optimum D^* is located on the boundary $\sum D_i^* = \Gamma$, whenever G is an irreducible and non-negative square matrix, and hence simplifies the search for D^* when G has this special structure. This simplification is relevant to our applications, because for both the multi-group and contact network models, G is non-negative and (for meaningful interaction topologies) irreducible. In the illustrating example, G is irreducible and non-negative, so we expect that an optimum D^* satisfies $\sum D^* = \Gamma$. In this case, D^* can be found by searching only through the first set of possibilities in Theorem 2.1. This search is formulated in Theorem 2.4 for some matrices of this sort. Before that, in Lemma 2.3, we characterize the pattern of the eigenvector associated with the dominant eigenvector under constraints $D_i \in [0, L]$ and $\sum D_i \geq \Gamma$.

Lemma 2.3. *Consider the matrix $D+G$, where D is diagonal, and G is an $n \times n$ irreducible non-negative symmetric matrix. Any matrix $D = D^*$ minimizes the dominant eigenvalue of $D+G$ subject to the constraints $\sum D_i \geq \Gamma$ and $D_i \in [0, L]$ if and only if the components of the eigenvector v^*_{max} associated with the dominant eigenvalue of D^*+G has a special pattern. In particular, it is required that $v^*_{max,i} < v^*_{max,j} < v^*_{max,l}$, for any i, j, k such that $D_i = L$, $0 < D_j < L$ and $D_l = 0$, and $v^*_{max,j}$ are identical for all j.*

Proof: First let us show necessity. Suppose v^*_{max} is the eigenvector associated with the dominant eigenvalue of D^*+G. The conclusion that $v^*_{max,i(\forall\ i,\ s.t.\ 0<D_i<L)}$ (where we have used the subscript notation to refer to the entries of v^*_{max} for which $0 < D_i < L$) are the same directly follows from Theorem 2.1, and the symmetry and non-negativity of G. More specifically, when G is a symmetric matrix, so is $D+G$, and thus $v_{max,i} = w_{max,i}$ for all i. Also, from Theorem 2.1, we know that the eigenvectors v^*_{max} and w^*_{max} of D^*+G satisfy $v^*_{max,i} = w^*_{max,i}$ and $v^*_{max,i}w^*_{max,i} = 1$ for each i such that $0 < D^*_i < L$. Finally, from positivity of G, we see that v_{max} has only positive entries. Combining, we recover that $v^*_{max,i(\forall\ i,\ s.t.\ 0<D_i<L)}$ are identical. Now we need to show that $v^*_{max,i(\forall\ i,\ s.t.\ D_i=L)} < v^*_{max,i(\forall\ i,\ s.t.\ 0<D_i<L)} < v^*_{max,i(\forall\ i,\ s.t.\ D_i=0)}$. Since D^* achieves the minimum dominant eigenvalue, decreasing D^*_i for i such that $D^*_i = L$ (making them less than L) or increasing D^*_i for i such that $D^*_i = 0$ (making them larger than 0) while maintaining $\sum D_i = \Gamma$ should increase λ_{max}. Eigenvalue sensitivity naturally leads to the inequality of eigenvector components, since the derivative of λ_{max} with respect to D_i equals $v^2_{max,i}$.

For the sufficient condition, we know that if $v_{max,i(\forall\ i,\ s.t.\ D_i=L)} < v_{max,i(\forall\ i,\ s.t.\ 0<D_i<L)} < v_{max,i(\forall\ i,\ s.t.\ D_i=0)}$ and $v_{max,i(\forall\ i,\ s.t.\ 0<D_i<L)}$ are the same, the corresponding D achieves a local minimum (from above). It follows from convexity (which can be proved easily using e.g. the Courant-Fisher theorem) that the local minimum is in fact global. □

Lemma 2.3 presents the pattern of the eigenvectors associated with the minimized dominant eigenvalue of $D+G$. This allows us to check whether a solution D is optimum by simply evaluating the dominant eigenvectors of $D+G$. Here we define the expression *diagonalize(*$\mathbf{v}$*)* as placing the ith entry of a vector $\mathbf{v}$ as the ith diagonal entry in a diagonal matrix, for $i = 1, ..., n$.

Theorem 2.4. *Consider a topology matrix G that is non-negative, irreducible, and diagonally symmetrizable. A matrix $D = D^*$ that minimizes the dominant eigenvalue of $D+G$ can be found using the following algorithm:*

1) *Find diagonal matrix Q such that $Q^{-1}GQ$ is symmetric*[¶]*. We denote $Q^{-1}GQ$ as $\hat{G}$.*

2) *Choose some gains D_i as 0 and some other gains D_i as 1, and rearrange the rows and columns of $D+\hat{G}$ in such a way that those D_i fixed at 0 and 1 are in the lower right corner. The permuted matrix can be decomposed as $\begin{bmatrix} \hat{G}_{11}+D_A & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22}+D_B \end{bmatrix}$, where the matrix D_A is unknown, and the matrix D_B has entries fixed at 0 or 1.*

3) *Solve the equation $-\mathbf{1}^T\hat{G}_{11}\mathbf{1} - \mathbf{1}^T\hat{G}_{12}(\lambda I - \hat{G}_{22} - D_B)^{-1}\hat{G}_{21}\mathbf{1} + \mathbf{1}^T\lambda\mathbf{1} = \Gamma - tr(D_B)$ for λ. This can be done through a simple numerical procedure, such as a gradient search*[‖]*.*

4) *Calculate D_A using $D_A = diagonalize(-\hat{G}_{11}\mathbf{1} - \hat{G}_{12}(\lambda I - \hat{G}_{22} - D_B)^{-1}\hat{G}_{21}\mathbf{1} + \lambda\mathbf{1})$. If $0 \le D_A \le LI$, and the pattern of eigenvector associated with the dominant eigenvalue of*

[¶] See [74] for a method for checking if a matrix is symmetrizable and if so how the diagonal matrix can be chosen to achieve symmetry.

[‖] This is a simple procedure because the optimization is with respect to a single variable. Standard optimization tools can solve this problem in minimal time.

$D+\hat{G}$ follows Lemma 2.3, an optimum D is diag(D_A, D_B). Otherwise, go to step 2) until a solution is achieved.

Proof: We know from similarity that the eigenvalues of $D+G$ are same as that of $D+\hat{G}$, where $\hat{G}=Q^{-1}GQ$ and Q is the positive diagonal matrix such that $\hat{G}$ is symmetric. Hence, we can without loss of generality consider designing D^* to minimize $D^*+\hat{G}$. Since all v_{max_i} (of D^*+G) whose corresponding D_i^* are not fixed at 0 or 1 are equal (let us normalize them to 1), we know an optimum D satisfies $\begin{bmatrix} \hat{G}_{11}+D_A & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22}+D_B \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{1} \\ \mathbf{v} \end{bmatrix}$, where D_B contains the entries in D that are 0 and 1. D_A must be found, and $\hat{G}_{11}$, $\hat{G}_{12}$, $\hat{G}_{21}$, and $\hat{G}_{22}$ are submatrices of $\hat{G}$ (upon appropriate permutation). A little bit of algebra leads to the solution of λ, D and v. If the pattern of the eigenvector associated with the dominant eigenvalue of $D+\hat{G}$ follows Lemma 2.3, and the D matrix is within the constraints $0 \leq D \leq LI$, we have found a global optimum solution according to Lemma 2.3. $\square$

Although we have presented an algorithm for diagonally-symmetrizable G, a slightly more complicated algorithm exists for all G that are non-negative matrices; we omit this algorithm here in the interest of space. We also note that the above algorithm may be computationally intensive, in that the steps may have to be repeated up to 3^n times to find the optimum. For some topology matrices G, the calculation of D^* can further be simplified. We will describe the procedure to calculate D^* under these circumstances in Theorem 2.6. As a preliminary step, let us first explicitly compute an optimal D when the constraints on individual D_i are relaxed:

Lemma 2.5. *Consider the matrix $D+G$, where D is diagonal, and G is an $n \times n$ symmetric and non-negative irreducible matrix. $\bar{D}$ that minimizes the dominant eigenvalue of $D+G$ subject only to the constraint $\sum D_i \geq \Gamma$ can be found as follows: First find $\bar{\lambda}_{max} = \frac{1}{n}(\Gamma + \sum_i \sum_j G_{ij})$ and then find $\bar{D}_i = \bar{\lambda}_{max} - \sum_j (G_{ij})$.*

Proof: First, we notice all entries of $\bar{v}_{max}$ are identical, based on Lemma 2.3 and the fact that here only the constraint $\sum D_i \geq \Gamma$ is enforced. Thus, an optimizing $\bar{D}$ and eigenvalue satisfy $(\bar{D}+G)\mathbf{1} = \bar{\lambda}_{max}\mathbf{1}$. Therefore, the $\bar{D}_i$'s make the row sums of $D+G$ equal. $\bar{D}_i$ also satisfies $\sum \bar{D}_i = \Gamma$, since G and thus $\bar{D}+G$ is symmetric. A little bit of algebra leads to the expressions for $\bar{\lambda}_{max}$ and $\bar{D}_i$. $\square$

Lemma 2.5 states that, without the constraints that $D_i \in [0, L]$, an optimum D (denoted $\bar{D}$) is the one that equalizes the row sum of $D+G$, i.e. resource is allocated to each part of the network so as to make all their impacts identical. However, when the individual D_i are constrained, sometimes an optimum $\bar{D}$ can not be reached. Building on Lemma 2.5, Theorem 2.6 illustrates an easy way to find an optimum D under several circumstances.

Theorem 2.6. *Consider a matrix $D+G$, where D is diagonal, and G is an $n \times n$ non-negative, irreducible, and diagonally-symmetrizable matrix. We can find $D = D^*$ that minimizes the dominant eigenvalue of $D+G$ subject to the constraints 1) $\sum D_i \geq \Gamma$ and 2) $D_i \in [0, L]$ using the following algorithm:*

1) Find diagonal Q such that $Q^{-1}GQ$ is symmetric. We name $Q^{-1}GQ$ as $\hat{G}$.

2) Calculate $\bar{\lambda}$ and $\bar{D}_i$ following Lemma 2.5, and check if $0 \leq \bar{D}_i \leq L$ for all i. Denote the set *of indices i such that $\bar{D}_i > L$ as $\mathcal{L}^+$ and the set such that $\bar{D}_i < 0$ as $\mathcal{L}^-$. If $\mathcal{L}^+ = \phi$ and $\mathcal{L}^- = \phi$ (i.e., both sets are empty), we have $\lambda^*_{max} = \bar{\lambda}$ and $D^*_i = \bar{D}_i$.*

3) *If $\mathcal{L}^+ \neq \phi$, and $\mathcal{L}^- = \phi$, we set $D_i = L$ for $i \in \mathcal{L}^+$. That is, D_i can be calculated by first applying: $D_i = L$ for $i \in \mathcal{L}^+$, $v_{1_i} = 1$ for $i \notin \mathcal{L}^+$, and $\sum_i D_i = \Gamma$, and solving the above equations for D_i as in Theorem 3. If $D_i \leq L$ $\forall i$, we have $D^* = D$. Otherwise, we must recursively reduce those D_i such that $D_i > L$ to L, and recompute D until all $D_i \leq L$.*

4) *If $\mathcal{L}^- \neq \phi$, and $\mathcal{L}^+ = \phi$, we set $D_i = 0$ for $i \in \mathcal{L}^-$. That is, D_i can be calculated by first applying: $D_i = 0$ for $i \in \mathcal{L}^-$, $v_{1_i} = 1$ for $i \notin \mathcal{L}^-$, and $\sum_i D_i = \Gamma$, and solving the above equations for D_i as in Theorem 3. If $D_i \geq 0$ $\forall i$, we have $D^* = D$. Otherwise, we must recursively increase those D_i such that $D_i < 0$ to 0, and recompute D_i until all $D_i \geq 0$.*

Proof: This theorem consider three cases. The simplest case is when $\mathcal{L}^+ = \phi$, and $\mathcal{L}^- = \phi$. In this case, obviously $D^* = \bar{D}$ is an optimum solution, as shown in Lemma 2.5. The cases when one of $\mathcal{L}^+, \mathcal{L}^-$ is ϕ and the other is not are similar, hence we only consider the case that $\mathcal{L}^- = \phi$, and $\mathcal{L}^+ \neq \phi$. In this case, an optimal solution has the special feature that $D^*_i = L$, for $i \in \mathcal{L}^+$. This proof is complicated, however the idea is as follows.

Let's say an unconstrained optimum ($\bar{D}$) yields m elements in $\mathcal{L}^+$ (m of the $\bar{D}_i$ are greater than 1), and let's consider an optimal solution when these $\bar{D}_i$ are constrained to L (while the other entries are left unconstrained for now). We will prove that the eigenvector components associated with these m elements are smaller than the eigenvector components associated with all the other elements (e.g. $\hat{v}_{max,i \in \mathcal{L}^+} < \hat{v}_{max,i \notin \mathcal{L}^+}$), and hence prove that this optimum is in fact a global one according to Lemma 2.3. We show this using induction. Moreover, if the solution obtained by fixing $D_i = L$ for $i \in \mathcal{L}^+$ still has some $D_i > L$, we will repeat the above process to suppress those $D_i > L$ at L.

Basis: Let's use $Z = \bar{D} + \hat{G}$ for the matrix that produces the unconstrained global optimal, call its dominant eigenvalue $\bar{\lambda}$, and assume (wlog) that the first m entries of D are too big (e.g., larger then L). Let's us first consider decreasing the gain in entry 1 to unity while optimizing the other gains (right now, we don't apply constraints to these gains). Here we denote D_i as gain in entry i. Then the matrix of interest changes from Z to $Z + \begin{bmatrix} -\alpha & & & \\ & \Delta D_2 & & \\ & & \ddots & \\ & & & \Delta D_n \end{bmatrix}$, where $\alpha > 0$ and $\sum_{i=2}^{n} \Delta D_i = \alpha$. We would like to prove two statements: first, the eigenvector component corresponding to entry 1 is less than the components associated with all the other entries (e.g. $v_{max,1} < v_{max,i\neq 1}$); and second, the gain corresponding to entries from 2 to m are still at least L.

Recalling Theorem 2.1, we know that all $v_{max,i\neq 1}$ are the same, say c_1. For convenience, let us call $v_{max,1}$ as v_1. Hence, the eigenvector associated with the dominant eigenvalue (denoted by $\hat{\lambda}$) for the new matrix is $\hat{v} = [v_1 \; c_1 \; \ldots \; c_1]$. The Courant-Fischer theorem [73] tells us that $\bar{\lambda} = max_v v^T Z v$ and $\widehat{\lambda} = max_v v^T (Z + \Delta) v$. $\widehat{\lambda}$ can be written as $\widehat{\lambda} = max_{v_1, c_1} (\hat{v}^T Z \hat{v} - v_1^2 \alpha + c_1^2 \alpha)$. When we decrease the gain in entry 1, we know that maximum eigenvalue of the new matrix, say $\widehat{\lambda}$, is larger than $\bar{\lambda}$. And we also know $\hat{v}^T Z \hat{v} < \bar{\lambda}$, since $\hat{v}$ is not the eigenvector for $\bar{\lambda}$. Thus, from the expression for $\widehat{\lambda}$, the first entry in the eigenvector for the new matrix is smaller than the rest of the entries (i.e. $v_1 < c_1$). Thus the first statement is proved.

For the second statement, we prove it as follows. Since $v_1 < c_1$, the eigenvector associated with the optimal gain changes from $\mathbf{1}_n$ to $\mathbf{1}_n + \begin{bmatrix} v \\ \\ c\mathbf{1}_{n-1} \end{bmatrix}$, where we know $v < 0 < c$ (since only one gain has been moved so far). Further, we know that maximum eigenvalue of the new matrix, say $\widehat{\lambda}$, is larger than λ. Plugging into the eigenvector equation and doing some algebra, we finally get the following: $(\widehat{\lambda} - \lambda)\mathbf{1}_{n-1} + c(\widehat{\lambda}\mathbf{1}_{n-1} - Z_{2:n,2:n}\mathbf{1} - Z_{2:n,1}\frac{v}{c}) = (1+c)\begin{bmatrix} \Delta D_2 \\ \vdots \\ \Delta D_n \end{bmatrix}$. However, since $\widehat{\lambda} > \lambda$, $v < 0 < c$ and Z is a non-negative matrix, we recover that all entries in the expression on the left are positive, and hence the change in gains $\begin{bmatrix} \Delta D_2 \\ \vdots \\ \Delta D_n \end{bmatrix}$ must be all positive and so each other gain strictly increases. Thus, we see that the gain corresponding to entries from 2 to m are still at least L.

Induction: Suppose that we bring any l of the $D_i > L$ to L ($l < m$). Let us assume first that the eigenvector components corresponding to the l entries are less than the components associated with all the other entries (e.g. $v_{max,1,\ldots,l} < v_{max,i>l}$); and second, that the gain corresponding to entries other than these l ones are still at least L. Let us show that after we bring another (the $l + 1^{st}$) D_i to L, we still have the appropriate eigenvector component majorization and condition on the gains.

Let us consider bringing $l + 1$ of the $\bar{D}_i$ from the unconstrained optimum s.t. $i \in \mathcal{L}^+$ to L. We can do this using two steps. The first step is to move all but one of the offending gains to L (l gains), and the second step is to bring the last gain to L. First note that this is possible, since after the first step, all other gains remain greater than L by assumption. Without loss of generality, for notational convenience, let us assume that we first bring the first l $\bar{D}_i$ s.t. $i \in \mathcal{L}^+$ to L, and then move $\bar{D}_{l+1}$. Denote the dominant eigenvalue after the first step as λ_1 and the one that after bringing the last gain to L as λ_2. Again applying the Courant-Fischer theorem, $\lambda_1 = max_v v^T(D + \hat{G})v$ and $\lambda_2 = max_v v^T(D + \hat{G} + \Delta)v$, where diagonal matrix Δ corresponding to changing $\bar{D}_{i+1}$ to L have entries as the following: $\Delta_{l+1,l+1} = -b$ $(b > 0)$, $\Delta_{i,i} = 0$ for $i \in [1, ..., l]$ and $\Delta_i = \Delta D_i$ for $i \in [l + 2, ..., n]$. We also have $\sum_{i=l+2}^{n} \Delta D_i = b$. With a similar argument to that given in the basis argument, we can show that the $(l + 1)^{st}$ eigenvector component is less than the (identical) components after position $l+1$. Repeating this argument with each $\bar{D}_i$ out of the $l+1$ possibilities set to L last, we can reach the conclusion that the eigenvector components corresponding to all the $l + 1$ entries are less than the rest common entries.

The proof that the remaining gains D_i increase (and hence that they remain larger than L if they were originally larger than L without the constraints) after bringing these $l + 1$ gains D_i to L is based on the knowledge that the eigenvector components corresponding to all the $l+1$ entries are less than the remaining (identical) entries. This can be proved formally in a very similar fashion to the case where a single gain is moved, which we have addressed in the basis step of the induction. Thus the details are omitted.

In case some other gains exceed L in the process, these can be reduced in the same fashion.

For the case that $\mathcal{L}^- \neq \phi$, and $\mathcal{L}^+ = \phi$, the proof is analogous to the case $\mathcal{L}^- = \phi$, and $\mathcal{L}^+ \neq \phi$ that we have proved here, and hence it is omitted. $\square$

Theorem 2.6 provides an easy way to calculate the diagonal matrix D that minimizes the dominant eigenvalue of $D+G$ for a diagonally-symmetrizable and non-negative G. We can first calculate an optimum $\bar{D}$ without the individual constraints on D_i, i.e., if every $\bar{D}_i$ satisfies its constraint, we have found an optimum. Otherwise, if $\bar{D}_i$ that violate their constraints are either all larger than L

or all less than 0, the actual optimal D_i for positions where constraints are violated are equal to boundary values. This allows us to quickly locate the D_i's at the boundary rather than to try all combinations of K at boundary to find an optimal solution. In fact, at most n cases (rather than 3^n) need to be considered. If $\bar{D}$ has entries less than 0 and greater than L at the same time, we must fall back on Theorem 2.4, i.e. search through the possible combinations of D_i at the boundaries.

Interpretations of the Optimal Design $D^* + G$

In this section, we have discussed designing a matrix D to minimize the dominant eigenvalue of $D + G$ subject to constraints. This design specifically allows us to allocate limited repair resources to a contact network, so as to best fight against the spread of a virus. From this viewpoint, it is instructive to study the structure of an optimizing $D = D^*$, and hence the structure of $D^* + G$.

The structure of an optimizing D is highly dependent on the structure of the matrix G, which describes the connection topology of the network. The theorems give us the insight that, for a symmetric G, the matrix D should be chosen to best equalize the row sums of $D + G$, within the permitted constraints. In terms of resource allocation, this means that placing the most resource (whether sensing capabilities, vaccinations, quarantine, or other resources)at the nodes that have strong connections best prevents virus spread. This makes sense since these nodes have the strongest potential to spread the virus throughout the network if they are infected, and similarly to heal the network when they are healthy. Eliminating viruses at these nodes as soon as possible can quickly quench the spread. In case the individual constraints prevent placing enough resource at a node, nearby nodes are provided with extra resources to prevent spread. It is interesting to note that our optimal control strategy in general does not shatter the contact network into multiple components (which has been shown in [75] to be infeasible because a very large number of vertices, even of high degrees, must be removed to break the network), but instead sufficiently impedes the spread of the virus (in a probabilistic sense) to reduce the basic reproduction ratio below 1 with limited resource. An interesting future direction may be to extend our design to the case where resources are allocated to many nodes (individuals) at once, so as to reflect the concept of placing epidemic sensing capacities at locations visited by many individuals as proposed in [75].

It is worth noting that this design is suitable for repair resource allocation before the break-out of a virus or real-time during a virus. In other words, this design is robust to the initial location of the virus. This is useful even when real-time allocation of resources after the start of an epidemic is not possible, or when it is hard to locate and respond to infected nodes network-wide in an epidemic. When the initially affected nodes are known, the design can be improved further using this additional information. We leave this improvement to later work.

We recall that our analysis of the contact-network model is based on a linear approximation, and so it is worthwhile to briefly consider the dynamics of the original nonlinear model upon application of our design strategy. A (continuous-time) Markov-process formulation of the nonlinear dynamic model (see e.g. [76]) makes clear that, in fact, epidemics always eventually die out in models such as ours, and so the time until die-off is of particular interest. Depending on the dominant eigenvalue of the next-generation operator, the die-off may either be rapid (e.g. logarithmic in the size of the network), or the virus may be essentially persistent (requiring a die-off time that is exponential in network size) [76]. Let us now consider applying our repair-resource design to networks of increasing size. Assuming that the network topology is symmetric, and that enough repair resources are provided to ensure that eigenvalues of the next-generation operator are uniformly within a bounded set strictly inside the unit circle, we can replicate the majorization-based argument of [76] to prove that the die-off time is logarithmic in network size.

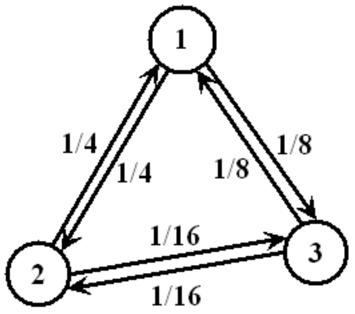


Fig. 2.1: Illustration of a three-node network topology.

Example We illustrate calculation of $D = D^*$ to minimize the dominant eigenvalue of $D + G$ for the nonnegative and symmetric topology matrix $G = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{4} & 0 & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{16} & 0 \end{bmatrix}$, subject to the constraints $D_i \in [0, 1]$ and $\sum_i D_i \geq \Gamma$ (see Fig. 2.1). We consider the following two cases.

1) $\Gamma = 1.5$. In this case, we have ${D_1}^* = 0.4167$, ${D_2}^* = 0.4792$, ${D_3}^* = 0.6042$, and $\lambda^* = 0.7917$ and $v_{max} = [1\ 1\ 1]^T$.

2) $\Gamma = 2.9$. In this case, we find that ${D_1}^* = 0.928$, ${D_2}^* = 0.972$, ${D_3}^* = 1$, which yields $\lambda^* = 1.2661$, and $v_{max} = [1\ 1\ 0.7047]^T$. We notice that the sum of each row of $D^* + G$ is not identical in this case, and that D_1 is increased by a larger fraction than D_2 to make up for the resource deficiency at node 3.

A brief discussion of the procedure for finding the optimal in the example is worthwhile. When $\Gamma = 1.5$, the D_i obtained following step 2) in Theorem 2.6 are feasible. Hence this single step finds an optimum D. When $\Gamma = 2.9$, D_3 obtained following step 2) is larger than 1, while D_1 and D_2 are smaller than 1. This satisfies the condition of step 3). Hence by fixing D_3 at 1, and following the calculation in Theorem 2.4, we find an optimum D^*.

2.2.3 Designing $\lambda_{max}(KG)$ and $\lambda_{max}(D+KG)$

In this section, we address the other two design problems needed for virus-spreading control, design of diagonal K to minimize the dominant eigenvalue of KG and $D+KG$ (D diagonal) respectively.

The results are similar to those for the $D+G$ case, so the proofs are omitted. Here, we denote an optimum **gain** K as K^*, the optimum dominant eigenvalue of KG ($D+KG$) as λ^*_{max}, and the corresponding left and right eigenvector of KG ($D+KG$) as w^*_{max} and v^*_{max}. Because the results for KG and $D+KG$ problems are so similar, we present them together. We begin with a general structural necessary condition on an optimum:

Theorem 2.7. *Consider the matrix KG (or $D+KG$), where K and D are diagonal and G is a $n \times n$ matrix. Consider a matrix $K = K^*$ that minimizes the dominant eigenvalue of KG (or $D+KG$) subject to the constraints 1) $\sum K_i \geq \Gamma$ and 2) $K_i \in [0, L]$, assuming the dominant eigenvalue of KG is real and non-repeated. The optimizing K^* and the corresponding left and right eigenvectors v^*_{max} and w^*_{max} satisfy one of the following conditions:*

1) $\sum K^*_i = \Gamma$. In this case, for each i we either have $0 < K^*_i < L$ and $\lambda^*_{max} {K^*_i}^{-1} w^*_{max_i} v^*_{max_i} = 1$, or we have $K^*_i = L$ or $K^*_i = 0$.

2) $\sum K^*_i > \Gamma$. In this case, for each i we either have $0 < K^*_i < L$ and $w^*_{max_i} v^*_{max_i} = 0$, or we have $K^*_i = L$ or $K^*_i = 0$.

An optimum K can be found through a search algorithm, as stated in Theorems 2.8-2.11. Theorems 2.8 and 2.9 are for the KG case, and Theorems 2.10 and 2.11 are for the $D + KG$ case.

Theorem 2.8. *Consider a topology matrix G that is non-negative, irreducible, and diagonally symmetrizable. The $K = K^*$ that minimizes the dominant eigenvalue of KG subject to the constraints 1) $\sum K_i \geq \Gamma$ and 2) $K_i \in [0, L]$ can be found using the following algorithm:*

1) Find diagonal Q such that $Q^{-1}GQ$ is symmetric. We denote $Q^{-1}GQ$ as $\hat{G}$.

2) Set some K_i to 0 and some K_i to L, remove the rows and columns of $\hat{G}$ corresponding to the K_i chosen as 0, and rearrange the remaining rows and columns of $K\hat{G}$ in such a way that those K_i fixed at L are at the lower right corner. The resulting matrix can then be written as $\begin{bmatrix} K_A & 0 \\ 0 & LI \end{bmatrix} \begin{bmatrix} \hat{G}_{11} & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22} \end{bmatrix}$.

3) The eigenvalue λ satisfies $\lambda \mathbf{1}^T (\hat{G}_{11} - \hat{G}_{12}(L\hat{G}_{22} - \lambda I)^{-1} L\hat{G}_{21})^{-1}\mathbf{1} = \Gamma - trace(LI)$, and can be found through a simple numerical procedure. K_A can be found as $K_A = diagonalize(\lambda \mathbf{1}^T (\hat{G}_{11} - \hat{G}_{12}(L\hat{G}_{22} - \lambda I)^{-1} L\hat{G}_{21})^{-1})$. Check whether K_i is feasible.

4) Repeat with different K_i set to their boundary values until a global minimum λ is found.

Theorem 2.9. *Consider a topology matrix G that is non-negative, irreducible, and diagonally symmetrizable. The $K = K^*$ that minimizes the dominant eigenvalue of $D + KG$ subject to the constraints 1) $\sum K_i \geq \Gamma$ and 2) $K_i \in [0, L]$ can be found using the following algorithm:*

1) Find diagonal Q such that $Q^{-1}GQ$ is symmetric. We denote $Q^{-1}GQ$ as $\hat{G}$.

2) Set some K_i to 0 and some K_i to L, and rearrange the rows and columns of $D+K\hat{G}$ in such a way that those K_i fixed at 0 or L are at the lower right corner. The resulting matrix can then be written as $\begin{bmatrix} D_A & 0 \\ 0 & D_B \end{bmatrix} + \begin{bmatrix} K_A & 0 \\ 0 & K_B \end{bmatrix} \begin{bmatrix} \hat{G}_{11} & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22} \end{bmatrix}$.

3) The eigenvalue λ satisfies $(\lambda\mathbf{1}^T - D_A\mathbf{1}^T)(\hat{G}_{11} - \hat{G}_{12}(D_B + K_B\hat{G}_{22} - \lambda I)^{-1}K_B\hat{G}_{21})^{-1}\mathbf{1} = \Gamma - trace(K_B I)$, and can be found through a simple numerical procedure. K_A can be found as $K_A = diagonalize(\lambda\mathbf{1}^T - D_A\mathbf{1}^T)(\hat{G}_{11} - \hat{G}_{12}(D_B + K_B\hat{G}_{22} - \lambda I)^{-1}K_B\hat{G}_{21})^{-1})$. Check whether K_i is feasible.

4) Repeat with different K_i set to their boundary values until a global minimum λ is found.

When G is positive definite in addition to diagonally symmetrizable, we can show that $\lambda_{max}(KG)$ is a convex function. In this case, the search algorithms given in Theorems 2.8 and 2.9 can be simplified: we can check whether a solution is a local optimum and stop or continue the search accordingly, since a local optimum is guaranteed to be a global optimum. The simplified algorithms are given below. Theorem 2.10 is for the KG case, and Theorem 2.11 is for the $D + KG$ case.

Theorem 2.10. *Consider a topology matrix G that is non-negative, irreducible, positive definite, and diagonally symmetrizable. The $K = K^*$ that minimizes the dominant eigenvalue of KG subject to the constraints 1) $\sum K_i \geq \Gamma$ and 2) $K_i \in [0, L]$ can be found using the following algorithm:*

1) Find diagonal Q such that $Q^{-1}GQ$ is symmetric. We denote $Q^{-1}GQ$ as $\hat{G}$.

2) Find $K = diagonalize(\hat{G}^{-1}\mathbf{1}\Gamma/(\mathbf{1}^T\hat{G}^{-1}\mathbf{1}))$. If $0 \leq K_i \leq L$, this solution is optimal. Otherwise, go to step 3.

3) *Set some K_i to 0 and some K_i to L, remove the rows and columns of G corresponding to the K_i chosen as 0, and rearrange the remaining rows and columns of KG in such a way that those K_i fixed at L are at the lower right corner. The resulting matrix can then be written as* $\begin{bmatrix} K_A & 0 \\ 0 & LI \end{bmatrix} \begin{bmatrix} \hat{G}_{11} & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22} \end{bmatrix}$.

4) *The eigenvalue λ satisfies $\lambda \mathbf{1}^T(\hat{G}_{11} - \hat{G}_{12}(L\hat{G}_{22} - \lambda I)^{-1} L\hat{G}_{21})^{-1}\mathbf{1} = \Gamma - trace(LI)$, and can be found through a simple numerical procedure.*

5) *K_A can be found as $K_A = diagonalize(\lambda \mathbf{1}^T(\hat{G}_{11} - \hat{G}_{12}(L\hat{G}_{22} - \lambda I)^{-1} L\hat{G}_{21})^{-1})$. If $0 \leq K_i \leq L$, and the left eigenvectors associated with the dominant eigenvalue of KG has the pattern that the entries corresponding to $K_i = L$ are less then those corresponding to $0 < K_i < L$, and the latter are less than those corresponding to $K_i = 0$, this solution is optimal. Otherwise, go back to step 3.*

Theorem 2.11. *Consider a topology matrix G that is non-negative, irreducible, positive definite, and diagonally symmetrizable. The $K = K^*$ that minimizes the dominant eigenvalue of $D + KG$ subject to the constraints 1) $\sum K_i \geq \Gamma$ and 2) $K_i \in [0, L]$ can be found using the following algorithm:*

1) *Find diagonal Q such that $Q^{-1}GQ$ is symmetric. We denote $Q^{-1}GQ$ as $\hat{G}$.*

2) *Find λ that satisfies $\hat{G}^{-1}(\lambda \mathbf{1}^T - \mathbf{1}^T D)\mathbf{1} = \Gamma$ through a simple iteration. And then find K from $K = diagonalize(\hat{G}^{-1}(\lambda \mathbf{1}^T - \mathbf{1}^T D))$. If $0 \leq K_i \leq L$, this solution is optimal. Otherwise, go to step 3.*

3) *Set some K_i to 0 and some K_i to L, and rearrange the rows and columns of $D + KG$ in such a way that those K_i fixed at 0 or L are at the lower right corner. The resulting matrix can*

then be written as $\begin{bmatrix} D_A & 0 \\ 0 & D_B \end{bmatrix} + \begin{bmatrix} K_A & 0 \\ 0 & K_B \end{bmatrix} \begin{bmatrix} \hat{G}_{11} & \hat{G}_{12} \\ \hat{G}_{21} & \hat{G}_{22} \end{bmatrix}$.

4) The eigenvalue λ *satisfies* $(\lambda \mathbf{1}^T - D_A \mathbf{1}^T)(\hat{G}_{11} - \hat{G}_{12}(D_B + K_B\hat{G}_{22} - \lambda I)^{-1} K_B \hat{G}_{21})^{-1}\mathbf{1} = \Gamma - trace(K_B)$*, and can be found through a simple numerical procedure.*

5) K_A *can be found as* $K_A = diagonalize((\lambda \mathbf{1}^T - D_A \mathbf{1}^T)(\hat{G}_{11} - \hat{G}_{12}(D_B + K_B\hat{G}_{22} - \lambda I)^{-1} K_B \hat{G}_{21})^{-1})$. *If* $0 \leq K_i \leq L$*, and the left eigenvectors associated with the dominant eigenvalue of* $D + KG$ *has the pattern that the entries corresponding to* $K_i = L$ *are less then those corresponding to* $0 < K_i < L$*, and the latter are less than those corresponding to* $K_i = 0$*, this solution is optimal. Otherwise, go back to Step 3.*

2.2.4 Computation and Implementation Issues

Let us make a couple of notes on our methods. First, the reader may wonder why we have developed algorithms that are specialized to the stated decentralized design task rather than using standard optimization algorithms, so we will briefly comment on the computational and conceptual advantages provided by this approach. Second, we will briefly discuss the robustness of the design strategy.

Advantages of our Optimization Approach

Computationally, the design (optimization) problem studied here is hard, because it is a *decentralized* design/control problem with a cost based on an associated dynamics (e.g., a settling-rate cost). Decentralized problems of this sort are known in general to be NP-hard, see e.g. [77]. The particular decentralized problem of designing a diagonal matrix (say K) so as to place the eigenvalues of KG or $K+G$ in desirable locations subject to constraints falls within this class of challenging problems: in fact, there is no known general algorithm for deciding whether there is K that places the eigenvalues in the unit circle (or in the closed left half plane), let alone doing so with minimal resource [29, 78, 79]. Essentially, the difficulty in the problem stems from the fact that the cost depends in a complicated (and implicit) way on the design variables, and yet there are not sufficient degrees of freedom to construct the dynamics (modes) at will. Because of these limitations, standard optimization packages cannot be used to find the optimal design. Specialized semi-definite programming and in turn *linear matrix inequality* (LMI) techniques can be applied for some classes of symmetric topology matrices G (see e.g. [80, 81]), but these techniques are not applicable for the broad class of positive matrices considered here, nor can they be specialized to guarantee solution using only a small number of iterations as we have done for the symmetrizable case. More specifically, we have studied the KG and $D+G$ design problems for networks of up to 1000 regions (individuals), and find that typically at most 3 iterations of our algorithm are needed to exactly find the optimal. Also, we note that some clever manipulation is needed to pose the design problem as an SDP problem even in the symmetric case, so that implementation of the algorithm through this approach is also complicated. The essential difficulties in decentralized design/control problems have led to a burgeoning interest in recent years on design for modern networks [25, 26, 29] using graph-theoretic and matrix-theoretic methods; broadly, our techniques are aligned with this methodology and permit low-computation design for relevant network topologies.

We also contend that the analytical design method presented here provides a conceptual advantage over numerical techniques. In fact, the presented algorithms admit a simple conceptual interpretation, for the symmetrizable case: resources should be provided to regions (individuals) to equalize the impact of the infectives from each region to the other ones. In the case where the resource limits do not permit such allocation to a particular region (individual), neighboring regions must be provided with additional resources to compensate for the resource deficiency (and, equivalently, if a region has more resource allocation than needed due to the lower resource bound, resources in neighboring regions can be curtailed. We believe that this simple conceptualization of the optimal solution is valuable, especially, for achieving robustness: even when the model parameters and/or control models are inaccurate, one can allocate resources with the aim of equalizing spread impact and expect to obtain a good if not optimal solution.

Robustness of the Design

Second, an important practical concern in applying these algorithms to the epidemic-control problems introduced in Section 2.1.1 is whether or not the parameters in the models can be approximated with sufficient fidelity to achieve a high-performance design. In discussing these concerns, we first note that the parametric data needed for the optimizations are only those needed for analyzing the basic reproduction ratios of the two studied models (the multi-group model and the contact model). We thus refer the reader to existing work [24, 45] for discussions on how these parameters can be identified. Fundamentally, we notice that the parameters in both models are concerned with either with levels of interaction or with the infectiousness of the virus, and so counting of flows/interactions as well as infection relative to total interaction can be used to find the parameters. Also, the interaction parameters in particular can also be identified from historical epidemics in the network of interest. However, in that we are proposing control actions, and recognizing that the model parameters may indeed be uncertain (especially those that may need to be obtained in real time), we recognize that the robustness of the developed algorithm requires further study. Our preliminary efforts in this direction are very promising: for the Hong Kong SARS virus example, we have found that the optimal heterogeneous optimal design far outperforms a homogeneous design even when the parameters are uncertain by as much as 40% (see Section 2.3 for details). This high tolerance of the design to parameter errors is not surprising since the optimum is deeply tied to the network structure, and more specifically to the total impact of a region/node. These structural features are well-characterized even when individual parameters are uncertain. We note that this conceptual justification of robustness is yet one more advantage of addressing the decentralized design problem explicitly rather than trying to apply standard optimization tools.

2.3 Control of the Hong Kong SARS Epidemics

Recall that the chapter [24] developed a spatial model for the spread of SARS in Hong Kong's 18 districts (See Fig. 2.2), and proposed a homogeneous control for reducing the basic reproduction ratio R_0 to 1. In that model, the nominal corrective factor $\bar{f}_{ji}$ takes three different values contingent on the structural relationship of i and j: 1) $i = j$, 2) District i and j are contiguous, and 3) District i and j are not contiguous.

Here, we find a optimal heterogeneous control that uses the same total resource amount as the controller in [24]. This controller reduces the basic reproduction ratio to 0.64. Thus, we see that an epidemic can be stopped more quickly with the same control resources, by allocating more resources to some districts than others. Equivalently, it is easily shown that $R_0 = 1$ can be achieved even when the total control resource is reduced to 79% of the one with equal allocation (See Table 2.2).

This intelligent allocation takes advantage of the spatial structure of the population, by placing more control resources in the districts that are important for the spread of an epidemic (see Fig. 2.2).

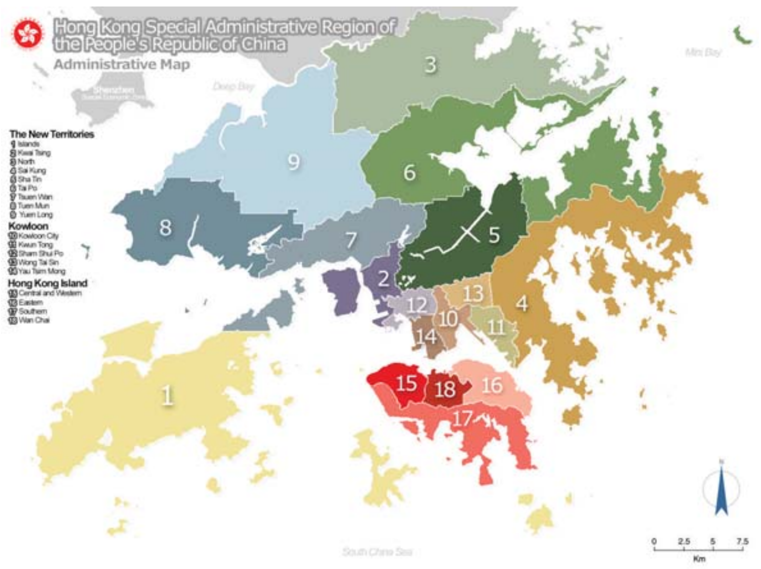


Fig. 2.2: The Map of Hong Kong's 18 districts (obtained from *http://en.wikipedia.org/wiki/Hong_Kong#Administrative_divisions*). We use the model parameters in [24]: $\bar{f}_{ii} = 1$, $\bar{f}_{ji} = 0.57$ when District i, j are neighbors, $\bar{f}_{ji} = 0.02$ when i, j are not adjacent, $\bar{\beta} = 0.062$, and $\bar{T} = 10.6$.

In this way, the limited control resources are best able to reduce the rate at which the epidemic diminishes. Such a control would reduce the impact on people's daily lives in some districts (which have less control resources allocated) and overall, while still stopping the virus spread quickly.

For illustration, we also consider how the heterogeneous resource allocation changes when the transmission coefficient within a district is increased (compared to the mixing rate between districts). As expected, increasing the local mixing rate makes use of spatial information less important, and also makes the allocation more homogeneous (See Table 2.2).

In all of these experiments, we see that the most resources are placed in Districts 5 and 7, and the smallest resources are placed in District 1. This is expected since Districts 5 and 7 are the ones with pivotal locations (e.g. with many neighbors) and hence the successful control of these districts is important in the control of disease spread. In contrast, District 1 is almost isolated and hence has the least contribution to the spread of diseases in Hong Kong.

We have also pursued a preliminary robustness analysis for our design. In particular, we have analyzed the performance of our design in the case where the actual parameters are different from the ones used for the design. For illustration, let us consider one of the designs shown in Table 2.2, say the one corresponding to $\bar{f}_{ii} = 3$. In this case, we see that a heterogeneous design reduces R_0 to 0.87 if the total amount of resource is that for which a homogeneous design reduces R_0 to 1. Now say say that each interaction parameter $\bar{f}_{ij}$ is in fact in error by at most 40%, or more specifically that the parameter is in fact $\alpha_{ij}\bar{f}_{ij}$, where α_{ij} is uniformly distributed between 0.6 and 1.4. Over 100 trials, our design achieves average $R_0 = 0.91$, as compared to average $R_0 = 0.88$ if the design were re-optimized for the new parameters. We thus see that the design still significantly outperforms a homogeneous design with 40% error in the parameters.

Tab. 2.2: Resource Allocation Needed to Reduce R_0 to 1.

	$\bar{f}_{ii} = 1$	$\bar{f}_{ii} = 1.5$	$\bar{f}_{ii} = 3$
District 1	0.9096	0.6222	0.3031
District 2	0.9096	0.5368	0.2525
District 3	0.5816	0.4338	0.2546
District 4	**0**	**0**	0.1919
District 5	**0**	**0.0160**	**0.0915**
District 6	0.5816	0.4242	0.1705
District 7	**0**	**0**	**0.0834**
District 8	0.9096	0.5926	0.2708
District 9	**0**	**0.0789**	0.1734
District 10	**0**	**0**	0.1419
District 11	0.5816	0.4553	0.2194
District 12	**0**	0.2118	0.1798
District 13	0.5816	0.4457	0.1985
District 14	0.9096	0.5428	0.2587
District 15	0.9096	0.4784	0.2475
District 16	0.9096	0.4784	0.2475
District 17	**0**	0.1919	0.1908
District 18	**0**	0.1919	0.1908
$\sum K_i$ using heterogeneous control	7.7842	5.7007	3.6667
$\sum K_i$ using homogeneous control	5.0138	4.4009	3.22

The K_i for each district are shown in the table. Notice that smaller K_i corresponds to more resources. $\sum K_i$ equals the subtraction of total utilized resources from N (the number of districts, e.g., 18 in this case). Note that the reduction in resource used for $\bar{f}_{ii} = 1$ is to $\frac{18-7.7842}{18-5.0138} \times 100\% = 79\%$ of the homogeneous allocation.

3. A SCALABLE METHODOLOGY FOR EVALUATING AND DESIGNING COORDINATED AIR TRAFFIC FLOW MANAGEMENT STRATEGIES UNDER UNCERTAINTY

As congestion in the United States National Airspace System (NAS) increases, coordination of en route and terminal-area traffic flow management procedures is becoming increasingly necessary, in order to prevent controller workload excesses without imposing excessive delay on aircraft. Here, we address the coordination of flow management procedures in the presence of realistic uncertainties, by developing a family of abstractions for implementable flow restrictions (e.g., miles-in-trail restrictions, ground delay programs, or slot-based policies). Using these abstractions, we are able to evaluate the impact of multiple restrictions on generic (uncertain) traffic flows, and hence to design practical flow management strategies. We use the developed methodology to address several common design problems, including design of multiple restrictions along a single major traffic stream and design of multiple flows entering a congested terminal area or Sector. For instance, we find that multiple restrictions along a stream can be used to *split* the backlog resulting from a single restriction, and use this observation to develop low-congestion designs. We conclude the discussion by posing a tractable NAS-wide flow management problem, using a simple algebraic model for a restriction.

3.1 Introduction

Traffic flow management (TFM) procedures have been used successfully for many years in the United States National Airspace System (NAS), to meet the capacity constraints resulting from arrival-rate and controller-workload requirements in the face of uncertainties including adverse weather, unexpected airborne delays, and departure delays [82–85]. Operationally, TFM is implemented using procedures such as Air Traffic Control (ATC) preferred routes, time-based metering, miles-in-trail (MIT), minutes-in-trail (MINIT), ground stops (GS), and ground delay programs (GDPs) (see [86–88] for details of the TFM mechanisms). In recent years, there has been a growing need to coordinate these TFM actions over multiple Centers or even NAS-wide: for example, congestion at Philadephia (PHL) airport needs to be alleviated by redistributing the delays and workloads further upstream [89]. Effectively bringing multiple Centers into collaboration for flow management is complicated because management actions cause complex correlations among aircraft in *uncertain* flows, both upstream and downstream. This need for coordination motivates study of TFM from a network point of view, which could provide a better understanding of global TFM performance, and hence permit better management of aircraft traffic flows using existing and new mechanisms. With this motivation, we develop a methodology for analyzing the performance of networked TFM strategies operating *in the face of realistic uncertainties*, and in turn design high-performance TFM strategies that can be implemented using existing mechanisms and/or new low workload procedures such as the slot-based approach [90].

A formulation of the multi-Center TFM problem was given in [92], where the challenges associated with implementing time-based metering of traffic to Philadelphia airport were illustrated. This work was followed by the development of frameworks for coordination, including a distributed scheduling architecture [91] and Multi-Center Traffic Management Advisor (McTMA)-based architecture [89]. Other work related to multi-Center TFM focuses on developing abstract deterministic models for traffic flow, and in turn pursuing TFM design by solving optimal scheduling problems in the context of the models [93, 94, 96, 97]. Our work here builds on both the implementation-focused studies [89, 91, 92] and the analytical approaches [93, 94, 96, 97], in that we use models that capture practical considerations of implementation/robustness, but in a way that permits analysis and design. Our studies build on network modeling efforts for air traffic flow [82, 100] as well as queueing models for flow restrictions (e.g., [86]).

Our study of multi-Center TFM recognizes the critical role played by uncertainties and implementation concerns, so it is worthwhile for us to briefly review the literature on these aspects. The impact of uncertainties on air traffic has long been recognized [82–85], and traffic control actions such as conflict avoidance maneuvers have been designed to be robust to uncertainties (e.g., [98]). From the perspective of flow management rather than detailed control, recent works in network modeling have implicitly or explicitly modeled uncertainty (e.g., [82, 86, 100]). The intrinsic complexities associated with implementing flow management have been widely recognized, and in fact a metric for the performance of implemented flow management strategies has been developed in [99]. This performance analysis of flow management strategies also recognizes the important role played by weather uncertainties, and shows how the impact of weather can be separated from the intrinsic performance of the method.

Here, we make two advances in network TFM: 1) we capture the specifics of existing TFM restrictions within the network models, and 2) we study the impact of coordinated TFM strategies on *generic* or typical uncertain flows, see [86] for such analysis for a single restriction. These efforts provide the benefit of *evaluation* of managed air traffic flows in advance since uncertainties in flows are considered, and lead to designs that are practical for implementation in the existing operational framework. The reader will note that we emphasize on the modeling and design of restrictions on aircraft *flows* since many currently-used restrictions (e.g., MIT, MINIT, some GDPs) have impact on aircraft flows rather than requiring scheduling of individual aircraft. Our goal is to model such restrictions in a scalable way, i.e. such that NAS-wide evaluation and design is permitted. We note that our focus on flow management under uncertainty is closely aligned with the perspective given in the chapters [101, 102], which study optimization of coordinated re-planning and testing of the optimal re-planning strategies in an uncertain simulation environment. Our approach builds on theirs in that we design management strategies that are robust to uncertainty rather than designing based on a forecast. Also, we obtain simple structural insights into good strategies, albeit at the cost of using much more abstracted models for aircraft flows.

To facilitate the NAS-wide design task described above, we need to pursue two abstraction tasks: one is to represent the traffic flow between regions (network flow/routing/splitting model) under uncertainty; and the other is to model restrictions in a way that would allow design on a network level. The key idea here is that we view this as a tractable network controller design problem with stochastic air traffic flows as inputs. Each traffic flow can be modeled as a stochastic process (e.g., Poisson Process), as justified in [82, 86]. The restriction placement can be viewed as a controller design, in the sense that the downstream flow is managed by the controller based on the rate of flow in the upstream. In our previous work [86], we understand the essential impact of a single restriction on traffic flow: the restriction smooths out the downstream flows (reduces their variance), at the cost of upstream backlog and delay. Unfortunately, as we note in [86], the single-restriction analysis does not easily scale to multi-restriction and hence multi-Center designs. The complexity of the relationship between upstream and downstream flow under restriction drives us to seek abstract restriction models that can capture the intrinsic relationship between the flows, while significantly reducing the complexity of the analysis, and so permitting multi-Center TFM design. In particular, we suggest a saturation model for restriction, as well as linear dynamic and algebraic approximations. As we shall show, majorization and linear system analysis are the tools that allow performance evaluation for the new models.

The remainder of the chapter is organized as follows: in Section 3.2, we introduce our framework for studying flow management restriction design and overview the family of abstractions for restrictions. In Section 3.3, we describe the saturation model, and show that it can be used to analyze flow statistics in several topologies. In Section 3.4, we describe the linear abstraction and use it for design of multiple restrictions. In Section 3.5, we give some preliminary discussions on using a simple algebraic model for network-wide flow control design.

3.2 A Family of Abstractions For Flow Management: Overview

Let us introduce a framework for studying coordination of flow management restrictions and overview our abstractions for individual restritions. Each abstraction captures the essential effects of restrictions on general aircraft flows, but at different levels of detail/tractability. Together, they permit evaluation and design of coordinated TFM.

Broadly, traffic in the NAS can be viewed as flows of aircraft (network flows) entering and leaving *regions* (Sectors/Centers/airports). Traffic flow management is implemented by placing restrictions on boundaries of the regions to avoid congestion in downstream regions, while not causing unsatisfactory delay. We are interested in evaluating the effects of practical restrictions on typical flows, which we expect to vary day-to-day due to unpredictable events such as weather and take-off time uncertainties.

We first consider the flow into each restriction. We model this arrival of aircraft (or **entering**

flow) in a quite-general way, as a stochastic arrival process. We denote the the arrival rate—the mean number of aircraft arriving at the boundary per unit time—as λ [*]. When a flow is an amalgam from several routes or represents departures from a busy airport, a Poisson process model is appropriate [82, 86]. We study both the general and Poisson models here.

Let us now consider the dynamics of the restriction (control) at each boundary. To ease understanding, we imagine a virtual buffer at each boundary, as shown in Figure 3.1. Every approaching aircraft automatically enters the buffer (at time t, the number of aircraft in buffer i, or the **buffer count**, is denoted by $b_i(t)$). However, only a portion of the aircraft in the buffer are allowed to enter the downstream region. We denote the stochastic process describing the flow of aircraft into the downstream region by $e_i(t)$, and call it the **crossing flow**. The number of aircraft being delayed by the restriction at time t (**backlog**) is denoted by $B_i(t)$. Note that $B_i(t)$ is different from $b_i(t)$ in that $B_i(t)$ does not count the aircraft that enter the buffer and then leave smoothly without being held by the restriction. The relationship of a boundary's crossing flow $e_i(t)$ with respect to its buffer count $b_i(t)$ is what we aim to design. Specifically, we develop several abstractions for implementable restrictions (e.g., MINIT, MIT, and slot-based restrictions) within this framework with the aim of achieving designs that can be put in place easily.

In some analyses, we explicitly model counts in downstream regions. We assume each aircraft in a stream takes a constant time T to cross a downstream region with a capacity of C aircraft. This assumption is often reasonable for en route airspace, where the aircraft follow a route through a region at an approximately constant speed. The number of aircraft in a downstream region (or

[*] As noted in [86], the arrival rate at boundaries are in fact time-varying. Here, we are concerned with flows during the busy part of the day or during particular common inclement-weather or other special scenarios (when the restrictions are needed), so we use a time-invariant model for simplicity. Many of the results generalize naturally to the time-varying case.

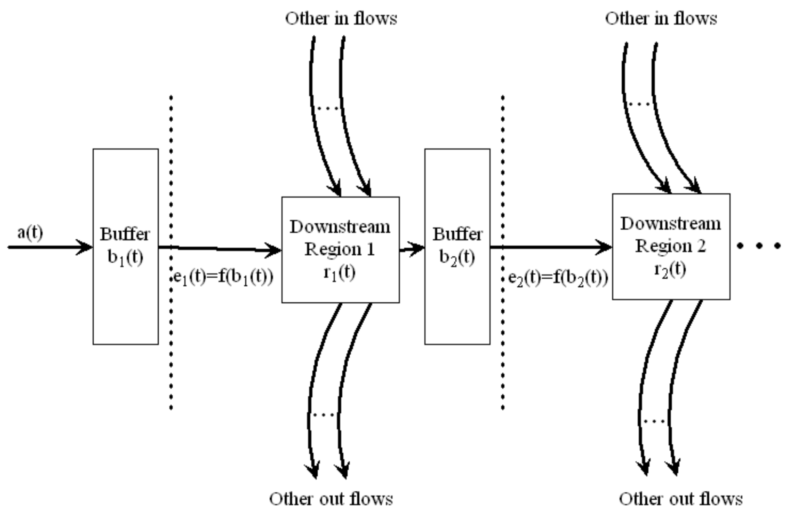


Fig. 3.1: Boundary restriction framework.

region count) is denoted as $r_i(t)$.

At a network level, we view streams of aircraft counting from various boundary restrictions as merging and splitting within the regions. While restriction design for various network topologies are of interest, a couple simple topologies are especially common, and so we often focus on these topologies or study them in examples. One common phenomenon is the existence of a major stream passing through a sequence of restrictions (for instance, traffic from the West Coast to Northeast airports) together with some minor flows entering and leaving the major flow. We thus often consider a stream of aircraft passing through a series of boundary restrictions, and aim to set these boundary restrictions for suitable management. A second common topology is one with a single capacity-constrained region with multiple flows that can be restricted before entering the region. Such a network model is appropriate, for instance, in developing restrictions around a crowded terminal area or Sector.

Our primary goal is to develop a series of abstract restriction models that capture the key attributes of en route and terminal area restrictions, while permitting analysis of networked (coordinated) restrictions to varying degrees. The models that we have studied, in order of increasing abstraction (and also greater tractability), are:

1) Detailed queueing models for en route and terminal area restrictions (see [86]);

2) A discrete-time saturation model;

3) A dynamic stochastic linear model;

4) An algebraic linear model, for design in arbitrary networks.

We shall develop the three more abstract models in detail in this chapter. We kindly refer the reader to our previous work [86] for details on the detailed queueing models, however let us give a brief overview of these queueing models to motivate the need for further abstraction. Queueing and queueing network models enjoy very wide application, including in representing aspects of both road and air traffic control [86,103–106]. Of particular interest, in [86] we modeled an en route flow restriction as a single-server queue with deterministic service times ($M/D/1$), and characterized the tradeoff between downstream smoothing and upstream backlog/delay using the classical analysis of $M/D/1$ queues. However, the queueing model is not amenable to a complete analysis of the downstream traffic, nor easily scalable to a network setting.

3.3 Saturation Restriction Model

In order to better understand a restriction's impact on flows and achieve design of coordinated restrictions, we develop a discrete-time abstraction of the queueing model that we call the **saturation restriction model**. This discrete time abstraction allows the analysis of downstream variability for Poisson inflows, and in turn permits qualitative and quantitative study of multi-Center restriction placement.

3.3.1 Description of the Saturation Restriction

The saturation restriction works as follows: during each time step of length ΔT, it allows a maximum of N_c aircraft to pass the boundary, while the remaining aircraft remain in the buffer. Formally, let us denote the number of aircraft arriving at a boundary between times $k\Delta T$ and $(k+1)\Delta T$ as $x[k]$, the number of aircraft allowed to pass the boundary during this time as $e[k]$, the number of aircraft in the buffer at time $k\Delta T$ as $b[k]$, and the backlog at time $k\Delta T$ as $B[k]$. These variables evolve as follows:

$$\begin{aligned} e[k] &= \begin{cases} b[k-1], & (b[k-1] \leq N_c) \\ N_c, & (b[k-1] \geq N_c) \end{cases} \qquad (3.1) \\ b[k] &= b[k-1] + x[k] - e[k] \\ B[k] &= b[k-1] - e[k] \end{aligned}$$

The parameter N_c of the saturation model can be matched with the specifics of actual TFM restrictions. In particular, many restrictions enforce that aircraft are separated by τ minutes, and hence that $N_c = \frac{\Delta T}{\tau}$ aircraft are allowed to pass the boundary in ΔT minutes. For instance, using this reasoning, N_c can be chosen as $\frac{\Delta T}{q}$ to represent q-MINIT restrictions and similarly a q-MIT restriction can be matched with $N_c = \frac{\Delta T v}{q}$ assuming that each aircraft travels at velocity v. In designing restrictions, it is worth noticing that N_c needs to be larger than the average aircraft flow rate $\lambda\Delta T$; otherwise, there would be an unlimitedly growing aircraft count in the buffer. We expect the setting of N_c will affect the flow, e.g., downstream volume, upstream backlog and delay. The mapping from a TFM restriction to N_c provides an approach to evaluate the restriction's impact on flow, and conversely a designed N_c can be matched with a restriction duration.

We also note that the saturation model captures the slot-based management strategy proposed in [90]. In particular, a restriction where aircraft are placed in slots that are q minutes apart can be represented using a saturation model with $\Delta T = q$ and $N_c = 1$. This representation captures the behavior of the slot model, with only the slight error that the initial delay of a fraction of a slot when an aircraft first approaches a boundary is not captured. Our approach thus permits evaluation of the slot-based strategy for stochastic entering flows.

The saturation model is a (tractable) discrete-time counterpart of the various detailed queueing models, in that it also restricts aircraft at busy times and permits them to go through at times with a small arriving flow. However, a couple limitations of this approximation of queueing models are worth noting: 1) the saturation model is best suited for queueing dynamics that exhibit a thresholding behavior, and 2) the model does not capture the queueing model's dynamics accurately over very short time intervals, i.e. intervals of smaller order than the times between aircraft.

3.3.2 Model Evaluation for Poisson Input Flow Statistics

For the saturation model, the statistics of upstream and downstream flows can be explicitly computed when the approaching flow is a Poisson process. In this case, the numbers $x[k]$ that enters the buffer at different time steps are independent because of the Poisson process' independent increment property [107]. Specifically, we recall that when the input process is a Poisson Process of rate λ, the probability that $x[k] = c$ aircraft arrive at time step k is as follows:

$$P_\lambda(c) = \frac{(\lambda \Delta T)^c e^{-\lambda \Delta T}}{c!}, \qquad c \geq 0. \tag{3.2}$$

Next, we contend that statistics of the buffer count $b[k]$, crossing flow $e[k]$, and backlog $B[k]$ can be determined from Markov chain analysis [72]. In the infinite-state Markov chain representation, each state $i \in \{0, 1, 2, ..., \infty\}$ represents a possible buffer count, and the weights p_{ij} descrive the probabilities that the buffer count transitions from one state i to another state j during one time step (i.e. $P(b[k+1]=j|b[k]=i)$. Based on the evolution equations (3.1) and the probabilistic description of $x[k]$ (3.2), the transition probability p_{ij} can be calculated:

$$p_{ij} = \begin{cases} P_\lambda(j), & 0 \le i \le N_c, \quad j \ge 0 \\ P_\lambda(j-i+N_c), & i > N_c, \quad j \ge i - N_c \\ 0, & i > N_c, \quad j < i - N_c \end{cases} \tag{3.3}$$

We are interested in the *steady-state* probability that the buffer count $b[k]$ equals i (we call it π_i), which enables us to calculate the statistics of flow in steady state. These probabilities can be found from the transition probability matrix using classical techniques for Markov chains [72]. From the steady-state probabilities π_i, the mean backlog in steady state can be calculated as $E(B[k]) = \sum_{i=N_c+1}^{\infty}(i-N_c)\pi_i$.

The mean crossing flow $E(e[k])$ equals the inflow rate $\lambda\Delta T$ since we require that N_c is larger than λ: this condition guarantees that the buffer aircraft count is not growing, and hence the average crossing flow is same as that of the entering flow. Moreover, the variance in the crossing flow can be calculated as $V(e[k]) = \sum_{i=N_c+1}^{\infty}(N_c^2\pi_i) + \sum_{i=1}^{N_c}(i^2\pi_i) - (\lambda\Delta T)^2$.

The above calculation shows that the saturation abstraction permits the calculation of flow statistics (e.g., mean backlog and mean and variance of crossing flow) using a standard Markov chain representation. Figure 3.6 illustrates the Markov chain analysis, in particular demonstrating its use in comparing restriction of various flows entering a congested region. However, the calculation of the flow statistics does not provide us a direct insight into a restriction's effect on non-Poisson flows. Also, unfortunately, the Markov chain analysis does not easily scale to the multiple-restriction case.

3.3.3 Model Evaluation, Arbitrary Aircraft Flows

We use the saturation model to gain qualitative insight into a restriction's effect on arbitrary stochastic aircraft flows. In turn, we analyze sequences of restrictions.

Before characterizing networked restrictions, it is important to understand a restriction's impact on a single generic flow, i.e. one that may originate from another restriction and hence be non-Poisson. We expect the placement of a single restriction to smooth the downstream flows at the cost of increasing backlog. The impact should become more pronounced as the restriction is made stronger, i.e. N_c is decreased. We formalize this notion in Result 1.

Result 1 *Consider two ergodic* [†] *aircraft flows with identical statistics approaching two saturation restrictions* c_1 *and* c_2 *respectively, where restriction 1 is weaker than restriction 2 (*$N_{c_1} > N_{c_2}$*). The variances of the downstream flows* $V(e_{c_1}[k])$ *and* $V(e_{c_2}[k])$ *and the mean of backlogs* $E(B_{c_1}[k])$ *and* $E(B_{c_2}[k])$ *are related as follows:* $E(B_{c_1}[k]) \leq E(B_{c_2}[k])$, $V(e_{c_1}[k]) \geq V(e_{c_2}[k])$.

Proof: The statistics of an ergodic process can be calculated using time-averages of one of its sample sequences. We study the effect of each restriction on an arbitrary sample sequence over a sufficiently long time.

First, the relationship of the mean backlogs can be determined from the backlogs at each time step, for any sample sequence of the input process. Since $N_{c_1} > N_{c_2}$, at the initial time step ($k = 1$), if an aircraft is able to pass c_2, it must be able to pass c_1; or equivalently, if an aircraft is delayed by c_1, it must be also delayed by c_2. Therefore, at the initial time step, the backlog of c_1 is smaller than that of c_2. At subsequent times, we have the same conclusion

[†] An ergodic process is one whose statistics can be estimated from the time average of a single sample sequence [72]. It is reasonable to assume $x[k]$, $B[k]$ and $e[k]$ are ergodic random processes, as a consequence of stationary and long-term uncorrelation.

that if an aircraft is delayed by c_1 it is delayed by c_2 too, since the backlog caused by c_2 is larger than that by c_1 and less aircraft are permitted through by c_2 than c_1. This leads to the conclusion that $B_{c_1}[k] \leq B_{c_2}[k]$ for any k and thus the time-average of the backlog of each sample flow has the relationship $\frac{1}{T}\sum_{k=1}^{T} B_{c_1}[k] \leq \frac{1}{T}\sum_{k=1}^{T} B_{c_2}[k]$ for all T. From ergodicity, we conclude $E(B_{c_1}) \leq E(B_{c_2})$. Noticing the relationship between the buffer count $b[k]$ and backlog $B[k]$, we recover through an analogous argument that $E(b_{c_1}) \leq E(b_{c_2})$.

The comparison of the variance of flows crossing the two boundaries can be analyzed from the manipulation of the two flow sequences $e_{c_1}[k]$ and $e_{c_2}[k]$ over some time. Firstly, the time average of $e_{c_1}[k]$ is not less than that of $e_{c_2}[k]$ because the backlogs have the relationship $B_{c_2}[k] \geq B_{c_1}[k]$ for all k. Secondly, the conclusion that $b_{c_1}[k] \leq b_{c_2}[k]$ for all k leads to two conclusions: 1) if $e_{c_1}[k]$ is larger than N_{c_2}, restriction 2 must be saturating and thus $e_{c_1}[k] \geq e_{c_2}[k] = N_{c_2}$; 2) if $e_{c_1}[k]$ is less than N_{c_2}, $e_{c_2}[k]$ must be no less than $e_{c_1}[k]$. The inequality for the flow average and the two relationships 1) and 2) above leads to the conclusion that the sum of $e_{c_1}[k] - N_{c_2}$ for those k such that $e_{c_1}[k] \geq e_{c_2}[k] \geq N_{c_2}$ is larger than or equal to the the sum of $e_{c_2}[k] - e_{c_1}[k]$ for those k such that $e_{c_1}[k] \leq e_{c_2}[k] \leq N_{c2}$. Thus, by reducing $e_{c_1}[k]$ by 1 for those $e_{c_1}[k] > N_{c_2}$ and adding it to those $e_{c_1}[k]$ with $e_{c_1}[k] < N_{c_2}$ step by step as shown in Figure 3.3, we can reshape the sequence of e_{c_1} to $\hat{e}_{c_1}$, which is same as e_{c_2}, except possible with extra $\hat{e}_{c_1}[k] > N_{c_2}$ for some k. Therefore, we have $V(\hat{e}_{c_1}[k]) \geq V(e_{c_2}[k])$ assuming ergodicity. In the reshaping process, the second moment and hence variance of e_{c_1} is decreased monotonically. From ergodicity, we have $V(e_{c_1}[k]) \geq V(\hat{e}_{c_1}[k])$. Thus, we see that $V(e_{c_1}[k]) \geq V(e_{c_2}[k])$. $\square$

This result clearly illustrates a boundary restriction's role in flow control in a general way: it reduces the variance of downstream flow, at the cost of increasing the mean backlog. A stronger restriction produces smoother downstream flow, with more backlog in the upstream.

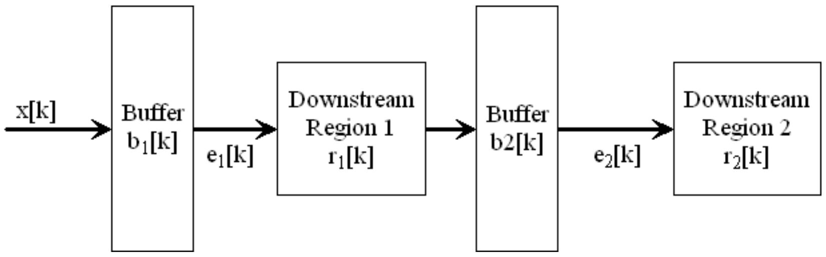


Fig. 3.2: A single flow passing through a chain of two restrictions.

The saturation model also permits evaluation of multiple coordinated restrictions, and hence facilitates development of coordinated TFM strategies. In Result 2, we discuss restriction placement for a sequence of two regions with a single flow passing through them, see Figure 3.2. This topology of aircraft flow is common in many situations, such as the flows along major routes from the Western states to the Northeastern states, which cross several Centers. An interesting question for this type of flow is where to place restrictions on the route, i.e. at the boundaries of downstream regions or further upstream. Result 2 gives insight into the combined effect of the multiple restrictions along the route, in terms of backlog and downstream congestion caused by the restrictions. For ease of presentation, we study a single flow that is restricted in several places; the result is useful in practice whenever there is a dominant aircraft flow passing through several restrictions.

Result 2 *Consider an arbitrary aircraft flow approaching a two-region chain. Placing a restriction* c_1 *at the first boundary with threshold* N_{c_1} *and another restriction* c_2 *at the second boundary with threshold* N_{c_2} *achieves the same total delay (on each aircraft)/backlog (over time) as placing a single restriction* c_3 *at the front of region* 1 *with the threshold designed as the strong restriction, i.e. as* $N_{c_3} = min(N_{c_1}, N_{c_2})$*. In fact, the crossing flow processes of the second region for the two schemes are identical.*

Proof: When $N_{c_1} < N_{c_2}$, we have $N_{c_3} = N_{c_1}$. In this case, the result is obvious since every aircraft passing N_{c_1} will pass N_{c_2} without any delay.

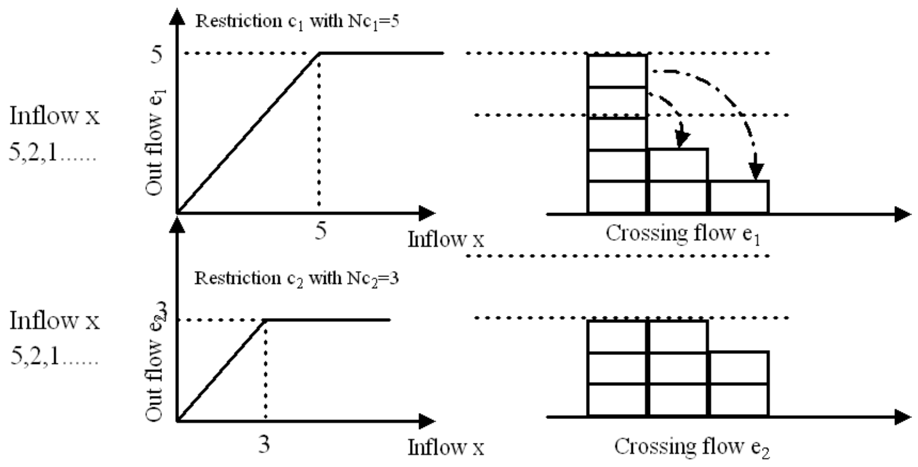


Fig. 3.3: Downstream flow variance analysis for two saturation restrictions with different thresholds.

When $N_{c_1} \geq N_{c2}$, we have $N_{c3} = N_{c2}$. Showing the equality of the delay of the single- and two-restriction cases is equivalent to showing that any aircraft crosses boundary of region 2 at the same time step for the two cases. The proof has two sides. First, we can prove that the delay of the two-restriction design is not smaller than the delay of the single restriction design, i.e. if an aircraft is able to cross boundary 2 for the two-restriction case, it must have crossed the boundary by the same time for the single restriction case. This is obvious since the delay on each aircraft would be the same for the two cases, if N_{c_1} were 0. Introduction of the first restriction for the two-restriction design can only delay aircraft, hence the delay is at least as large in the two-restriction case.

Second, we show that the delay for the two-restriction design is not worse than that of the single restriction design by contradiction. At the initial time step that the restrictions are set, the number of aircraft in region 2 in both cases are the same. Assuming the clain does not hold, there exists some time step when an aircraft is coming through boundary 2 in the single restriction case, but not yet in the two-restriction case. We denote the first aircraft satisfying the above condition as a1. All the aircraft before a1 in the flow are able to go through boundary 2 at the same time step in both cases. The only two reasons that a1 can be delayed in the two-restriction case, but not in the single restriction case, are: 1) for the two-restriction case, a1 has more aircraft in the line before it than the threshold N_{c_2}; and 2) a1 is not at boundary 2 yet because it was delayed at boundary 1. Neither is possible, as shown by the following argument: If 1) happens, we know that a1 is not the first aircraft which was delayed, since a1 has fewer aircraft before it in the buffer in the single restriction case than the two restrictions case. The controversy denies reason 1. For reason 2: this cannot happen, since the restriction at boundary 1 is weaker than that in the single restriction case. As we argued for Result 1, if an aircraft is able to coming across a stronger restriction as in the case of single restriction, it must be able to cross boundary 1 with a weaker restriction in the two-restriction case.

Thus, we have proved the delays are identical. The result for delays automatically implies the backlog result. □

For a chain of more than two regions with a single flow, Result 2 can be readily generalized to the following: placing multiple restrictions along the boundaries of the regions achieves the same total delay as placing a single restriction upstream, that is equal to the strongest of the original restrictions.

This result is interesting in that it shows for a major stream of aircraft, multiple restrictions placed along the route have the effect of splitting the delay, without increasing or reducing it. Also, placing a single strong restriction further upstream has the benefit of eliminating downstream congestion. For example, in the two-region case, using a single strong restriction before region 1 causes no congestion in region 1; when two restrictions are used, congestion may result from the backlog caused by the second restriction, as well as the increased variance of the downstream count due to the weaker restriction before region 1. In practice, when considering a single major flow passing a series of regions, the end region often has the most stringent capacity restriction since it represents the busiest region where many flows merge. For this case, Result 2 indicates that restricting flows upstream is preferable, since the downstream congestion is reduced with the same delay. It is wise to place the restriction at a boundary where the upstream region has little congestion concern, and so can absorb the backlog. In some cases, if such a upstream region that can hold all the backlog does not exist, we need to design multiple restrictions along the route to split up the delay and in consequence the backlog.

Result 2 also makes clear that, in contrast to current practice, multiple restrictions acting on a stream must be designed together because the upstream restriction impacts the delay/backlog caused by the downstream one. For instance, it is unwise to first place a restriction downstream and then place another one upstream to account for the backlog caused by the downstream restriction: the upstream restriction will change the backlog caused by the downstream restriction.

One further note about the analyses of the saturation abstraction is needed. Our results are a simplification of reality in that we have assumed stationarity in the process of aircraft arriving at boundaries, while in fact the arrival rates are time-varying. Fortunately, several of the above results—namely those concerned with mean backlogs and delays—generalize to the non-stationary case. Also, we reiterate that the flows during busy periods are often indeed stationary; if not, they can almost always be captured as arrival processes whose rates are themselves Markovian stochastic processes (e.g., as Markov-modulated Poission processes). The above analyses fully generalize to this case.

3.3.4 Canonical Example: Restricting Flows into PHL

Using the saturation restriction model, we have found that multiple restrictions along a stream can be used to split delays/backlogs. This suggests that multiple restrictions may be effective in partially pushing backlog upstream near complex terminal areas, such as the Philadelphia International Airport (PHL). Here, for illustration, we have evaluated the impact of using multiple restrictions on the entire flow arriving at PHL.

We use the arrival times of all aircraft coming to PHL in January 2006 as the inflow. Say that we wish to limit the flow into the airport to 6 planes per 20 minutes (e.g., because of arrival capacity requirements or constraints on nearby airspace). We have considered three combinations of the restriction strengths N_1 and N_2 on the flow that each achieve the requirement. For each case, the mean backlog and distribution of backlogs due to each restriction have been found (Table 3.1 and Figure 3.4). As expected, the total backlog for the three cases are the same. A more stringent restriction upstream reduces the backlog caused by the second restriction. This informs us that designing a stringent restriction at a upstream region will succeed in moving the backlog further upstream. A good choice would be to locate a upstream region with little traffic congestion concern, and place all the backlog there. If such a region does not exist, we can split the backlog carefully among some upstream regions (the $N_{c_1} = 7$, $N_{c_2} = 6$ case).

N_{c_1}	N_{c_2}	E_{B_1}	E_{B_2}
10	6	0.0542	2.7852
7	6	0.9380	1.9014
6	10	2.8394	0

Tab. 3.1: The mean backlog of the two regions for three different restriction settings.

3.3.5 Another Example: Which Flow Should Be Restricted?

The saturation abstraction also facilitates restriction design for other flow topologies. Here, we illustrate through an example that the saturation model helps us to choose which of multiple flows

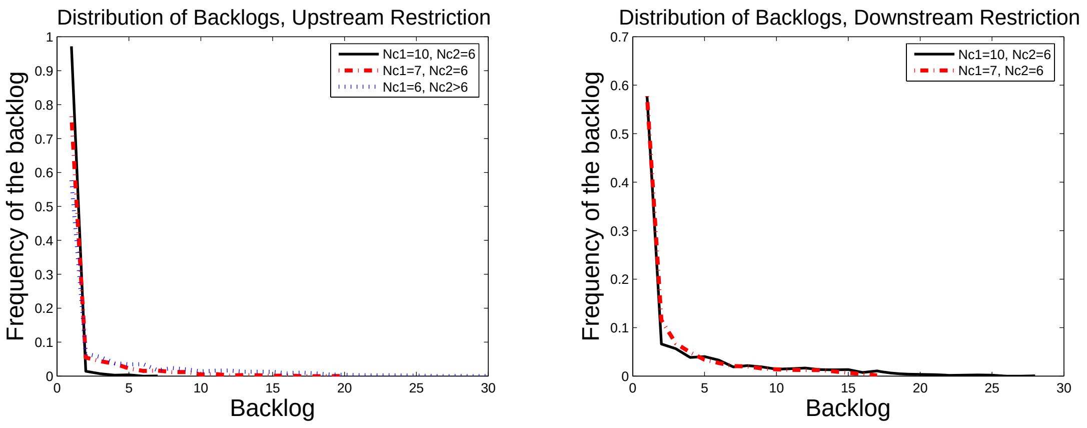


Fig. 3.4: Distribution of backlogs for three different combinations of restrictions. The two restrictions can be carefully selected ($N_{c1} = 7$, $N_{c2} = 6$) to split the backlog between the two regions. These distributions were computed using actual arrival data from January 2006.

entering a congested region to restrict.

Let us consider the effect of an unexpected or incorrectly-predicted stratus event at San Francisco airport (SFO). Stratus at SFO severely limits the arrival capacity of the airport, because only one of two parallel runways can be used. In cases where stratus impacts the airport unexpectedly, or where the time at which the stratus will clear is underestimated, the rate of already-airborne traffic approaching the airport may exceed the the permitted traffic. If these approaching flows are not restricted upstream, backlogs 10-15 aircraft in the terminal airspace may result at certain times of day (see Figure 3.5). Given the complexity of the airspace around SFO, such a backlog can unacceptably increase controller workloads, and hence upstream en route management is needed.

Aircraft traffic approaches SFO along several jet routes, along which restrictions can be placed. Abstractly, we can roughly separate the traffic flow approaching SFO into Northbound, Southbound, and Westbound traffic[‡], where the Westbound flow is the major one (see Figure 3.5). If an unexpected stratus event necessitates placement of a restriction upstream, the Oakland ARTCC in

[‡] The frequency of Eastbound traffic is small enough to be negligible.

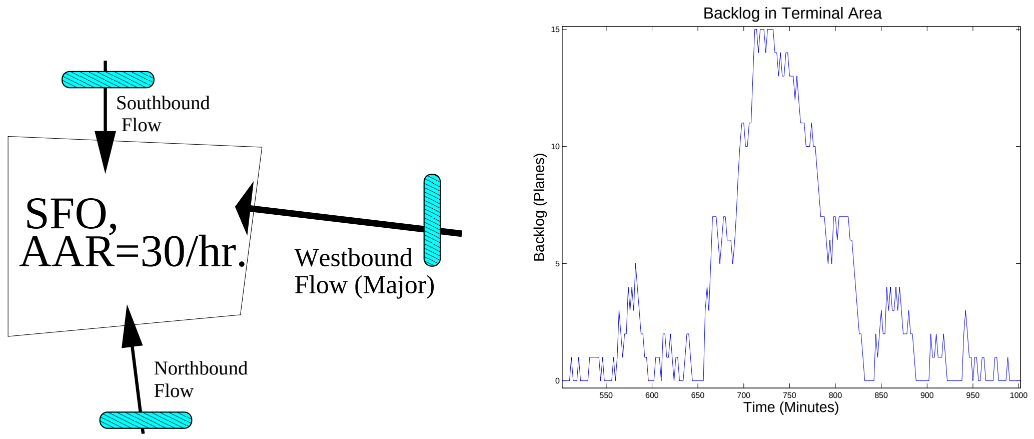


Fig. 3.5: a) Abstract illustration of traffic flows entering SFO and arrival capacity during stratus event. b) Backlog at SFO during stratus event assuming flows are not restricted upstream; backlog was computed using actual arrival data from June 1, 2006.

coordination with the ATCSCC must decide which flow(s) to restrict.

The Markov chain analysis of the saturation restriction indicates that restriction of the major aircraft flow is most effective in smoothing the downstream flow with minimum backlog and delay (see Figure 3.6). In Figure 3.7, we have compared restriction of the major (Westbound) flow with restriction of the Northbound flow, for traffic approaching SFO on a particular day (June 1, 2006). These experiments bear out that restriction of the major flow achieves sufficient decrease of terminal-airspace backlog with lower upstream backlog and delay. Thus, from the perspective of minimizing average delay and preventing upstream congestion, we find that restriction of the major flow is most effective, as predicted by the saturation restriction model.

3.4 A Linear Abstraction

Linear abstractions of boundary restrictions are very appealing because explicit expressions of flow statistics and cross-statistics can be developed and, in consequence, multiple restrictions can

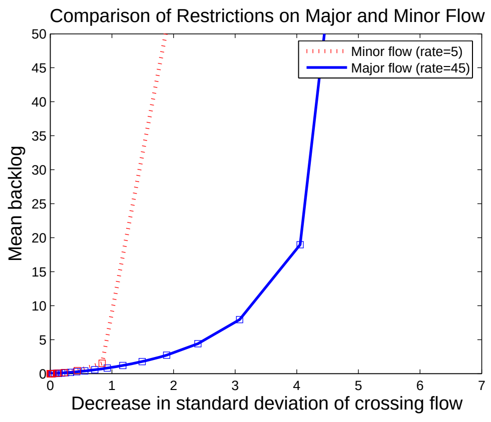
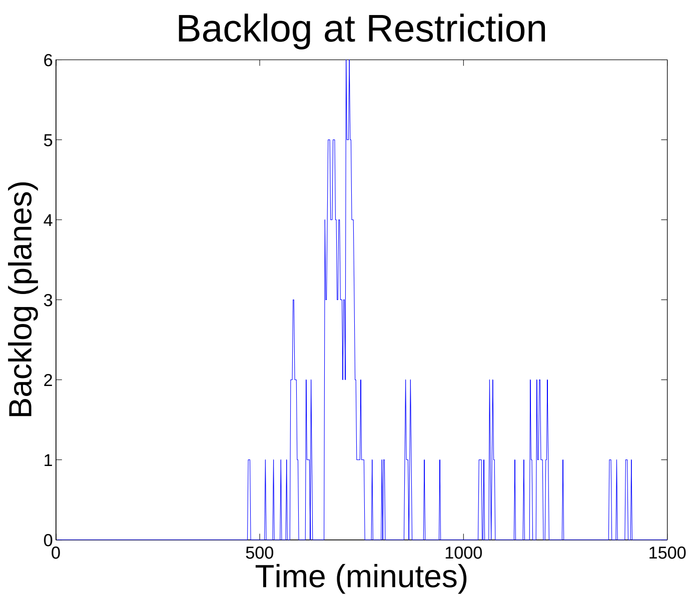


Fig. 3.6: Comparison of restrictions on major and minor flows using the saturation model. We find that restriction of the major flow is more effective in decreasing downstream flow variability, for a given backlog.

be designed to shape Sector counts. Here, we show that a linear model can capture the intrinsic characteristics of flow restrictions, and yet is suited for multi-region analysis.

3.4.1 Model Description

Our discrete-time linear abstraction for a boundary restriction is shown in Figure 3.8. Between times $k\Delta T$ and $(k+1)\Delta T$, the number of aircraft allowed to cross the boundary $e[k]$ is calculated as a fraction (denoted by a) of the aircraft in the buffer at the previous time step plus a constant c. In Section 3.4.3, we will argue that such a model can be obtained through a stochastic linearization of the saturation model. For the linear model, we find it convenient to explicitly represent traffic in downstream regions, since statistics of these traffic flows can be obtained. For simplicity, let us assume that each aircraft takes a fixed number of time steps say L to cross the downstream region. (This is often quite reasonable for an en route restriction, where each aircraft in the flow is usually traveling at roughly the same speed.) In summary, the dynamics of the linear abstraction are the following:



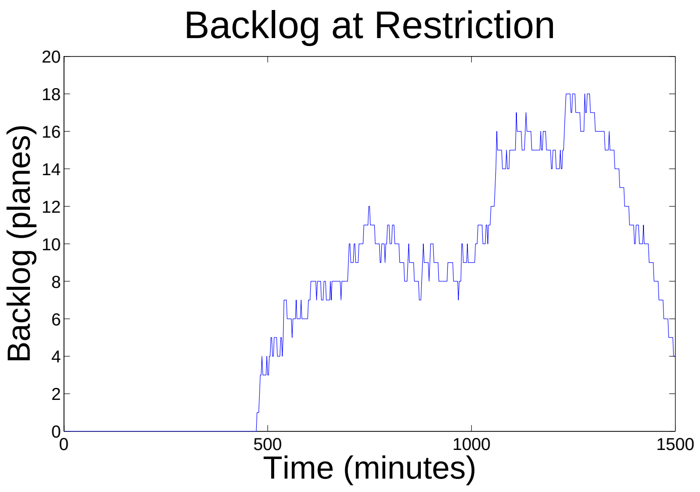


Fig. 3.7: Upstream backlog caused by restriction of the a) major flow and b) minor flow, for the purpose of limiting backlog at SFO during stratus event to 9 aircraft.

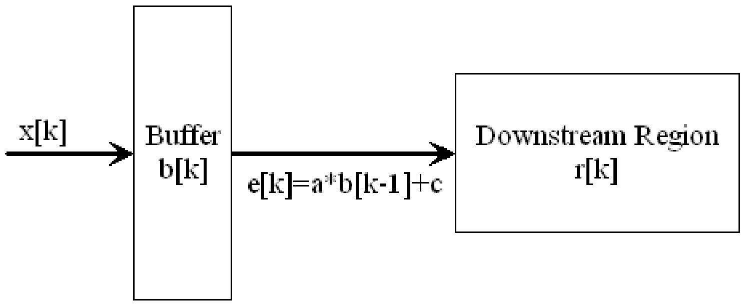


Fig. 3.8: Linear boundary restriction scheme.

$$
\begin{aligned}
e[k] &= ab[k-1] + c \\
b[k] &= b[k-1] + x[k] - e[k] \\
B[k] &= b[k-1] - e[k] \\
r[k] &= \sum_{k=1}^{L} e[k-L+1]
\end{aligned}
\tag{3.4}
$$

where $b[k]$, $B[k]$, and $e[k]$ are as defined before, and $r[k]$ is the number of aircraft in downstream region, or downstream region count, at time k.

3.4.2 Analysis of the Linear Abstraction with Poisson Input

We are concerned with two measures that indicate the performance of a boundary restriction, namely downstream region count and upstream backlog [86]. For a Poisson input, the dynamic and steady state statistics of these measures can be calculated from the linear system representation, using the classical two-moment analysis of linear systems driven by random processes. Here, let us present the steady-state mean and variances of the backlog, downstream region count, and crossing flow, when the input process is Poisson with rate λ. The mean and variance of the backlog $B[k]$ caused by the restriction are $E_B = \frac{1}{a}(\lambda\Delta T - c) - \lambda\Delta T$ and $V_B = \frac{(1-a)^2}{1-(1-a)^2}\lambda\Delta T$; the mean($E_e$) and variance ($V_e$) of the crossing flow are $E_e = \lambda\Delta T$ and $V_e = \frac{a^2}{1-(1-a)^2}\lambda\Delta T$; and the mean ($E_r$) and variance ($V_r$) of the downstream region count are $E_r = L\lambda\Delta T$ and $V_r = \left\{\frac{La^2}{1-(1-a)^2} + \frac{2a^2}{1-(1-a)^2}\sum_{k=1}^{L-1}(L-k)(1-a)^k\right\}\lambda\Delta T$.

Let us briefly interpret these results, from the perspective of designing restrictions. First, we note that $\lambda\Delta T$ has to be smaller than C/L (where C is capacity of the downstream region) to be able to reduce downstream congestion while not causing growing delay in the upstream. The parameters a ($a \leq 1$) and c are responsible for the downstream and upstream performance. A decrease in a decreases the variance of the downstream region count, but increases the mean and variance of backlog, as shown in Figure 3.9. An increase in c reduces the mean of the backlog, and does not affect the statistics of the region count. Thus, based on these observations, it is tempting to design a to make the variance of region count small enough, and then choose c to make the mean backlog arbitrarily small. However, such a restriction is unachievable in practice, because it requires movement of more aircraft into the downstream region than are in the buffer. For a restriction to be achievable, we need c to be small, and in this case, the boundary restriction that we proposed also reduces the downstream congestion with the cost of upstream backlog.

Specifically to obtain a good performance while using an achievable restriction, we need to choose parameters a and c carefully. Let us consider the following two cases:

1) Large aircraft inflow. For large λ ($\lambda\Delta T$ close to $\frac{C}{L}$), the prevention of downstream capacity violation is the focus of the design. Suppose no restriction is placed at the boundary; the variance of crossing flow is $\lambda\Delta T$, which is very likely to cause congestion. By placing a restriction with small a, we can significantly reduce the variance of region count (in another words, smoothen the aircraft flow). In this case, we can have an achievable restriction even with a moderate c, since the number of buffered aircraft is large, and so the buffer will not be overdrawn even with moderate c.

2) Small aircraft inflow. For small λ ($\lambda << \frac{C}{L\Delta T}$), it is a rare event that downstream congestion occurs. In this case, we can choose a to be large (close to 1) to reduce the backlog. Essentially in this case, we are using little control, so c must approximately be 0.

The above analysis indicates that the linear restriction can be chosen to resemble the saturation restriction, as shown in Figure 3.10. With small inflow λ, we design large a and small c to resemble the linear-increasing region of a saturation restriction; with large λ, we design small a and moderate c to resemble the constant region of the saturation restriction. Using this *stochastic linearization*,

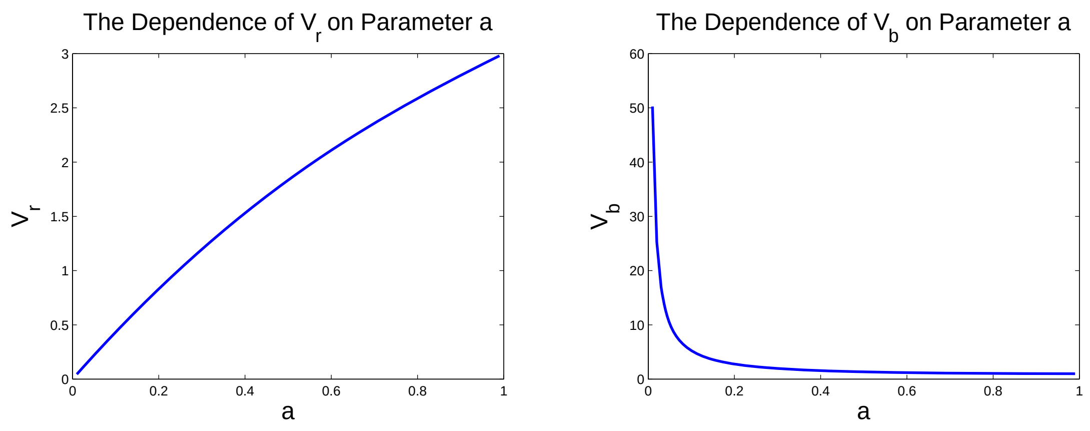


Fig. 3.9: Dependence of downstream region count's variance (V_r) and the backlog's variance V_B on the linear model parameter a. Again, we see a tradeoff between backlog and downstream variance.

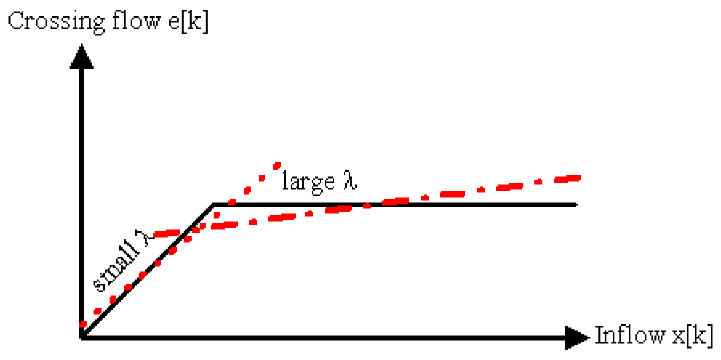


Fig. 3.10: Stochastic linearizations of saturation restriction.

we can approximate the saturation restriction with a linear abstraction, and hence gain significant advantage in its tractability.

3.4.3 Restriction Design

The similarity between the linear model and the saturation model provides us with an approach for restriction design. As developed in Section 3.4.2, the linear model allows the explicit evaluation of downstream count and backlog. The parameters of the linear model a and c can be calculated from N_c of the saturation model, so as to match the variance of the crossing flow and the mean backlog for the two models (assuming a Poisson flow). We thus can use the linear model to check the performance of a saturation model and hence of an actual restriction (e.g MINIT or MIT). Therefore, using the linear abstraction, we can design restrictions to meet downstream region-count goals.

As an example, we study the performance of some MINIT restrictions acting on a Poisson inflow with rate 0.2 (see Table 3.2). Here, we assume that the discrete time interval is $\Delta T = 20$ and that each airplane takes 3 time steps to cross the downstream region. This example shows that the analysis of downstream region count variance permitted by the linear abstraction can help us design appropriate MINIT restrictions. Also, the values of the parameters a and c in the table verify that the linear restriction abstractly represents the saturation restriction: a is close to 1 and c is close to 0 when the inflow $\lambda\Delta T$ is small compared to N_c; and a is close to 0 and c is moderate when the inflow $\lambda\Delta T$ is comparative to N_c.

MINIT	N_c	E_B	V_e	a	c	V_r
2	10	0.004	3.950	0.994	0.021	11.95
4	5	1.156	1.688	0.593	0.940	8.365
5	4	14.713	0.160	0.077	2.561	3.920

Tab. 3.2: Matching between the saturation model and linear model aids in MINIT restriction design for the purpose of reducing variance in regional aircraft counts. Here, $\Delta T = 20$, $\lambda = 0.2$, $L = 3$, $\lambda\Delta T = 4$.

For the above case 2, trajectories of the region count and upstream backlog for the saturation restriction and corresponding linear restriction are shown in Figure 3.11. The plots show that the linear abstraction captures the characteristic behavior of restriction, that downstream variability is reduced at the cost of upstream backlog.

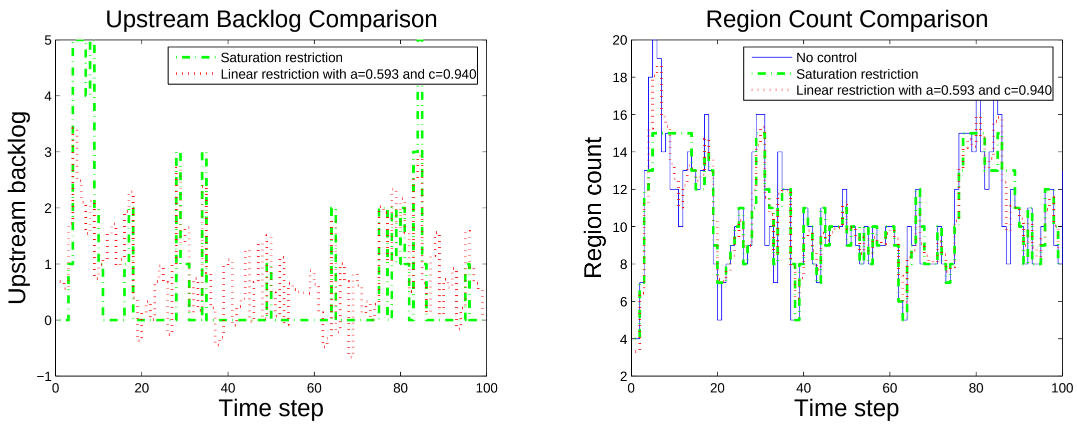


Fig. 3.11: Upstream backlog and downstream region count upon use of saturation restriction model and corresponding linear restriction model, assuming mean inflow rate $\lambda\Delta T = 4$, $N_c = 5$, and $L = 3$.

The simulation also indicates some limitations of the linear abstraction. Since we are matching the two models only through two steady state statistical measures, details of downstream region count and upstream backlog trajectories have some differences. The differences result from the fact that the saturation restriction model permits all aircraft to come through at low traffic times, while for the linear restriction, only a fraction of aircraft can come through even at times of low traffic. However, since the two steady state statistical measures mostly capture the important characteristics of flow under restriction, we believe this abstraction is often valid, which provides us with a lot of flexibility in design.

3.4.4 Analysis of Multiple Restrictions

When considering design of multiple restrictions, the classical analysis of linear systems with stochastic inputs allows us to easily determine the relationship between the linear restrictions and the statistics of flows. Thus, using the equivalence between the saturation model and linear model, the effect of multiple restrictions on statistics of flows in multiple regions is made analyzable. Specifically, we suggest the following procedure for analyzing a network of m restrictions:

1) Transfer possible combinations of restrictions durations to corresponding saturation parameters.

2) Calculate the steady-state crossing flow variance and mean backlog for each N_{c_i} using the standard Markov chain analysis given in Section 3.3.2.

3) For each restriction, match the two statistical measures with those of the linear restriction, by properly choosing the values of a_i, c_i for each i (as described in Section 3.4.3).

4) Calculate flow statistics for the entire network (e.g., downstream region counts spreads) using the linear models.

5) Design the proper MINIT restrictions that satisfy the performance requirements of the flow statistics.

The performance analysis for multiple regions using the linear model is similar to that for a single region. In particular, we can develop a system of difference equations for the evolution of the network, and analyze system output statistics (backlog and region count statistics) with the knowledge of input flow statistics, using the classical 2nd-moment analysis of a linear system driven by a stochastic process [107]. We do not give a full formulation in the interest of space, and since we will suggest an even simpler model for preliminary network TFM design.

Let us illustrate optimal restriction design using the linear abstraction, for the important case of two regions traversed by a single major flow. As discussed above, for possible combinations of MINIT restrictions at the two boundaries, we can calculate the corresponding saturation model parameters N_{c_1} and N_{c_2}, and then map these to linear restriction parameters a_1, c_1, a_2 and c_2 by equivalencing the statistics of flow variance and mean backlog. Thus, we obtain a model with two linear restrictions driven by a single flow, as shown in Figure 3.12. Once the linear restriction parameters have been obtained, the difference-equation description for the two-region case can be

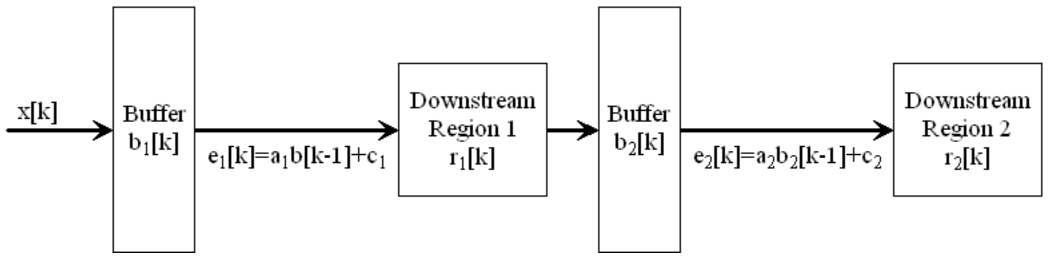


Fig. 3.12: Linear boundary restriction model for a chain of regions.

systematically constructed, and in turn first- and second-order transient and steady-state statistics for region counts, backlogs, and delays can be obtained. In the interest of space, we omit these expressions. This example illustrates that the linear restriction 1) permits study of transient aspects of the flow dynamics, and 2) allows not only analysis but also design of multiple restrictions.

Noting the ability of the linear model to evaluate the statistics of region count, we discuss further the result stated in Section 3.3.3 that a single restriction placed before the first region in a chain achieves better performance than two restrictions placed in front of both regions. Using the linear abstraction, we can revisit the result, from the perspective of downstream region counts. In order to compare the cases, we enumerate combinations of N_{c_1} and N_{c_2} and calculate the statistics of the two downstream region counts and the upstream backlog using the linear model. This analysis shows that using $N_{c_1} \geq N_{c_2}$ produces nearly the same total backlog as placing a single restriction of strength equal to N_{c_2} in from of the first region, while significantly increasing the variance in the number of aircraft in region 1 because of the backlog caused by restriction 2. The analysis confirms that placing a single restriction at the front reduces downstream variability in aircraft counts.

In summary, the linear abstraction captures the intrinsic impact of a boundary restriction on flows, and yet permits computation of steady-state and dynamic network flow statistics, including regional aircraft count statistics. Because the linear model is amenable to analysis of dynamics, we can also use it to evaluate coordinated TFM strategies under time-varying uncertainties including weather events with uncertain duration/scope (e.g., fog at SFO). This direction will be pursued in future work.

3.5 Algebraic Buffer Model and Network-wide Management Problem: Brief Introduction

The final model that we develop is a highly-abstracted algebraic description of a flow-management restriction. This simplistic model is motivated by the need for systematic *design* of flow-management strategies for an arbitrary network. Such design problems are in their essence *decentralized controller design problems* (see [28] for background, and [5, 29] for our efforts), and remain extremely challenging. We thus take the perspective here that such design is best achieved using the simplest plausible model, with more accurate models subsequently being used to evaluate and refine the design. With this goal in mind, we develop a simple algebraic model for the dynamics of a restriction, and describe its application in network-wide flow control design. In the interest of space, we only overview the model and give a few results here.

3.5.1 The Algebraic Model for a Single Restriction

At the most abstract level, a restriction simply serves to decrease downstream variance at the cost of upstream backlog. Very crudely, we can assume a simple linear dependence of the downstream variability and upstream backlog in steady-state on the strength of the restriction; such a linear dependence is roughly borne out by comparison of upstream backlog and downstream variance in the saturation abstraction. Specifically, we use a single parameter $a \in [0, 1]$ to describe the strength of the restriction. Assuming an input process with rate λ and *variability* v, we model the resulting steady-state backlog as $B = \gamma a \lambda$ (where γ is a scale parameter) and the variability of the downstream flow as $w = (1 - a)v$.

3.5.2 Network Model

We use the highly abstracted algebraic model for a restriction to pose a network-wide flow-management design problem. Our aim here is formulate the flow management problem in such a way that graphical insights into the structure of good flow management designs can be developed (for instance, to answer the question of whether it is better to restrict long traffic flows or cross traffic). Also, as throughout this work, we focus on designing flows (rather than guiding individual aircraft) using the network model, with the motivation that practical management strategies can be designed in this way.

In this network model, we are concerned with the characteristics of aircraft traffic just before and after "boundaries" in the airspace, including waypoints between Sectors en route and control points for arrival and departure traffic, see Figure 3.13). We assume that the network has n boundaries in total, labeled $1, \ldots, n$. Each boundary i is assumed to have a mean flow rate λ_i through it, which is not affected by the management scheme. We use the algebraic model for a restriction from Section $V.A$ to capture flow management at each boundary. That is, we represent the variabilities of flows before and after each boundary i (as v_i and w_i, respectively), as well as the backlog (B_i) caused by the boundary restriction. The variabilities are related as described above, by $w_i = (1 - a_i)v_i$, and the backlog is given by $B_i = \gamma a_i \lambda_i$. Here, the strength of the restriction a_i is a design parameter in some cases (e.g., for a MINIT restriction), and pre-set in other cases (e.g., for a restriction representing the arrival capacity or rate in a terminal area). Let us define $\mathcal{A}$ as the set of parameters a_i that are designable.

We now have left to describe how the flows merge and split within regions (between boundaries) and how they enter the airspace. By doing so, we will model the variability v_i of the aircraft flow approaching each boundary. Let us consider the following two cases:

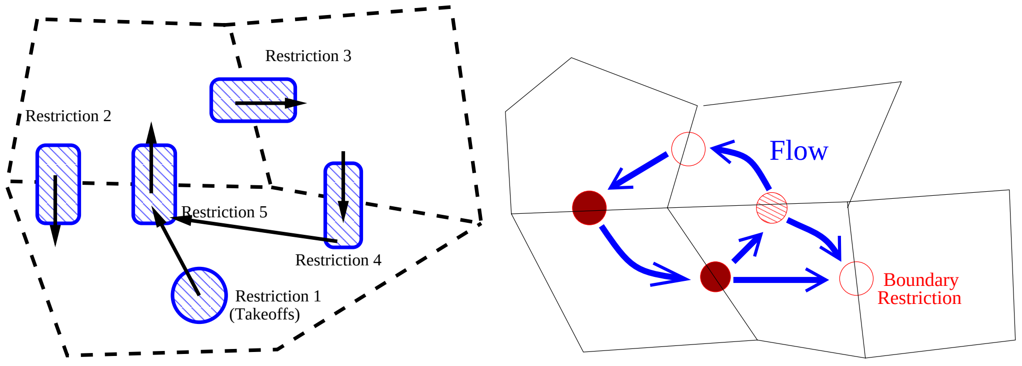


Fig. 3.13: *Upper:* Diagram of the network-wide flow-management model. *Lower:* An optimal restriction topology for a network with five congested regions–dark shading indicates a strong restriction, and light shading a weak one. Notice that the strongest restrictions are placed upstream of multiple congested regions.

1) For flows entering the airspace from the ground (take-off flows), the variability v_i is well-modeled as the variance of the number of arrivals in a Poisson process of rate λ_i during a unit interval, i.e. the variance of a Poisson random variable with mean λ_i.

2) Flows approaching other boundaries are formed by splitting/merging of flows in a region in the airspace (including flows entering the airspace from the ground, and flows that have passed through other restrictions). Since these various flows that combine to form the approaching flow are typically independent processes, it is sensible that the approaching flow's variability is a combination of the variabilities of these flows. In other words, we contend that the variability v_i can abstractly be written in the form $v_i = \sum_{j=1}^{n} w_j g_{ji}$, where the constants g_{ji} can be obtained by determining how flows merge/split within the region. We notice that the g_{ji} specify a connection network for restrictions: for a particular pair $(i,\ j)$, $g_{ji} > 0$ implies that there is a flow from restriction j to restriction i.

We note that our abstract model is similar in structure to the *Eulerian Traffic Flow Model* developed in [108], but represents flow variabilities rather than densities and explicitly represents design parameters.

The design problem of interest is to set the parameters $a_i \in \mathcal{A}$ to get desirable variability in flows or regional counts, while maintaining small backlogs (so that total counts in upstream regions do not exceed thresholds, and aircraft are not subject to long delays). With just a little effort, one can show that the variabilities decrease mononotonically as the restrictions are strengthened (a_i are increased), while the backlogs increase monotonically with a_i. Thus, the design goal is to appropriately trade off variability with backlog, by setting $a_i \in \mathcal{A}$. A natural aim is to choose a_i to optimize a performance measure that is based on the variabilities and backlogs. For instance, a cost measure of interest might aim to capture the total impact of the control on the aircraft counts in a couple critical regions, by combining the variabilities of flows in the region with backlogs caused by restrictions on flows out of the region. Many such measures are well-approximated as being quadratic in the backlogs B_i and variabilities W_i. We have addressed this design problem, as one application of a comprehensive methodology for decentralized controller design, in [5]. Our design methodology yields insight into the topology of the optimal restriction placement, and hence provides a starting point for more refined designs. In the interest of space, we omit details from this chapter, but illustrate the design using a small (five-region) example, see Figure 3.13.

3.6 Conclusion

Air traffic flow management in the NAS is complex and interrelated. We have taken the prespective that good flow management strategies must be designed at a network level, by taking into consideration traffic in multiple Centers in the presence of uncertainty and using realistic management capabilities. In order to obtain optimal network-level flow management strategies, we emphasize the full understanding of restrictions' impact on generic traffic flows, with the aim of developing restriction abstractions for analyzable network evaluation and optimization. In this chapter, we examine four abstractions, namely, the detailed queueing model, the discrete-time saturation model, the dynamic linear model, and the algebraic linear model. The queueing model best approximates the details of restrictions, and permits some analysis of the flow statistics with a Poisson inflow. We suggest the saturation restriction model as an approximation of the queueing model with the ability of identifying both the upstream and downstream flow statistics of a single restriction under a Poisson inflow. This model can be used to study restrictions in simple topologies, but lacks the scalability for a full network analysis. Furthermore, we introduce a stochastic linearization of the saturation restriction model. This linearization has the advantage of 1) finding explicit expressions for the flow and region-count statistics; 2) only requiring the statistics of inflow rather than a Poisson inflow; and 3) permitting a network-level analysis and design. Finally, we develop a highly-abstracted algebraic linear model, and pose the problem of network-level optimization of restrictions using this model.

4. SENSITIVITY OF NATIONAL AIRSPACE SYSTEM PERFORMANCE TO DISTURBANCES: MODELING, IDENTIFICATION FROM DATA, AND USE IN PLANNING

We study of the sensitivity of traffic flow management (TFM) performance in the United States National Airspace System (NAS) to disturbances, such as weather-driven capacity/flow variations and gradual changes in route usage. We make the argument that these sensitivities can be roughly computed using queueing models for flow-management actions, and so postulate that performance becomes much more sensitive to disturbance in congested airspace. Next, historical data on the sensitivity of TFM performance to weather and other uncertainties is used to support the postulate of increasing sensitivity with increased congestion. Finally, we put forth the idea that performance sensitivity information can aid in planning TFM (e.g., planning airspace reconfiguration or aircraft routing), by showing that optimally- or well- designed queue banks and queue networks have very special sensitivity structures and hence that planning actions should aim to achieve these structures.

4.1 Introduction

The United States National Airspace System (NAS) is continuously subject to alteration. In the short term, disturbances including convective and winter weather, runway/airport maintenance, and security-related closures lead to changes in flows and capacities. Over a longer period, traffic densities increase at disproportionate rates at different locations in the airspace, while improvements/realignments in the traffic flow management (TFM) system modify both traffic patterns and capacities. While each of these variations in flows or capacities may impact the NAS performance, it is well understood that some have much more acute impact than others. For example, Sridhar and coworker's empirical tool for predicting delays from weather and traffic counts, the *weather-impacted traffic index* (WITI), demonstrates that severe weather in particular regions (the Northeast and Upper Midwest) have disproportionate effect on delays [109, 110]. In the same vein, improved flow-management strategies at critical airports or en route locations can significantly reduce delays throughout the airspace [93].

The observed hyper-sensitivity of NAS performance to *disturbances* (for our purposes, capacity and flow-density changes) at certain critical congested locations suggests that TFM planning should focus on such hyper-sensitive locations. We contend that new strategies for flow management—including local strategies such as airspace flow programs (e.g., [111]) and reconfiguration (e.g., [112]), as well as radical global alterations such as use of free flight [113]—must ameliorate this hyper-sensitivity to be effective. In particular, reducing hyper-sensitivity both allows the system to better withstand disturbances, and as we shall argue in Section 4.4, helps reduce congestion and delays overall. To implement these TFM strategies in the most effective way, we thus need to analyze their impact on NAS performance-measure sensitivities (e.g., delay or backlog sensitivities). While planning in the air traffic flow management system already implicitly accounts for sensitivities, in the sense that locations that are perceived to be bottlenecks during e.g. inclement weather are allocated more control resources, disturbance sensitivities have not been characterized in terms of traffic parameters nor systematically used for airspace planning. The purpose of this chapter is to introduce the notion of disturbance sensitivity in TFM planning. We do this in two steps. 1) We give a systematic methodology for modeling the sensitivity of NAS performance to disturbances/modifications and identifying these sensitivities from data. 2) We make the argument that sensitivities can inform evaluation and design of various strategies, because sensitivities throughout the NAS assume a special structure for optimal or high-performance designs.

To delineate the contributions to TFM planning made by our study, let us briefly connect it to the existing literature. First, our study builds on recent efforts to characterize the impact of weather on traffic flows and TFM performance [109,110,114,115]. From this evaluation standpoint, our work shows that sensitivities are a means for understanding the impact of weather disturbances, and gives a causation between congestion and disturbance sensitivity. Second, we notice that there is an extensive literature on developing optimal strategies for traffic flow management, including for such diverse tasks as airspace configuration, route planning (including under inclement weather conditions), terminal and en-route flow restriction, and capacity reassignment (e.g., [93, 94, 111, 112,116–120]). While these various optimization algorithms each help in mitigating congestion, the air traffic system is so complex and extensive that practical global strategies for flow management are difficult to evaluate, let alone optimize. This is not least because characterization of useful performance measures in the presence of weather and other uncertainties is complicated. Our effort here is not meant to supplant the optimization tools developed in the literature, but rather to show that sensitivities are useful measures that can help in testing and improving flow management strategies.

The approach that we take for estimating the sensitivity of NAS performance measures to disturbances is based on queueing theory. Queues have long been used to model numerous aspects of the NAS (including departure and arrival processes, aspects of surface operations, en route flow restriction, and long-term planning, among others) [86, 103, 104]. Here, we put forth that the impact of disturbances on NAS performance can be characterized by considering the sensitivities of queue performance measures to capacity and inflow changes. In turn, we further show that flow management should be planned to *equalize* scaled sensitivities at the design locations in order to optimize overall NAS sensitivity. We motivate and validate this approach, as follows:

- In Section 4.2, we review the use of queueing models in air traffic flow management, and present the sensitivity analysis for the prototypical M/D/1 queues. In particular, we find the sensitivity of backlogs/delays to congestion changes, showing that increased congestion leads to much higher sensitivity to disturbances.

- In Section 4.3, we give evidence of the increased sensitivity to disturbances in high-congestion locations using historical data as well as relevant literature, to validate the modeling approach. This validation also clarifies how sensitivities can be inferred or compared from data.

- In Section 4.4, we consider planning of flow management strategies from the perspective of the sensitivity analysis. Specifically, we argue that efforts that *equalize* sensitivities improve NAS performance, and show how this idea can be used for such tasks as controller workload redistribution and route re-planning.

4.2 The Queueing Model

Queueing models have been widely used to represent various en route and terminal area management restrictions acting on air traffic flows [86,103,104]. An advantage of using queueing models is that they provide a systematic way to analyze traffic flow statistics and hence evaluate the performance of management strategies, in the presence of uncertainties [95]. As an example, in [86], the performance measures (e.g., average delay/backlog) of various en route TFM strategies (e.g., MIT/MINIT, Time-based Metering, and Intelligent Control) are compared assuming a typical Poisson flow. In that work, MINIT and MIT restrictions are modeled as M/D/1 queues (Poisson input, deterministic single server). Furthermore, TFM actions on multiple Centers or NAS-wide can be viewed as a network of queues. Very similar queueing models have been developed for arrival and departure as well as surface traffic [103, 104]. In [2], we considered the design of both en route and terminal area TFM restrictions in a multi-Center region to achieve desired performance. By capturing the key features of the detailed queueing model in terms of flow statistics, we came up with more abstracted models (e.g., saturation model, stochastic linear model, and algebraic linear model), and by using these simplified models, we posed the NAS-wide TFM restriction design problem as a tractable constrained optimization problem. The artical [2] is especially important to our current development, since it shows that the NAS is well-represented as a network of capacitated queues.

In this chapter, we use the idea that the NAS can be viewed as a network of queues to inform longer-range planning of traffic flows and flow management. To begin, we use the queueing model to analyze the sensitivity of traffic delay/backlog to a disturbance, which alters congestion due to the change of either traffic flow rate or capacity. The disturbance on congestion may be either positive or negative: congestion may increase due to unexpected weather events; and it may decrease due to effective planning, e.g., airport construction, route re-planning, improved management facilities, and increased human or facility resources. Before pursuing the disturbance sensitivity analysis, let us give a description of the prototypical queueing model used for our analysis.

4.2.1 Model Details

Broadly, we consider a stream of air traffic flow entering/leaving a region (e.g., entering a Sector, at a fix, or arriving at an airport). A TFM action (e.g., an en route program such as an AFP or spacing for arrivals at airports) incurs backlog and delay on aircraft, while shaping the crossing flow. The TFM action can very often be modeled as a single-server queue: each incoming aircraft waits in line at the boundary, and the first one in the waiting list is served for some time (e.g., passes through the AFP region) and leaves the boundary. In particular, M/D/1 queues (deterministic single sever queues) are widely used to model various TFM actions (en route, take-off, landing, taxi-in, and taxi-out). This is because the actions generally ensure the time/distance difference between two adjacent crossing aircraft, and this fixed difference can be reflected in modeling through a deterministic constant serving time, with the assumption that each aircraft has a similar speed. Because of the wide applicability of M/D/1 queueing models in modeling air traffic, we use this model for our analysis here, though similar sensitivity computations can be obtained for other queueing models.

Specifically, here we model the incoming air traffic flow as being a Poisson process with rate λ. This memoryless stochastic representation is representative of many aggregate flows in the airspace, in particular ones that are mixtures of several independent flows, see [86] and [82] for a justification. Hence, in a time interval T, the distribution of the number of airplanes approaching is given by the Poisson Probability Mass Function:

$$P(N = N_c) = \frac{\lambda T e^{-\lambda T}}{N_c!}, N_c = 0, 1, 2, ... \tag{4.1}$$

Moreover, we model a boundary action/restriction as having a (deterministic) service rate λ_c, or in other words a serving time of $\frac{1}{\lambda_c}$ (see Figure 4.1). This model for example could be used to represent a $\frac{1}{\lambda_c}$-MINIT restriction or an airport arrival process with AAR of λ_c. We refer to λ and λ_c as the **inflow rate** and **restriction stength** (or **capacity**) of the queue, respectively.

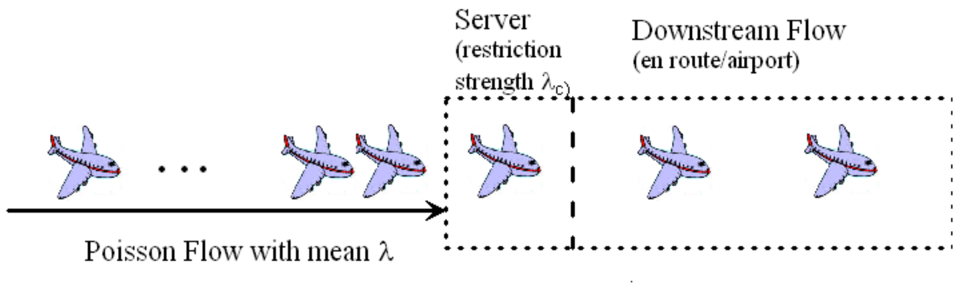


Fig. 4.1: Queueing Model

4.2.2 Sensitivity Analysis

Based on the M/D/1 queue representation, we can find the statistics of performance measures such as backlog and delay imposed by a TFM action [86]. The mean backlog is

$$E(B) = \frac{\lambda^2}{2\lambda_c(\lambda_c - \lambda)} \tag{4.2}$$

and the mean delay is

$$E(D) = \frac{\lambda}{2\lambda_c(\lambda_c - \lambda)}. \tag{4.3}$$

Now let us study the sensitivity of delay/backlog with respect to congestion. To do so, let us define the **congestion level** ρ as the ratio $\frac{\lambda}{\lambda_c}$. For inflow rates λ near the restriction rate λ_c, the congestion level ρ is near 1, which represents a highly congested server. Meanwhile, $\rho << 1$ implies that the region has a lot of resources (e.g., runways, airline spacing, human controllers) that are not utilized.

The congestion level at a TFM restriction is subject to change due to unexpected weather events, and due to re-organization of traffic flows/region capacities through planning. Specifically, in the face of severe weather, the restriction λ_c is decreased to, say, $\tilde{\lambda}_c < \lambda_c$ due to the reduced capacity, and hence the congestion level is increased by $\Delta\rho = \frac{\lambda}{\lambda_c} - \frac{\lambda}{\tilde{\lambda}_c}$. Right after the weather event, the capacity returns to normal, but the inflow rate may be increased to, say, $\widetilde{\lambda}$ due to the waiting delayed aircraft. Hence the congestion level is increased by $\Delta\rho = \frac{\widetilde{\lambda}-\lambda}{\lambda_c}$ from its nominal value. Similarly, congestion may increase simply because of increased traffic demand in a region. The congestion level can also be decreased through remedial strategies. For instance, re-planning aircraft routes helps to reduce the inflow rate to a boundary, say to $\widetilde{\lambda}$, and hence the congestion level is changed by $\Delta\rho = -\frac{\widetilde{\lambda}-\lambda}{\lambda_c}$. Similarly, investment in airport runway expansion, reallocation of human controllers, and improved TFM decision-making schemes can increase a region's capacity, say to $\widetilde{\lambda_c}$, so that congestion level is changed by $\Delta\rho = -\frac{\lambda}{\lambda_c} + \frac{\lambda}{\widetilde{\lambda_c}}$. The fact that all these different mechanisms change congestion levels indicates the importance of finding the sensitivity of backlogs/delays to congestion.

The backlog's sensitivity to congestion level can be obtained from Equation 4.2 by taking the derivative of $E(B)$ with respect to ρ. This yields a sensitivity $S_B(\rho)$ given by

$$S_B(\rho) = \frac{\rho(2-\rho)}{2(1-\rho)^2}. \tag{4.4}$$

Equation 4.4 clearly demonstrates the nonlinearity of the sensitivity of the backlog. As seen from Figure 4.2, the impact of a small disturbance on congestion level becomes more critical with the increase of the congestion level. This fact suggests two important viewpoints:

- TFM actions or airspace with high congestion levels are more sensitive to unexpected weather events. Much more backlog can be produced due to capacity variations in these sensitive regions than in other regions. These backlogs can propagate from these sensitive regions to the network and greatly worsen TFM performance NAS-wide.

- In terms of planning (e.g., route replanning, allocation of human controllers, facility improvement, and runway expansion), allocating resources to the regions that have high congestion levels will reduce the backlog more effectively. Hence, these sensitive regions that have higher congestion levels are more critical in planning.

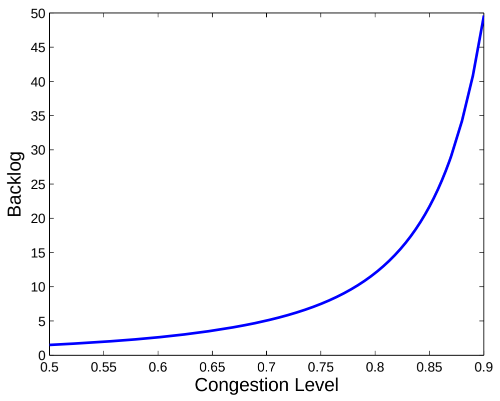


Fig. 4.2: Sensitivity of backlog with respect to congestion level

We note that a very similar analysis can be used to determine sensitivity of average queueing delays. Unlike the backlog, the queueing delay is not explicitly a function of the congestion ρ, and so we find it more convenient to separately compute the sensitivity to inflow and restriction rate changes. Specifically, the sensitivity to inflow rate changes is captured by $S_D(\lambda) = \frac{1}{2(\lambda_c - \lambda)^2}$, and the sensitivity to restriction rate changes is captured by $S_D(\lambda_c) = \frac{-\lambda(2\lambda_c - \lambda)}{2\lambda_c^2(\lambda_c - \lambda)^2}$. The sensitivity analysis of delays provides insight into weather impact and planning in the same way as the analysis of backlog.

4.3 Evidence for Congestion-Dependence of NAS Sensitivities

In this section, we give evidence supporting that the sensitivity of NAS performance to disturbances varies widely with the disturbance's location, and more particularly that queueing models predict the dependence of sensitivity on performance. This supporting evidence also clarifies how sensitivities can be identified/compared from historical data, and related to congestion measures. With the motivation that we are introducing a broad approach to planning, we pursue canonical examples for various disturbances (e.g., weather-based capacity changes and evolution of flow densities) and locations (terminal-area and en route). We note that the examples are pursued in varying levels of detail (in some cases, we give numerical verification of the queueing-theory predictions from historical data, while in other cases only citing relevant qualitative results), but each gives credence to the described sensitivity analysis.

Example 1: Terminal Area Delays due to Winter Weather

Severe weather—including convection, winter weather, stratus, and high winds—is the most significant cause of delays in the NAS [85]. Here, as an example, we study the impact of winter weather on departure delays. In particular, we characterize the sensitivity of delays at eight airports in the Northeast corridor to capacity reductions due to winter weather in December 2007 and January 2008, as well as in December 2006 and January 2007. We use the **Aviation System Performance Metrics** (ASPM) database to perform this comparison. In particular, from the ASPM data, we obtain historical **Airport Acceptance Rates** (AARs) and **Airport Departure Rates** (ADRs) as well as actual traffic demands, and hence capacity utilization or congestion level. We also obtain the average arrival and departure delays during Instrument Flight Conditions (IFC) periods, which specify the inclement-weather periods at these airports. At the simplest level, we wish to verify that the sensitivity to weather disturbances grows with the nominal congestion level (the average congestion level during the three-month period). To this end, we have tabulated the number of **excessive delay days**—i.e., days in which the average delay is more than twice the average delay for the whole period—as well as the average congestion at each airport (Table 4.1). As a second comparison, we have also tabulated the fraction of arriving airport delayed more than one hour at each airport, as a measure of high sensitivity. We see that the number of excessive delay days exhibits a strong dependence on the nominal congestion, with lightly congested airports

(Pittsburg, Providence) having only one or two excessive delay days and the busiest (New York's Laguardia and John F. Kennedy airports) having 10-12 excessive delay days. Similarly, the busiest airports have a much larger fraction of highly-delayed aircraft. This tendency for the busiest airports to have excessive delays verifies the higher sensitivity of queueing systems with higher congestion levels.

Tab. 4.1: We tabulate excessive delay days (days in which the delay is twice the average for that terminal) and congestion at Airports in the Northeast and Upper Midwest, during December 2007 and January 2008. We also list the fractions of aircraft arriving at each airport that were delayed more than one hour During December 2006 and January 2007. High congestion airports are more likely to have excessive delays.

Airport	**% Congestion**	**Excessive Delay Days**	**High-Delay Fraction**
PIT	13	2	0.048
PVD	16	1	—
IAD	35	4	0.053
BOS	42	12	0.039
BWI	44	7	0.028
DCA	45	6	0.035
MDW	46	7	0.042
PHL	55	12	0.072
EWR	56	8	0.11
ORD	58	10	0.084
LGA	58	12	0.081
JFK	61	12	0.080

We can potentially obtain a more refined characterization of the sensitivity to winter weather, by accounting for variations in weather severity at the airports. Of note, Boston's Logan airport (BOS) appears to have an unusally high number of excessive delay days; it is plausible that this excess is caused by a higher severity in the weather at BOS as compared to the other terminals. To check whether this is the case, we have compared the reduction in capacity (ADR/AAR) at BOS during IFC periods as compared to other terminals with similar nominal congestion (e.g., IAD). As a preliminary analysis, we have compared the variability in the capacity at BOS as compared to IAD, and find that the spread is indeed larger at BOS.

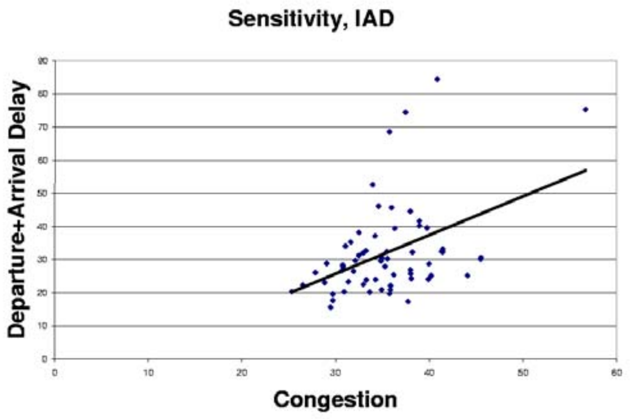
Finally, as a more detailed study, we have plotted daily average delays against congestion on that day (see Figure 4.3), for two airports (Washington' Dulles Airport, IAD, which has moderate congestion; and Providence Airport, PVD, which has low congestion). Linear regression of the delay with respect to the congestion is also performed[*]. We reach two conclusions from this detailed survey: first, it lends credence to the argument that a queueing mechanism underlies the delay sensitivity of the airports, and 2) it permits detailed comparison of each airport's sensitivity. We indeed observe much higher sensitivity at IAD.

Example 2: Effect of Disturbances on En Route Delays

Weather events also engender en route delays, by forcing re-routing of aircraft along less optimal routes, restriction of flows using AFPs and MIT/MINIT restrictions, and ground-based flow management. As with terminal-area delays, we would expect en route delays to be more sensitive to weather in highly congested areas.

The variable sensitivity of NAS-wide total delay (i.e., the total of both airborne and terminal

[*] We have checked that the regressions meet the criterion of the F-test, to ensure that sufficient data is present.



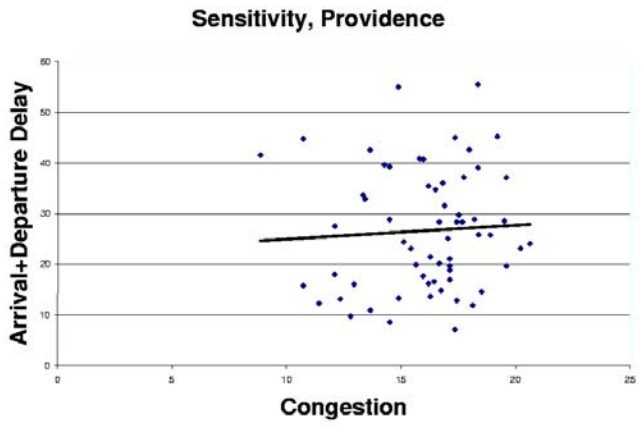


Fig. 4.3: The dependence of the average aircraft delay on the daily congestion level is shown for two airports. We notice that the average delay exhibits a weak dependence on the congestion level (the regression line is $E(D) = 0.276\rho + 22.2$) for the low congestion airport, PVD. Meanwhile, there is a stronger dependence ($E(D) = 1.168\rho - 9.334$) for the moderate-congestion airport, IAD.

area delay, for all flight legs during a period), is borne out by the numerical delay-prediction tool of Sridhar and coworkers [109, 110]. In particular, this empirical tool regresses total delay in terms of a weather-coverage- and traffic-density- based measure known as the **weather-impacted traffic index** (WITI). What is important to us here is that WITI scores for certain critical regions (in particular, the Northeast and Upper Midwest) contribute disproportionately to the total delay in the regression. On the highest delay days, the WITI for these critical regions are the ones that are exaggerated. In other words, the regression coefficients for these terms are larger than for other WITI regressors, suggesting that the sensitivity of the NAS performance (in terms of per-aircraft delays) to disturbances in these regions exceeds that of other regions. Noting that the critical regions are the ones with highest congestion, the study of regional WITI matches with the prediction of increased sensitivity obtained through queueing models. In this sense, our study can be viewed as giving a causation for the dependency of total delay on WITI and regional WITI scores.

To give further evidence for this higher sensitivity, we compare the en route average aircraft delays for two city pairs over three months (Nov. 2007 through Jan. 2008) using ASPM data. In particular, we compare en route delays for DFW-to-IAD flights, which do not pass through the highly congested Northeast corridor, and for DFW-to-BOS flights, which do pass through the Northeast corridor. We find that the standard deviation in an aircraft's en route delay is larger by roughly a factor of 1.5 for DFW-to-BOS flights even though the mean delays are similar, indicating the higher sensitivity to disturbances of the flights passing through the congested area.

Example 3: Sensitivity of Delays to Increased Traffic

The higher sensitivity of NAS performance measures at high-congestion locations is also reflected in the dependence of delays on long-term changes in traffic demands. Here, we study the dependence of average aircraft delay on traffic demand for nine airports with varying congestion levels, over a span of 15 years. In particular, we have studied aircraft arrivals into 9 terminals of varying congestion levels. For each terminal, we have regressed the annual average delay for arriving aircraft with respect to the total arrival traffic. As expected, the five congested terminals (JFK, LAX, ORD, PHL, and SFO) have a strong dependence of delay on traffic demand, while the remaining terminals (MSP, PHX, SEA, and SLC) have much weaker dependence, see Table 4.2.

4.4 Using the Sensitivity Analysis for Planning

In Sections 4.1, 4.2, and 4.3, we have verified that NAS performance is acutely sensitive to some disturbances and much less so to others. Fundamentally, we expect that knowledge of sensitivities may help us in planning and evaluation of new TFM strategies, in that high sensitivities are indicative of large delays and also are of concern themselves. In this section, we give a preliminary study on the use of sensitivity information in planning air traffic flow management strategies. Specifically, we first show that the optimal flow management schemes for banks/networks of queues are ones that *equalize* sensitivities to local disturbances in a certain scaled sense. We then use this insight to inform planning of various traffic flow management strategies, including reconfiguration and controller-redistribution ones. We find it most convenient to develop these design results in two steps: first, for banks of non-interacting queues (which may for instance represent multiple airports, or en route congestion points with largely uncorrelated flows); and second, for a network of queues with routing among them.

Tab. 4.2: For nine airports, we regress the average delay incurred on the aircraft entering the airport in terms of the percent change in annual traffic demand. The slopes of the regression lines are shown in the table. We note that the five highly congested airports (LAX,SFO,ORD,PHL,JFK) have strong dependences while the less congested ones (PHX, MSP, SEA, SLC) have much weaker dependences.

Airport	**Regression Slope**
JFK	0.88
LAX	1.00
MSP	0.17
ORD	2.26
PHL	1.70
PHX	-0.59
SEA	0.38
SFO	1.19
SLC	0.13

A couple further notes about the ensuing development are needed. First, we note that our analysis is focused on optimizing total backlogs, however a very similar analysis can be used to minimize delays. Second, we stress that our methodology is not focused on providing precise numerical results on optimal strategies, but instead on informing various planning tasks through use of sensitivity information.

4.4.1 Planning for a Bank of Queues

Here we consider a bank of n queues, i.e. a set of n queues that are operating on their own, or in other words do not have traffic flowing between them (see Figure 4.4). We assume an inflow rate to queue i equal to λ_i; when we are designing inflows, we shall assume that these flows originate from a single major flow of rate λ. Each queue i's restriction strength is denoted as λ_{c_i}. We define the congestion level related to queue i as $\rho_i = \frac{\lambda_i}{\lambda_{c_i}}$, hence the backlog of queue i is $E(B_i) = \frac{\rho_i^2}{2(1-\rho_i)}$, and sensitivity is $S_B(\rho_i) = \frac{\rho_i(2-\rho_i)}{2(1-\rho_i)^2}$ according to the development in the previous section. We give the designs that minimize the total backlog using two planning schemes: 1) reconfiguration of the flows; and 2) human-controller redistribution.

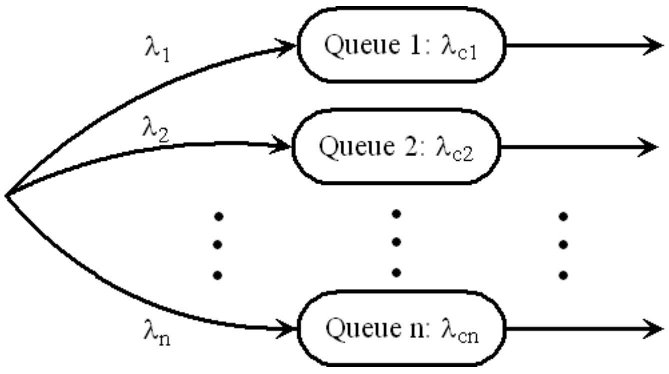


Fig. 4.4: Bank of queues

Reconfiguration and Route Re-Planning

Reconfiguration—or redrawing of region boundaries to ameliorate human-controller workload concerns and resulting congestion—is an area of intense current research [112, 118–120]. Although reconfiguration has been widely studied, however, a key difficulty lies in choosing measures to optimize. From a planning standpoint, our approach to reconfiguration may be valuable for reducing delays and sensitivity to adverse weather. In similar fashion, re-planning of routes may mitigate congestion and delay. From our perspective, both problems resolve to that of reconfiguring the inflows to reduce backlogs. Specifically, we consider the following optimization problem:

Design Problem 1 Consider a bank of n queues, as shown in Figure 4.4. Each queue i has a fixed restriction strength λ_{c_i}. We assign the inflow rate λ_i for each queue i so that the total backlog of the queues $\sum_{i=1}^{n} E(B_i)$ is minimized, subject to the following constraints:

- $\sum_{i=1}^{n} \lambda_i = \lambda$, where the total inflow rate λ is positive, and less than $\sum_{i=1}^{n} \lambda_{c_i}$.
- $\lambda_i \geq 0$.

We refer to the optimal inflow rate in queue i as λ_i^*, and the corresponding congestion as ρ_i^*.

We show in Theorem 4.1 that the sensitivities of the queues' backlogs have a simple relationship at the optimum:

Theorem 4.1. *Consider Design Problem 1. The optimal inflow rates λ_i^* satisfy the following condition: there exists a constant C such that $\frac{S_B(\rho_i^*)}{\lambda_{c_i}} = C$ for all i.*

Proof: This result follows directly from constrained optimization using Lagrange multipliers [71]. Specifically, the Lagrangian associated with the objective function and constraints is $L = \sum_{i=1}^{n} E(B_i) + C(\lambda - \sum_{i=1}^{n} \lambda_i) + u_i\lambda_i$, where the constants C and u_i are nonnegative. Taking derivatives of the Lagrangian with respect to C, λ_i, and u_i for all i, we obtain:

$$\frac{S_B(\rho_i^*)}{\lambda_{c_i}} - C + u_i = 0 \quad \forall i \tag{4.5}$$

$$\sum_{i=1}^{n} \lambda_i^* = \lambda$$

$$u_i\lambda_i^* = 0 \quad \forall i$$

Assuming $u_i = 0$ and solving the first two equations in Equation 4.5, we obtain $\lambda_i^* > 0$ for all i. From convexity, this solution is the global optimum. Thus, we see that $\frac{S_B(\rho_i^*)}{\lambda_{c_i}} = C$ for all i. □

This theorem shows that for the optimal flow allocation, the ratio between sensitivity and restriction strength is common among all queues. The proof of the theorem also gives the design of the optimal flow allocation: by rearranging the conditions given in the theorem together with the first constraint in the problem formulation, we can obtain the optimal flow allocation. The details of the algebra are unimportant for our purposes here.

The optimal design presented in Theorem 4.1 informs planning of reconfiguration and route-selection strategies in several ways:

1) The design is based on useful measures of performance (i.e., small backlog or delay), and hence permits design and evaluation with these measures in mind. Of particular interest, for a particular plan, we can ckeck the sensitivity to restriction-strength (capacity) ratio for each congestion point, and so decide whether the performance is near-optimal. Such an approach would be helpful e.g. in evaluating the configuration design in [112], in the case where capacities vary throughout the airspace.

2) The design result suggest a data-driven methodology for iteratively improving reconfiguration strategies. In particular, from historical data, we can obtain sensitivities of backlogs/delays on various routes, as well as the capacities of the congested points along the routes. Our design shows that inflows should be reduced through route selection or reconfiguration at locations where the sensitivity-to-capacity ratio is high.

Controller Redistribution

Assuming that the flow rates are fixed, we can also redistribute control resources to minimize the total backlog. For instance, for en route flow restriction, human controllers can be re-assigned to mitigate capacity concerns. The problem can be formulated as follows:

Design Problem 2 Consider a bank of n queues, as shown in Figure 4.4. Each queue i has an approaching Poisson flow with fixed rate λ_i. We assign restriction strength λ_{c_i} to queue i for each i, so that the total backlog of the queues $\sum_{i=1}^{n} E(B_i)$ is minimized, subject to the following constraints:

- $\sum_{i=1}^{n} \lambda_{c_i} = \lambda_c$ (i.e., the total capacity resource is fixed). Here, the constant λ_c is greater than $\sum_i \lambda_i$;
- $\lambda_{c_i} \geq \lambda_i$.

We denote the optimal restriction strengths (capacities) by $\lambda_{c_i}^*$, and the corresponding congestion by ρ_i^*. Theorem 4.2 gives the structural condition of the optimal controller allocation.

Theorem 4.2. *Consider Design Problem 2. The optimal restriction strengths $\lambda_{c_i}^*$ satisfy the following condition: there exists a constant C such that $\frac{S_B(\rho_i^*)\rho_i^*}{\lambda_{c_i^*}} = C$ for all i.*

Proof: The proof is very similar to the proof of Theorem 4.1 and hence is omitted.

Similarly to the flow reconfiguration, for the optimal controller redistribution, the backlog sensitivities of each queue are equal in a scaled sense. To obtain the optimal controller allocation, we can solve the condition given in Theorem 4.2 together with the first constraint given in the problem formulation.

We note that the re-distribution result also shows how new controller resources should be assigned, in particular to reduce $\frac{S_B(\rho_i)\rho_i}{\lambda_{c_i}}$ where this measure is large. This observation may be especially helpful for planning improvement at airports, in that airports with critical need for improvement can be identified.

4.4.2 Planning for an Interacting Network of Queues

Finally, we study design of inflow rates for an acyclic *network* of queues, with the motivation of gaining insight into route-planning in a more general way. In particular, we characterize the minimum backlog design in terms of backlog sensitivities along paths in the network. We discuss application this analysis to re-allocation of routes in the NAS, either for the purpose of enacting long-term improvements in performance or for planning re-routes for common adverse-weather or high-traffic conditions. We again stress that we do not seek to capture all the details involved in route planning, but rather give a rubrik for what high-performance routing strategies are, for the purpose of planning.

The Network Model

We consider a network of queues that represent flows along multiple routes between one traffic source and one destination (Figure 4.5). Specifically, we consider a network of n restrictions or queues, labeled $1, \dots, n$, with traffic of total flow rate λ approaching Queue 1, and leaving the network from Queue n. We assume that traffic flow is permitted along the edges in a directed acyclic graph, i.e. that there are a set of ordered Queue pairs $\{i, j\}$ (where WLOG $i < j$) such that traffic flow is permitted from Queue i to Queue j. We refer to these Queue pairs as **flow edges** in the network, and refer to the set of such edges E as the flow edge set. Without loss of generality, we assume that there is a flow path (a path of flow edges) from Queue 1 to each other queue, and similarly that there is a flow path from each queue to Queue n. We find it convenient to diagram the queueing network, see Figure 4.5. We note that an arrow is drawn from Queue i to Queue j in the diagram if and only if flow is permitted between the queues.

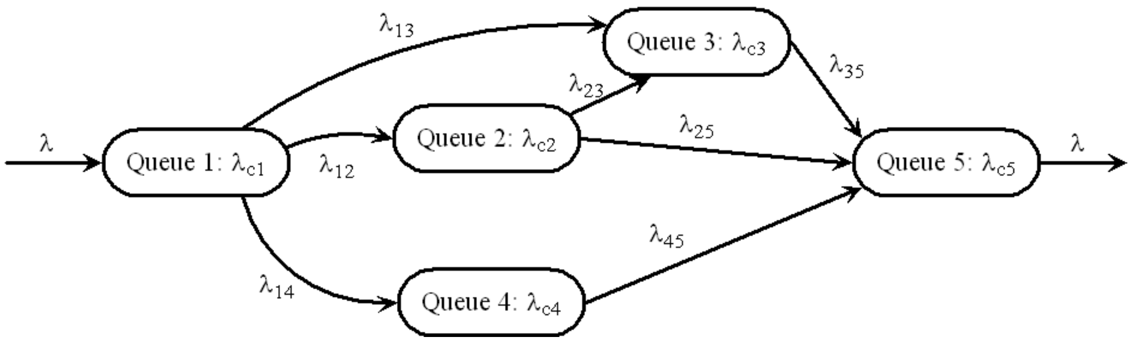


Fig. 4.5: Network of queues

We assume that Queue i has a strength or capacity λ_{Ci} for the traffic between this source and destination. We assume that, for any set of queues whose removal separates the network into multiple pieces, the total capacity is at least λ: this requirement is necessary to permit the entire flow to traverse the network without backlog growing in time.

Meanwhile, we assume that the traffic flow from Queue i to Queue j is a Poisson process of rate λ_{ij}. If there is a flow edge between Queue i and Queue j, then the flow rate λ_{ij} is a nonnegative constant. If there is not a flow edge between the queues, then $\lambda_{ij} = 0$.

We enforce that total flow into each queue is equal to the total flow out of the queue, i.e.

$$\begin{aligned}
&\sum_{j\neq 1} \lambda_{1j} = \lambda \\
&\sum_{j\neq i} \lambda_{ij} = \sum_{j\neq i} \lambda_{ji}, \quad i = 2, \ldots, n-1 \\
&\sum_{j\neq n} \lambda_{jn} = \lambda
\end{aligned}$$

The reader will note that we have made the simplifying assumption that the flow into each Queue i is Poisson; this assumption is often appropriate in air traffic management applications given that mixing of flows occurs between the bottleneck queues, see [82, 86] for details. We refer the reader to [2] for more accurate queueing network models, which explicitly capture the effect of smoothing at one restriction on delays/backlogs at the next. We are currently pursuing routing design for one such queueing network model, namely a network of M/M/1 queues. However, these models are not critically needed for the planning tasks pursued here, so we defer a treatment to future work.

We refer to the expected backlog at Queue i by $E(B_i)$, and note that the expected backlog is given by $S_B(\rho_i) = \frac{\rho_i(2-\rho_i)}{2(1-\rho_i)^2}$, where now $\rho_i = \frac{\sum_{j \neq i} \lambda_{ij}}{\lambda_{Ci}}$. We notice that the sensitivities of $E(B_i)$ to each capacity and flow rate can be computed, as in the proofs of Theorems 4.1 and 4.2.

Holistically, we refer to the model as the **traffic flow queueing network**.

Flow Rate Design and its Application to Route Selection

Several design problems may be of interest for the traffic flow queueing network. Specifically, as with the design for banks of queues, both capacity selection and flow rate selection can be pursued. However, noting that the inflows to each queue are assumed to be Poisson regardless of the dynamics of upstream queues, we immediately see that the capacity design problem resolves to corresponding problem for a bank of queues, and so no further development is needed. Thus, we focus here on the problem of designing flow rates between queues, to reduce backlog.

Specifically, the design problem that we address is to select the flow rates λ_{ij} for $\{i, j\}$ in the flow edge set E, so as to minimize the total expected backlog $\sum_{i=1}^{n} E(B_i)$. Here, we note that the nonnegative rates λ_{ij} are constrained to satisfy $\sum_{i=1}^{n} \lambda_{ij} \leq \lambda_{Ci}$. We refer to this task as the *flow-rate design problem*. We use the notation λ_{ij}^* for the optimal flow rates, and refer to this design as the **optimal flow assignment**. We use the notation ρ_i^* for the inflow to Queue i for the optimal flow assignment.

As with the design for banks of queues, it turns out that we can learn much about the optimal flow assignment by considering the sensitivities of the backlogs to the flow rates. In particular, we find that the backlog sensitivities satisfy a set of insightful conditions, as detailed in the below theorem. Before presenting the theorem, we find it convenient to define a sensitivity notion for a path. In particular, consider a path $\{j_1, j_2, \ldots, j_q\}$ from Queue $j_1 = i$ to Queue $j_q = n$. We define the **total backlog sensitivity** (TBS) for the path as follows:

$$TBS = \sum_{r=2}^{q} S_B(\rho_{j_r}) \frac{1}{\lambda_{C,j_r}}. \tag{4.6}$$

That is, the TBS can be computed by finding the sensitivity of the backlog to the congestion along each path edge, scaling each sensitivity by the queue capacity, and summing over the edges. Conceptually, the TBS is an aggregate measure for the sensitivity of the backlog to changes in flows along the path.

We are now ready to present the theorem on sensitivities for the optimal solution:

Theorem 4.3. *A flow assignment is optimal if and only if the total backlog sensitivity (TBS) along all paths from Queue* 1 *to Queue* n *are the same.*

In words, a flow assignment is optimal only if the TBSs along all paths from each queue to the destination are equalized; conceptually, such an assignment achieves an extremum since the backlog is insensitive any differential rearrangement of flow rates (subject to the flow conservation constraints). The fact that the optimal is in fact a minimum follows naturally from the convexity of the cost. Since the analysis is quite similar to that for a bank of queues, we omit the detailed proof. Again, we note that the optimal allocation can be obtained straightforwardly from the sensitivity conditions.

The above result on the structure of optimal flow allocations is instructive for planning of routes, either for overall improvement of NAS performance or for particular common weather scenarios. In either case, the optimization result shows the following: when multiple routes from a source to a destination are available, a good route selection is one for which the total backlog sensitivity (TBS) along each path is similar. This observation can be used for route planning, as follows: from historical data, estimates of TBSs can be obtained; in turn, the sensitivities can be used to obtain improved route selections. This approach may be useful, for instance, in splitting flows among multiple routes in high-congestion or inclement-weather scenarios, see [116] for background on probabilistic planning in these circumstances.

Let us conclude our development by pointing out a couple connections and future directions of the routing-design study:

1) The result presented here is closely connected with our ongoing efforts to design controllers and/or graphs to shape an associated dynamics (e.g., [1, 2, 5]). These various efforts have the common theme that we identify the *structure* of well-designed graphs or networks, and hence compute designs that achieve high performance. Our other studies have focused on deterministic linear network dynamics; this effort is a step toward applying such structural design strategies to queueing networks.

2) Two enrichments of the presented design strategy are especially important. First, our routing design does not yet account for nominal differences in cost (e.g., delay, fuel cost) among the various options. Such differences are often present, and so making the tradeoff between nominal-cost differences and queueing costs is important. Second, our design does not explicitly try to reduce backlog sensitivity but rather only the backlog itself (though the resulting optimum is related to the sensitivities). In situations where disturbances are common, reduc-

ing the sensitivies themselves may be important.

5. A NETWORK MODEL FOR ACTIVITY-DEPENDENT SLEEP REGULATION

We develop and characterize a dynamical network model for activity-dependent sleep regulation. Specifically, in accordance with the activity-dependent theory for sleep, we view organism sleep as emerging from the local sleep states of functional units known as cortical columns; these local sleep states evolve through integration of local activity inputs, loose couplings with neighboring cortical columns, and global regulation (e.g. by the circadian clock). We model these cortical columns as coupled or networked activity-integrators that transition between sleep and waking states based on thresholds on the total activity. The model dynamics for three canonical experiments (which we have studied both through simulation and system-theoretic analysis) match with experimentally-observed characteristics of the cortical-column network. Most notably, assuming connectedness of the network graph, our model predicts the recovery of the columns to a synchronized state upon temporary overstimulation of a single column and/or randomization of the initial sleep and activity-integration states. In analogy with other models for networked oscillators, our model also predicts the possibility for such phenomena as mode-locking.

5.1 Introduction

Sleep is a fundamental process in human and animal life, that comprehensively impacts both our day-to-day existence and our long-term growth and development. The fundamental importance of sleep has fostered extensive study on its neurological characteristics and mechanisms (e.g., [121, 122]). This research has been complemented by efforts to mathematically model the sleep-wake cycle as a homeostatic (regulation) process, with the aim of giving predictive descriptions of sleep dynamics (e.g., [123–125]). In the *activity-dependent* or use-dependent theory for sleep [122, 126], the fundamental units that transition between sleep and wake states (as reflected by functional changes in these units) are groups of tightly-connected neurons in the cortex known as *cortical columns*. The biochemical and bioelectrical mechanisms underlying the sleep/wake transition in each cortical column are modulated by *local* activity, i.e. by electrical and biochemical drives that project a particular function (e.g., the twitching of a particular whisker on a rat) onto an individual cortical column. The transition is also modulated by loose network couplings among the columns, and by sleep regulatory circuitry. Our aim here is to develop a mathematical model for this network of cortical columns, that captures the fundamentals of the activity-dependent mechanism of sleep.

It is worthwhile to connect our modeling efforts with the existing models concerned with sleep regulation. Sleep has been extensively modeled at the behavioral level (e.g., [123]). These simple models capture 1) the projection of the circadian rhythm into sleep dynamics and 2) some homeostatic regulation of the sleep state, at a whole-organism level. However these models do not capture either the spatial structure or the biochemical/bioelectrical pathways underlying activity-dependent sleep. Cortical columns (and more generally neuronal assemblies) are well-known to be basic building blocks for sleep and memory, and there has been some interest in modeling their dynamics. In particular, a variation of the classical Wilson-Cowan model has been shown to display the periodic responses characteristic of excitatory/inhibitory processes in cortical columns [127]. Recently, a more complicated model for cortical column dynamics has been developed, that explicitly codes the notion of a sleep state as well as the activity-dependent evolution of assembly dynamics [128]. While these models represent the regulatory role played by cortical columns, they cannot capture the translation of local activity (activity at one or a small number columns) into a global sleep state. Also of interest, network models (often comprising excitatory/inhibitory agents as components) have been used to capture a variety of neuronal dynamics in e.g. the thalamus and subthalamic nucleus including in sleep regulatory circuitry (e.g., [129–132]). In this direction, Hill and Tononi have developed a circuit-level model of the thalamocortical system, that permits examination (through simulation) of the nominal sleep-wake transition and the slow-wave sleep dynamics [133]. Massimini and coworkers have put forth and tested the hypothesis that the network connectivity changes between waking, REM, and NREM sleep, respectively; their effort can be viewed as an explicit modeling of the network topology's evolution, and hence complements our efforts to model overlayed dynamics [134]. The evolution of synaptic weights (connectivities) is also explored using a mean-field analysis of a network model, in [135]. Meanwhile, cortical activity has also been represented using continuum models rather than networks with discrete elements, see e.g. [136]. On the other hand, activity-dependence has also recently been considered in the sleep modeling literature [137], though only at the level of detail of a two-process model. The current work advances the existing modeling efforts by making explicit the role played by the cortex in sleep and its regulation, in particular by 1) making explicit the incorporation of local activity in sleep regulation through the cortical columns and 2) capturing networked interactions among the columns.

The biochemical and electrical mechanisms of sleep include the processes by which local activity, coupling of neighboring cortical columns, and regulatory circuits modulate **sleep regulatory substances** (SRSs), as well as the mechanisms by which accumulated SRSs cause the functional changes associated with sleep-state change. Here, we abstractly view each column's intricate dynamics as an activity-integrator that modulates a functional sleep state; using this abstraction, we represent the cortical columns as a network of activity-integrators with associated functional states, that further interact through loose coupling and through regulatory circuitry. With this coarser or *network-level* model, we are able to study how the local dynamics of the columns can foster formation of a global sleep state.

Specifically, our modeling efforts contribute to ongoing research in the following ways:

1) From the perspective of sleep research, our network model captures analytically the combined roles of local activity inputs, coupling between cortical columns, and regulatory circuitry in formation and evolution of a global sleep state. As such, it is depictive of the activity-based theory for sleep developed by Krueger and co-workers [122,126], and permits exploration of the sleep-state dynamics under the premises of the theory. While our primary aim here is to give a plausible description and analysis of activity-dependent sleep, the model holds promise in the long run as a tool for prediction and design, for instance in characterizing the effects of sleep deprivation and/or designing drugs that impact regulation.

2) In that we study regulatory dynamics defined on a graph, our work also explicitly connects sleep modeling with the ongoing effort to model and in turn control dynamical networks, e.g. [1, 25, 140]. We note here that the activity-dependent theory for sleep regulation matches the developing paradigm for control in modern engineered networks, wherein highly limited agents interact through *localized* network couplings (with possibly rather complex or arbitrary coupling topologies) to achieve a global regulation task (see, e.g., the overviews [141,142] or the articles [68, 143]). Within this broad domain, our efforts are especially connected with ongoing research in both the physics and electrical engineering communities on synchronization or *agreement* in oscillator networks (see e.g [144–147] in the physics and natural-sciences literature, and [140, 148–153] in the engineering community). From this perspective, the key contributions of the work are the

1) introduction and characterization a general network model for synchronization with external inputs, and 2) the study of a novel nonlinear dynamic model for oscillators.

The chapter is organized as follows. In Section 5.2, we motivate and formulate the network model. Section 5.3 characterizes the model and illustrates its predictive capability, through both simulation and analysis.

5.2 Network Model Formulation

In this section, we propose a network model for the interaction of cortical columns, which shows promise in predicting activity-dependent regulation of sleep. Specifically, we represent individual cortical columns as very simple but interacting activity-based regulators, and explore the role played by the network interactions in translating local activity inputs as well as global regulatory-circuit signals into whole-animal sleep. Our model captures both the spatial structure and temporal characteristics of sleep identified in the activity-dependent theory [122, 126].

In the model that we propose, the interactions among the cortical columns are critical to the rapid formation of a global sleep state. This paradigm of local interactions leading to a global state has been of considerable interest to the complex-systems modeling community (e.g., [140, 157]) as well as the network-control community (e.g., [1, 25, 141]). A key feature of the networks considered in this literature is that they are built of agents with very simple internal dynamics, but quite complicated interactions that lead to interesting global dynamics. We note that the model developed here is of the same form, and hence indicates a new application for this complex networks theory. Also of interest, the model described here can be viewed as having an intrinsic mechanism for the emergence of a global state, but complementarily also can be viewed as using both external inputs and feedback through network coupling to achieve regulation. In this sense, this study connects the modeling paradigm pursued in the complex-system community with the regulator-design paradigm of the network-control community.

The following are the key points of the activity-dependent theory for sleep used in model development [122,126]. During awake periods, individual cortical columns integrate (store) activity information (or energy for activity relative to available energy) through biochemical and electrical means, in entities known as sleep regulatory substances; when this integrated activity becomes large enough, the cortical column transitions to a sleep state (a state exhibiting unresponsiveness to sensory stimuli, certain increases in synaptic plasticity, etc). It is postulated that the transition to a sleep state is also impacted by spatially-close cortical columns that are already in a sleep state. These columns tend to drag the awake column toward the sleep state through biochemical and electrical means, and hence foster the formation of a global sleep state. Similarly, a cortical column in the sleep state can be viewed as containing processes that gradually return to a waking state (either through inhibition of the processes inducing sleep, or through other integrative processes); again, nearby columns that are in an awake state have an impact. Besides the activity-dependent dynamics and couplings, sub-cortical global regulatory circuits impose a circadian rhythm and also permits rapid waking under stimulus.

Based on the above description, we abstractly model each cortical column using a *sleep state variable* and an *activity variable*, which evolve in time due to integration of local activity, as well as interactions with other columns and global regulation. Precisely, let us consider a network of n cortical columns. Each cortical column i is described by a binary sleep state variable $S_i(t)$, where $S_i(t) = 1$ indicates that the column is in the sleep state, while $S_i(t) = 0$ indicates a wake state. We also associate a continuous-valued **total activity variable** (or simply activity variable in short) $x_i(t)$ with the cortical column i, which indicates the total activity since waking when the column is in the wake state, and indicates the total restoration effort in the sleep state.

In addition to the internal variables for each column, the model comprises a **network topology** describing the *strengths* of interactions between cortical columns. In particular, for each pair of cortical columns i and j, we use a fixed nonnegative weight w_{ij} to capture the strength of the effect of cortical column i on cortical column j. We find it convenient to assemble the weights into a (possibly asymmetric) topology matrix $W \stackrel{\triangle}{=} [w_{ij}]$. Also, we draw a network graph comprising n vertices labeled $1, \ldots, n$, with an edge drawn from i to j if and only if $w_{ij} > 0$. One can postulate several plausible formative odels for the network topology; however, the fundamental observed behaviors of the model are not dependent on the specifics of the network topology beyond its connectedness, so we leave the representation general.

We also assume the existence of a global clock signal, which eventually enforces (under normal activity conditions) that the cortical columns not only synchronize but transition between the sleep state and wake state at environmentally-appropriate times, i.e. according to a circadian rhythm. In humans, the clock signal is maintained by the suprachiasmatic nuclei (SCN), and distributed globally through neuronal connections, see e.g. [125] for details and modeling methods. Here, we denote the scalar clock signal by $C(t)$, where $C(t) = 1$ indicates that the organism should be awake, $C(t) = -1$ indicates that sleep is desirable, and $C(t)$ between -1 and 1 indicates weaker proclivities for waking/sleep.

Now we are ready to describe the evolution of the sleep state variables and the activity variables. Broadly, the activity variable gradually increases for each column during awake periods (depending on activity at the column), and gradually decreases during asleep periods. The sleep state variable changes when the activity relative to provided energy reaches thresholds; this threshold conceptually represents either an energy-deficit level, or a biochemical state, such that sleep occurs. Specifically, let us first consider the evolution of a cortical column i that is currently in the awake state ($S_i(t) = 0$). We model the activity variable $x_i(t)$ and sleep state variable $S_i(t)$ as evolving as follows:

- $\dot{x}_i(t) = +u_i(t) + \sum_{j=1}^{n} S_j(t) step(x_j(t) - E_j) w_{ji} + \alpha_i(1 - C(t))$
- $S_i(t) \to 1$ if $x_i(t) - E_i > T_i$,

where $u_i(t)$ is the activity input at the cortical column i at time T, $E_i(t)$ is the energy available during the awake period, T_i is the sleep threshold, and *step*() is a function that equals 1 for positive arguments and 0 for negative arguments. We notice that, nominally, the activity variable $x_i(t)$ integrates the activity at the column over time, but the cortical column is more quickly driven toward the sleep state when connected cortical columns have recently entered the sleep state ($S_j(t) step(x_j(t) - E_j) > 0$). The strength of this interactive response scales with the weight w_{ji}.

Similarly, the activity variable and sleep state variable evolve during the asleep period, as follows:

- $\dot{x}_i(t) = -r_i(t) + \sum_{j=1}^{n}(1 - S_j(t)) step(E_j - x_j(t)) w_{ji} + \alpha_i C(t)$
- $S_i(t) \to 0$ if $E_i - x_i(t) > T_i$,

where $r_i(t)$ is called the recovery input to cortical column i, and represents restoration of the activity variable prior to waking (which is connected to the repair and synaptic development occurring during sleep). Again, we note that the activity variable integrates both local input and signals from nearby cortical columns that have recently entered the awake state.

We holistically refer to the model as the activity integrator-network (AIN). Let us reiterate the connection of the AIN with the extensive literature on network control. Over the several years, there has been extensive research concerned with analyzing dynamics defined on a graph, and relating characteristics of such dynamics with structural characteristics of the underlying graph, see [73, 142, 158] for overviews of some important aspects of this analysis. Recently, control theorists have realized that understanding network structure further is critical to *controlling/designing dynamics* on a network, in such diverse fields as autonomous vehicle team formation, sensor networking, and virus-spreading control [1, 25, 68, 141]. What these various works have in common is that individual agents with very simple internal dynamics achieve a global task through network interactions. The AIN falls within this paradigm, in that cortical columns with essentially integration and thresholding capabilities achieve global sleep regulation. Within this broad class, the AIN is most closely connected to models for oscillator networks and rotational agreement, though the specifics of the dynamics differ from the models in the literature, e.g. [148–151]. One very significant novelty in our development, from a modeling and control-theory standpoint, is that we consider the impact of external input signals (disturbances) on the dynamics.

Let us conclude our formulation of the AIN by noting two limitations of the model. First, we have entirely excluded modeling of the humoral mechanisms for sleep, see e.g. [160], because our efforts are focused on the local couplings in the cortex that underly sleep. Second, we have not attempted yet to model all the time- and state-dependent variations of the network structure/parameters that are observed in the sleep cycle (e.g. [126]). Most prominently, the coupling parameters between the cortical columns would be expected to change between the sleep and wake states, and also during sub-intervals of sleep and waking (e.g., REM sleep, high-activity waking periods).

5.3 Prediction of Whole-Animal Sleep: Simulations and Analysis

We illustrate that the AIN dynamics match the predictions of the activity-dependent sleep theory through simulations (Section 5.3.1) and system-theoretic analysis (Section 5.3.2). We note

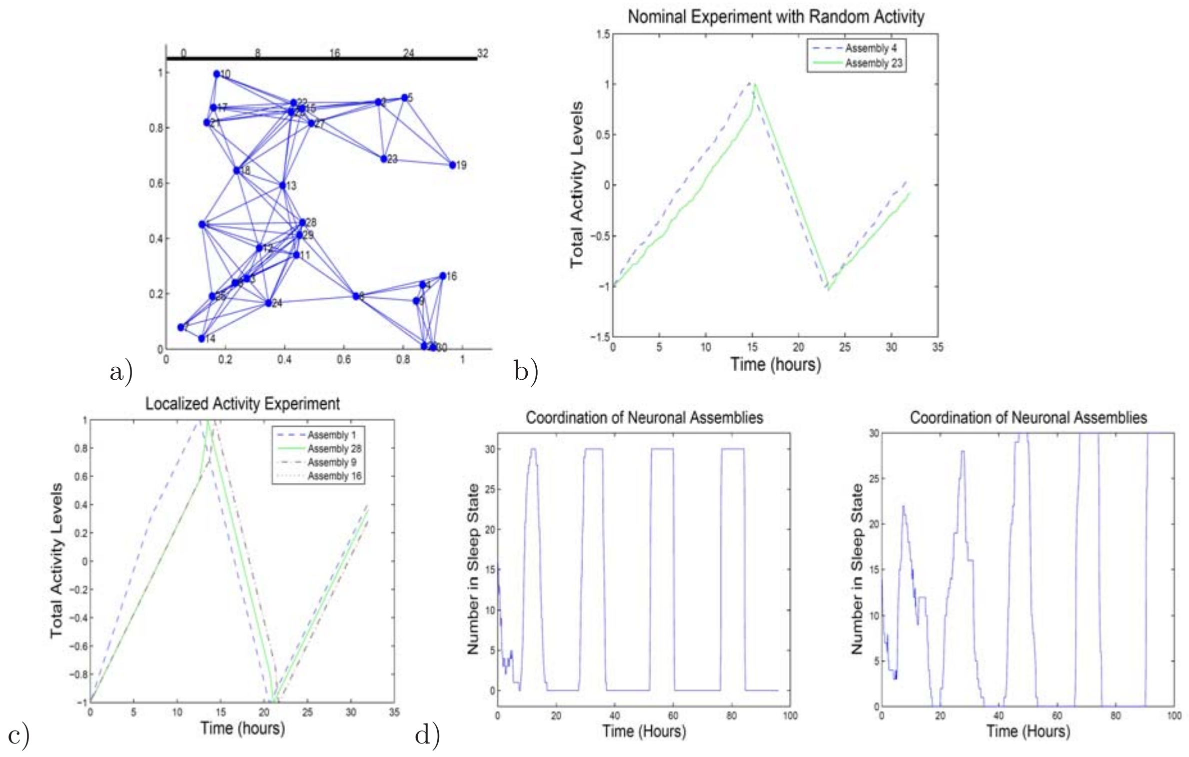


Fig. 5.1: a) The network topology for the 30-cortical-column example is illustrated. b) The baseline activity simulation is shown. c) The local overstimulation experiment is simulated; activity variables for five representative cortical columns (neuronal assemblies) are shown. d) The coordination experiment is simulated, for two different interaction strengths. Higher interaction strengths yield faster coordination.

that our efforts characterize both the internal dynamics and the input-to-state behavior of the AIN.

5.3.1 Illustrative Simulations

We illustrate the combined role of the activity inputs and network interactions in the AIN dynamics, using several canonical simulations. We present simulation results using a network with 30 cortical columns with identical internal dynamics, see Figure 5.1.

Baseline Activity Simulation Under normal rest or light activity conditions, a reasonable assumption is that the cortical columns are initially synchronized, and the activity inputs at each column are independent stochastic signals with identical statistics. In Figure 5.1, we show time-traces of two columns' total activity variables in the example AIN, under baseline activity conditions. The simulation illustrates that the loose couplings between columns are needed for maintaining coordination: because of the loose coupling, we see that the transition to the sleep state flows in a wave-like fashion through the network.

Local Overstimulation Experiment Experiments in which one or a small number of cortical columns are overstimulated have been of particular interest in the sleep community, because they permit evaluation of the claim that sleep is activity dependent. For instance, the impact on a rat's sleep response of repeatedly moving a single whisker has been studied [159]. We simulate such an experiment, by driving one or a small number of cortical columns with an input that is significantly larger than the nominal. In particular, we overstimulate one cortical column for a period, causing it to quickly enter the sleep state. Once the column has entered the sleep state, nearby columns begin to rapidly transition toward a sleep state, with the rate of transition become more pronounced as more columns enter the sleep state, see Figure 5.1. Thus, the sleep state spreads rapidly throughout the network, before the nominal falling-asleep time. We also note that the columns become even further coordinated during the subsequent transition from sleep to waking.

Coordination Experiment It has been postulated that cortical columns with initially uncorrelated sleep states eventually achieve coordination, because of the interactions between the columns. To capture this instance in our model, we initialize each cortical column with a random total activity variable value and a random sleep state, and observe the responses of the columns over several days. Our simulations indicate that, indeed, the cortical columns become coordinated over time, with the duration needed for coordination depending on the strengths of interactions between the columns (Figure 5.1).

The simulations together highlight the critical role played by both the activity inputs and the network couplings in regulating sleep, in the presence of varying activity inputs.

5.3.2 System-Theoretic Analysis

We conclude our study of the AIN with a preliminary system-theoretic analysis of its dynamics. Explicit analysis of the AIN dynamics is both difficult and valuable from a system-theory standpoint. The novelty (and complexity) in the analysis stems from three aspects of the model: 1) the (novel) nonlinear dynamics, 2) the fact that the model represents a distributed system or network defined on an *arbitrary* graph, and 3) the need for characterizing the response to (deterministic or stochastic) *external* signals. Our analysis here characterizes, largely at a qualitative level, the dual role played by the activity inputs and the network couplings in governing sleep-state evolution. We stress here that the this study is, to the best of our knowledge, the first effort to characterize the *external* (disturbance rejection) properties of a complex-network model such as this one. Specifically, we pursue two analyses:

1) We study the internal stability properties of the system, i.e. we study the approach to synchronization (coordination) from an initially-uncoordinated state (and assuming identical activity inputs).

2) we study the external stability or disturbance-response properties of the AIN, in particular verifying that the network coupling permits the AIN to remain coordinated in the presence of activity-input variations.

We characterize the AIN's dynamics for arbitrary network topologies, but for convenience we shall assume that the cortical columns in the AIN have identical internal dynamics (that is, E_i, T_i, and α_i are the same for all cortical columns); it is easy to see that the results naturally transfer to an inhomogeneous network with scaled inputs.

Internal Dynamics: Approach to Synchronization

Here, we analyze the approach to synchronization (perfect coordination) of the AIN when the activity/recovery inputs are all the same, i.e. we study the internal stability of the AIN[*]. We notice that this analysis characterizes return-to-synchronization of the AIN upon local over-stimulation, as well as the coordination when e.g. a lesion has caused the cortical column to become unsynchronized. We shall study the approach to synchronization in two steps, first in the case where the perturbation of the cortical columns from their synchronized state is *small* (i.e., local syncronization) and second in the general case. In the general (large-perturbation) case, we will

[*] We notice that the synchronization problem is in fact a partial stabilization problem in that only state differences need approach the origin; however, it can equivalently be viewed as a stabilization problem in a relative frame, so we loosely use this terminology.

only give a preliminary illustration of the analysis through a simple two-assembly example.

Let us begin by formally defining synchrony for the AIN. To do so, we first find it convenient to define *relative* activity variables. WLOG, we define these relative states with respect to the integrated activity of cortical column 1. In particular, we define the **relative activity variable** z_i for agent i, $i = 2, \ldots, n$, as $z_i = x_i - x_1$. We also define the **relative sleep state** y_i as $y_i = S_i - S_1$. Synchrony is naturally defined in terms of the relative activity variables and relative sleep states:

Definition 5.1. *The AIN is said to be synchronized at time t if $z_i(t) = 0$ and $y_i(t) = 0$ for $i = 2, \ldots, n$.*

As a preliminary step, we formalize that the syncronized cortical columns remain so when the activity inputs to the columns are the same; that is, we note that any synchronized state is a (relative) equilibrium in this case:

Theorem 5.2. *Consider an AIN whose cortical columns have identical internal dynamics (E_i, T_i, and α_i are the same for all i). If the AIN is initially synchronized and the activity/recovery inputs to the cortical columns are identical, then it remains synchronized at all time $t \geq 0$.*

Proof: From the fact that the cortical columns and activity inputs are identical, it follows automatically that $\dot{z}_i = 0$ for $i = 2, \ldots, n$ whenever the columns are synchronized. Thus, synchronization is maintained. □

To present the small-perturbation result, we find it convenient to combine the sleep state variable and activity variable into a single *angular state*, which describes the "distance" along the sleep-wake cycle traveled by the cortical column from a reference point (say the occurrence of waking). Formally, let us define the **angle** θ_i of column i as follows:

- When the cortical column is awake, $\theta_i = 180\frac{x_i-(E_i-T_i)}{2T_i}$.

- When the cortical column is asleep, $\theta_i = 180 + 180\frac{E_i+T_i-x_i}{2T_i}$.

Notice that the cortical column's angle moves from 0 to 180 during the wake state, and from 180 to 360 during the sleep state. This notion of an angle is a clever way to incorporate both the activity variable and sleep state variable into a single scalar, and hence to differentiate between asleep and awake columns that have equal activity variables.

We also find it useful to define angle differences, to describe the "distance" along the sleep-wake cycles between two cortical columns. Specifically, for two cortical columns with angles θ_i and θ_j, we define the **angular distance** $d(\theta_i, \theta_j)$ as follows: $d(\theta_i, \theta_j) = (\theta_i - \theta_j + 180) mod 360 - 180$. This measure equals the shorter of the two angles between the two columns' angles. We note that the network is synchronized at time t if and only if $d(\theta_i(t), \theta_j(t)) = 0$ for all i, j.

Let us now present the asymptotic-synchronization result in the case where the cortical columns are perturbed only a small amount from synchronization. For simplicity in presentation, we describe only the case where the identical input to each cortical column is a positive constant during the wake period and a negative constant during the sleep period, although the analysis generalizes naturally to the case where the columns have identical but non-constant nominal inputs. To highlight the role played by the network, we also exclude the SCN input in the analysis. Here is the result:

Theorem 5.3. *Consider an AIN whose cortical columns have identical internal dynamics (E_i and T_i are the same for all i, and $\alpha_i = 0$), and have identical constant activity inputs $u_i(t) = \bar{u}$ and recovery inputs $u_i(t) = -\bar{u}$. Also assume that the AIN has a connected network topology. Then there exists $M > 0$ such that if $|d(\theta_i(t_0), \theta_j(t_0))| \leq M$ for all i, j at some time t_0, then the synchronized state is attractive, i.e. $d(\theta_i, \theta_j) \to 0$ as $t \to \infty$ for all i and j.*

Proof: Choose any M less than 90, and let $r(t)$ and $s(t)$ be the pair of cortical columns for which

$|d(\theta_{r(t)}(t), \theta_{s(t)}(t))|$ is largest[†], and assume WLOG that $d(\theta_{r(t)}(t), \theta_{s(t)}(t)) > 0$. Notice that the angle of cortical column r can be viewed as "leading" the angles of the other cortical columns, while the angle of s can be seen as "lagging". Now let us consider the time-derivative of $d(\theta_{r(t)}(t), \theta_{s(t)}(t))$. Let us separately consider four cases. If both cortical columns $r(t)$ and $s(t)$ are awake (Case 1) or asleep (Case 2), then all the columns are awake (respectively, asleep). In this case, $\dot{x}_i$ is identical for all columns including $r(t)$ and $s(t)$, and so the time-derivative of $d(\theta_r(t), \theta_s(t))$ is zero. Now consider the case where column $r(t)$ is in the sleep state and column $s(t)$ is in the wake state (Case 3). From the activity-variable update, it is clear that the angle of $s(t)$ is increasing at least as quickly as the angle of $r(t)$ (see Figure 3). We find the same result in the case that $r(t)$ is in the sleep state and $s(t)$ in the wake state. We thus immediately find that the time-derivative $d(\theta_{r(t)}(t), \theta_{s(t)}(t))$ is always nonpositive. In turn, we recover that $d(\theta_{r(t)}(t), \theta_{s(t)}(t))$ is a non-increasing function of time that is also lower-bounded. We thus recover that $d(\theta_{r(t)}(t), \theta_{s(t)}(t))$ approaches a limit.

What remains to be proved is that, in fact, the limiting value of $d(\theta_{r(t)}(t), \theta_{s(t)}(t))$ is 0 and hence the AIN is attractive to a synchronized state. Let us prove this by contradiction. In particular, assume that the limiting value is some $\gamma \neq 0$. Next, notice that the leading column $r(t)$ is the same one for all $t \geq t_0$, since when it is in either the sleep or wake part of the cycle, no other (lagging) column in the same state can overtake it. Now, consider the ordered sequence of lagging columns $r_2(t), ..., r_n(t) = s(t)$, and consider the measure $\sum_{i=2}^{n} |d(\theta_{r(t)}(t), \theta_{r_i(t)}(t))|$. It is easy to see that, during each sleep/wake cycle, this measure strictly decreases by at least a fixed positive amount (which incidentally is linear in γ). Since each relative angle $|d(\theta_{r(t)}(t), \theta_{r_i(t)}(t))|$ is non-negative, we thus recover that all such relative angles, including $|d(\theta_{r(t)}(t), \theta_{s(t)}(t))|$, eventually decrease. Thus, we have a contradiction. $\square$

[†] Notice that these columns apparently may change with time.

Next, let us study the asymptotics of the AIN for arbitrary initial conditions (i.e., for large perturbations). The global stability of nonlinear-oscillator networks such as this one are well-known to be complicated, see e.g. [148–150]. One prominent characteristic of these oscillator networks is the possibility for *mode-locked trajectories*, or in other words equilibrium trajectories that do not correspond to synchronized states. Here, let us demonstrate using a two-cortical-column example that the AIN also can have such mode-locked trajectories, although in this case the mode-locked trajectory is not attractive.

Theorem 5.4. *Consider an AIN with $n = 2$ cortical columns with identical internal dynamics (E_i and T_i are the same for all i, and $\alpha_i = 0$), and identical constant activity inputs $u_i(t) = \bar{u}$ and recovery inputs $u_i(t) = -\bar{u}$. Also assume WLOG that $w_{21} \geq w_{12} > 0$. Now consider that cortical column 2 has an initial angle $\theta_2(t_0)$. Then there is exactly one initial angle for cortical column 1 such that the AIN does not synchronize and instead reaches a periodic orbit; for all other initial angles, the AIN synchronizes.*

Proof: First we notice that, as proved in Theorem 5.3, once we have $|d(\theta_1(t), \theta_2(t))| \leq 90$, then $|d(\theta_1(t), \theta_2(t))|$ is monotonically non-increasing and in fact decreases during each cycle; thus, the two columns become synchronized. Hence, we only need to consider $90 < |d(\theta_1(t), \theta_2(t))| < 180$. In the case that cortical column 2 is "leading", we can easily see that $|d(\theta_1(t), \theta_2(t))|$ also decreases by at least a fixed amount with each cycle. Thus, there exists a time t such that $|d(\theta_1(t), \theta_2(t))| \leq 90$, which implies synchronization.

In the case that cortical column 1 is leading, $|d(\theta_1(t), \theta_2(t))|$ may change non-monotonically. For instance, when the two columns are in different sleep states and both network couplings are activated, $|d(\theta_1(t), \theta_2(t))|$ increases (enlarges); and then when they are still in opposite states and only column 1 drives column 2, $|d(\theta_1(t), \theta_2(t))|$ decreases (shortens). Thus, we expect the possibility for extension-shrinkage-constant half-cycles, which would imply that the cortical columns do not synchronize. In fact, based on the initial angles of the two columns, we have three different outcomes. First, if $\theta_2(t_0)$ is larger than a particular angle, after a half-cycle the distance enlarged is bigger than that shortened, and hence the overall $|d(\theta_1(t), \theta_2(t))|$ increases. Eventually the distance increases beyond 180° after some half-cycles, and column 2 leads instead. Synchronization then is achieved as proved above. Second, if $\theta_2(t_0)$ is smaller than this angle, the distance shrinks after a half-cycle, and eventually decreases below 90°. Again, synchronization is guaranteed. Third, the distance enlarged is exactly the same as that shortened for a half-cycle. In this case, we have mode-locking; in particular, $|d(\theta_1(t), \theta_2(t))|$ is periodic. Simple calculation shows that the initial angle of column 2 that delineates the three outcomes is $\theta_2(t_0) = 270 - 90^\circ \frac{w_{12}}{w_{21}}$. $\square$

Since the mode-locked state is not an attractive one, we notice that in practice the cortical columns will not evolve to this state. However, the existence of the mode-locked state indicates the possibility that the cortical columns will remain away from synchronization for an extended period. This possibility for extended asynchronization may be reflective of such phenomena as part-brain sleep in e.g. dolphins.

Disturbance Response of the AIN

A key postulate of the activity-dependent theory for sleep is that the cortical columns maintain coordination for variable activity levels and inputs, but their sleep state dynamics are also modulated by the activity inputs. This dual task is fundamentally achieved through the interplay of local activity integration at individual columns and network couplings among the columns. Here, we verify that coordination among the columns in the AIN is maintained in the presence of persistent variations in the activity inputs, but the predicted durations of sleep/waking are dependent on the local inputs. The verification of coordination in this case fundamentally requires study of the disturbance-rejection (or *external stability*) properties of the AIN. We stress that a disturbance-rejection analysis constitutes an entirely new focus in the study of oscillator networks (see e.g. [161] for a discussion of why the disturbance rejection of even simple nonlinear systems, let alone networks, is so complicated).

For the AIN, verification of coordination in the presence of input variations (disturbances) follows naturally from the initial-condition analysis of the AIN. In particular, we obtain the following:

Theorem 5.5. *Consider an AIN comprising identical cortical columns that are driven by activity inputs $u_i(t) = \overline{u} + du_i(t)$, where $\overline{u}$ is a strictly positive constant. Let us call the angle difference between the leading column $s(t)$ and the lagging column $r(t)$ at an initial time t_0 by θ_{init}. For each $\theta_{init} < 90$, there exists $\widehat{\beta} > 0$ such that for all $\beta < \widehat{\beta}$, if $||du_i(t)||_\infty \leq \beta$ for all t, then 1) $|d(\theta_{r(t)}(t), \theta_{s(t)}(t))| < 90^o$ for all $t \geq t_0$, and 2) $|d(\theta_{r(t)}(t), \theta_{s(t)}(t))| \leq C\beta$ for all sufficiently large t and for some constant C.*

The proof of the theorem follows automatically from the proof of Theorem 5.3: specifically, the observation that the decrease in $\sum_{i=2}^{n} |d(\theta_{r(t)}(t), \theta_{r_i(t)}(t))|$ over a cycle is linear in its value, together with the fact that the absolute integral of $du_i(t)$ over one cycle is bounded, yields the desired results. We thus omit the details.

The above theorem points out the critical role played by the network coupling in achieving and maintaining a coordinated sleep state: without the coupling, the columns would lose coordination over time. While the columns remain coordinated through the couplings, however, the sleep-state evolution nevertheless is modulated by the activity input.

PART II: NETWORK DESIGN

Part II together with the Part III form the heart of the thesis. In Part II, we systematically develop tools for the design of modern dynamical networks. The tools, which are applicable to both autonomous-agent and infrastructure networks, achieve high performance design of both nodal and edge properties of networks and also allow static (memoryless) controller design.

Part II is organized as follows. Chapter 6 introduces our philosophy of network design for modern network applications—designing high performance controllers that exploit network structure—and presents several results in this direction. Chapter 7 describes the structural approach of designing graph edges so as to maximize the algebraic connectivity. Chapter 8 is concerned with using time-scale techniques to shape network dynamics through designing only a subset of graph edges. Through these developments, we notice that eigenvectors are tightly tied to network performance. Hence, in Chapter 9, we develop further results on properties of dominant eigenvectors associated with non-negative matrices, and in Chapter 10, we find the explicit expression for eigenvector components of a Laplacian graph using only eigenvalues of the graph and its coalesced graph.

6. A NEW FOCUS IN THE SCIENCE OF NETWORKS: TOWARD METHODS FOR DESIGN

In recent years, a realization that networks are ubiquitous in the natural and engineered worlds has led to burgeoning interest in finding commonalities in their structures and dynamics. Here, we introduce a new design focus in this science of networks, by proposing generic methods for *synthesizing network controllers that exploit topological structure.* That is, we motivate a canonical controller synthesis problem for networks that has application in such diverse areas as virus-spreading control and air traffic flow management. We address this design problem by using new techniques from *decentralized control theory.* Specifically, we mesh optimization machinery together with eigenvalue sensitivity and graph theory notions to identify general structural features of optimally-actuated networks. From these features, we are in turn able to explicitly construct high-performance controllers, i.e. ones that best exploit the networks' topological structure. Our general approach for controller design is important both because it provides broad insight into the structure of well-designed networks, and because it contributes engineering solutions in numerous application areas (for instance, reduction of management delays and human-controller workload in air traffic systems).

6.1 Introduction

Historically, efforts have been made to model various network's dynamics, including electrical power system transients [163], metapopulation dynamics [172], and (more recently) biological-oscillator synchronization [147], among many others. In the last twenty years or so, these individual efforts have evolved into a "science of networks" (see, e.g., [158, 162, 164, 173, 174, 177]). That is, scientists and engineers have identified commonalities among the structures and dynamics of many natural and man-made networks. They have thus sought to codify the typical structural features of networks, and in turn to understand how these features modulate the networks' dynamics. There have been numerous outcomes of this science of networks, ranging from statistical prediction of failure event sizes [166] to conceptual discussions about whether the structures of engineered (designed) and natural networks are similar [165]. These studies have been buttressed by an impressive body of analytical research on matrix algebra, which proves valuable for network analysis because interactions between network components (parts) can be codified using matrices. Of particular note in this broad domain are works on *D-stability* and *diagonal Lyapunov stability* (which are useful e.g. in studying population dynamics) [78], efforts to characterize pertinent classes of matrices such as *nonnegative matrices* [74], and the more recently developed field of *spectral graph theory* [73].

While the science of networks has been extensively developed, however, it is our contention that these efforts have largely focused on modeling, that is on predicting networks' structural and dynamic characteristics. What has heretofore *not* been addressed in a general way is the critical task of network management and design, i.e. of efficiently using available resources to improve a network's dynamic performance by exploiting its topology. In myriad application areas, addressing such design tasks would permit significant improvement of network performance (for instance, reduction of air traffic delays, or less intrusive containment of virus spread). Just as networks have commonalities in their structures and dynamics, one would hope that management/design problems for various networks could be abstracted to a common core, and in turn that features of good designs (or of networks upon design) could be identified. Here, we motivate through many examples a canonical problem of network *controller* (or management-scheme) synthesis. Through this canonical problem, we make clear that network controller synthesis problems *require new matrix analysis methods*, and hence develop matrix-theoretic tools for synthesis. In the process, we also identify characteristics of well-controlled networks.

In discussing the need for design, it is worth noting that communication-network engineers (in such domains as ad hoc networking, Internet congestion control, and computer-work elimination, among others) have recently begun contemplating the role played by a network's graph in system design (e.g., [167, 170]) and have studied optimization of certain network (routing or control) algorithms (e.g., [168, 171, 176]). These efforts are meshed with our philosophy that network performance must be shaped/optimized through utilization of the network topology. However, the graph-theoretic studies such as [167, 170] largely focus on *evaluating* the impact of the graph structure on network performance for a particular design, or at best pursue design of a few global graph statistics (e.g. node-degree distribution). On the other hand, the heterogeneous design studies are focused on optimizing static (steady-state) performance measures [176] using e.g. a linear programming formulation, or on optimizing congestion/throughput at single bottlenecks [168, 171]. In contrast, we motivate and address the design of network *dynamics* using fine (local) controls, while still making explicit the role played by the graph topology. We believe strongly that our efforts to *control/shape* the network dynamics by assigning local resources to exploit the graph topology is of significant interest in communications applications, in addition to the network applications that we develop here.

The tools that we develop are inspired by problems from the field of *decentralized control theory* [28, 66], and in fact contribute to research efforts in this field. In contrast to the complex networks literature, control theory has long focused on *designing* dynamics through feedback. Decentralized control theory—which is concerned with systems whose parts each have incomplete ability to observe and actuate the dynamics—can potentially address network management/control problems. Unfortunately, decentralized controllers historically have been designed to achieve performance goals by making individual system components *robust* to any network impact [66]. Such a paradigm is not viable for modern networks, in which individual parts cannot possibly achieve performance aims without using their network connections, and hence modern control strategies *must take advantage of the network topology.* With this requirement in mind, only the seminal work of Wang and Davison and works derived thereof—which give conditions for the existence of *stabilizing* dynamic controllers for a very broad class of decentralized systems—are viable for control of modern networks [28]. In fact, beyond the existence results provided in [28] and in [79] (which considers *static* or memoryless controllers), very little is known about complex network control. In particular, novel methods for *synthesizing* high-performance controllers are badly needed.

Over the last five years or so, there has been a recognition that practical decentralized control requires analyzing the structure and dynamics of complex networks, using graph theory methods [25, 26]. This recognition has led to the formulation of a suite of algorithms and controllers for *autonomous-agent networks* (e.g., vehicle teams, sensor networks) that use the network's graph topology. In turn, various tests for checking whether such algorithms/controllers can complete desired tasks have been developed. However, systematic tools for *designing* high-performance controllers have not been developed even for these applications. Our recent work [29] proposes a methodology for designing high-performance heterogeneous controllers (ones that provide different amounts of resource/actuation) for certain simple autonomous-agent dynamics. Our efforts here show that similar tools—which are based on optimization machinery together with eigenvalue sensitivity and graph algebra notions—are applicable to an important family of decentralized design problems for both autonomous-agent and internally-coupled networks, including some constrained ones. Our results also highlight that very simple and structurally-insightful design tools can be obtained for certain special classes of network topologies, such as ones with nonnegative weights.

The remainder of the chapter is organized as follows: in Section 6.2, we motivate the importance of decentralized design using several applications, including virus-spreading control and air traffic management. In Section 6.3, a canonical design problem that addresses these applications is formulated. Specifically, the problem of designing diagonal matrices D and/or K to minimize a cost that is a function of $D + KG$, where the square matrix G represents the network topology, is introduced. Section 6.4 introduces the analytical methods used to solve these design problems, and summarizes the design—some details can be found in Chapter 2. Finally, examples are pursued in Section 6.5.

6.2 Controller Synthesis Problems in Modern Networks

Decentralized controller design is a critical need in myriad network applications. Here, we formulate network controller synthesis problems in some detail for two applications: 1) virus spread control and 2) coordinated air traffic flow management. We also briefly describe synthesis problems in two other domains, namely autonomous vehicle control and iterative numerical solution of systems of linear equations. Each synthesis task is an example of the canonical problem pursued in this chapter. More extensive formulations of these four design tasks can be found in [1, 2, 29].

6.2.1 *Spatially Heterogeneous Virus Spreading Control*

The impacts of recent biological virus epidemics (SARS, foot-and-mouth disease) and computer viruses highlight the need for mathematically modeling virus spread, and in turn developing controllers that reduce spread with limited resources. Epidemic control can be viewed as reducing the **basic reproduction ratio** R_0 (defined as the average number of secondary infections produced during an infected individual's infectious period, when the individual is introduced into a population where everyone is suspectible [30]). For virus spreads in heterogeneous populations, R_0 is often computed as the dominant eigenvalue[*] of the **next generation operator (matrix)** of a **multi-group model** (see [50]). Although spatial structure is believed to greatly impact epidemic spread [24], there is little work in the literature on designing controls (e.g. isolation and quarantine) that exploit the network's topological structure to reduce epidemic spread with limited resources. We seek to design limited resource allocations (controls) that optimally reduce R_0, by exploiting the topological structure.

In the interest of clarity, we pursue design for a basic (susceptible-infected-susceptible, or SIS) heterogeneous multi-group model, with the understanding that similar designs can be developed for more detailed models. Specifically, the multi-group model for spatial epidemic spread developed in [24] can be generalized to admit spatially-heterogeneous control. In this case, the next generation matrix is

$A = \beta T \, diag(T_i r_i N_i) \left(diag(h_{ii}) + diag(c_i) \begin{bmatrix} 0 & h_{21} & \dots & h_{n1} \\ h_{12} & 0 & \dots & h_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ h_{1n} & h_{2n} & \dots & 0 \end{bmatrix} \right) [(diag N_i)]^{-1}$, where β is the transmission coefficient (incorporating infectiousness and average contact rate), T is the nominal infectious period, N_i is the population in region i, and each h_{ji} is a corrective term that accounts

[*] R_0 is a real number since an interaction network structure is non-negative and irreducible.

for the relative rate of inhomogeneous mixing between regions i and j. The region-specific scaling parameters T_i, r_i and c_i ($\in [0,1]$) are the controls that admit design:

1) The parameter r_i scales the contact rate of individuals in region i, which decreases virus spread both locally and to spatial neighbors. Note that r_i can be reduced by isolation of closely connected and fairly isolated groups such as a school/college, or by vaccination.

2) The parameter c_i scales the contact rate of individuals from outside regions to a region i. The external contact rate factor c_i can be reduced by prohibition of travel from other regions to region i, or similarly isolation of arriving travelers for longer than the incubation period.

3) We allow the average duration of the infectious period in each region i to deviate from T by a factor of T_i, which can be reduced by shortening the time between symptom appearance and hospitalization in region i.

Control measures such as isolating people who may have contacted an infected individual (e.g. through isolation of a neighborhood) reduce both T_i and r_i.

Our aim is to design $T_i r_i$ and/or c_i to minimize the basic reproduction ratio (i.e. the dominant eigenvalue of A) subject to the individual constraints on these parameters and subject to total resource constraints ($\sum T_i r_i$ or $\sum c_i$ larger than a lower bound). For instance, to design $T_i r_i$, we can form the following constrained optimization problem:

Problem 1. Design a diagonal matrix K such that $\lambda_{max}(KG)$ is minimized, where K is subject to the following constraints:

1) $tr(K) = \sum K_i \geq \Gamma$;

2) $0 \leq K_i \leq 1$ for all i.

Here, the **actuation matrix** K equals $diag(t_i r_i)$, the **topology matrix** G captures the remainder of the dynamics; the constraints on $T_i R_i$ yield that $0 \leq K_i \leq 1$ and that the K_i in total exceed a **lower bound** Γ (since much resource is needed to make K_i small), see [1] for further details.

In the case that we design c_i, we can similarly formulate the following design problem:

Problem 2. Design a diagonal matrix K such that $\lambda_{max}(D + KG)$ is minimized, where K is subject to the constraints that

1) $tr(K) \geq \Gamma$;

2) $0 \leq K_i \leq 1$ for all i.

Equivalently, we address the problem of reducing R_0 to a desirable value (e.g., the critical threshold $R_0 = 1$) using minimal amount of resource. Such a systematic design of these inhomogeneous control parameters can provide guidance for effective epidemic control.

Automaton models that capture the evolution of *individuals'* infection states through interaction are also widely used in epidemic modeling, especially in characterizing computer viruses. We have also formulated the problem of designing optimal spatially-heterogeneous controls in the context of an automaton model (in which probabilities that individuals are infected evolve). This decentralized design problem is similar to the one detailed above, see [1] for details and [23] for background. We stress here that the design problems introduced here are decentralized ones, in that the controls act on populations traveling to or from one region, or on individuals.

6.2.2 Coordinated Air Traffic Flow Management

One major complexity in air traffic flow management (TFM) is that flow restrictions (e.g., en route miles-in-trail restrictions, ground delay programs) only act locally, and yet must be coordinated to achieve network-wide objectives [92]. Appropriately designing decentralized management strategies can permit significant reduction in traffic delays, while simultaneously reducing human-controller workload and reducing safety concerns.

The bulk of the literature on coordinated flow management either focuses on how strategies can be implemented in practice [89], or pursues design of coordinated TFM assuming that aircraft flows are deterministic and further that individual aircraft can be scheduled at each restriction [93]. In fact, uncertainties (due to take-off time variations and weather) critically impact flows, and also many flow restrictions simply enforce spacing between aircraft rather than permitting arbitrary scheduling. With these concerns in mind, we have developed a family of abstractions for restrictions acting on stochastic flows, each of which demonstrate the essential tradeoff between downstream variability and upstream backlog/delay effected by a restriction [2]. Here, we incorporate a highly abstracted algebraic model for a restriction into an Eulerian network flow model (which captures take-off/splitting/merging flows) to pose a network-wide flow-management design problem. This formulation allows us to design restriction strengths, and further yields graphical insights into the structure of good flow management designs.

Our model captures aircraft-count variabilities before and after n **boundary restrictions** (which we denote v_i and w_i respectively), as well as the backlogs B_i caused by these boundary restrictions. We assume a simple linear dependence of the downstream variability and upstream backlog (in steady-state) on the strength of the restriction. Specifically, for boundaries within the airspace, we have $w_i = (1 - a_i)v_i$, and $B_i = \gamma a_i \lambda_i$, where $a_i \in [0, 1]$ is the restriction strength, γ is a scaling factor, and λ_i is the mean number of aircraft passing the boundary. For boundaries i corresponding to flows entering the airspace, we assume variabilities corresponding to Poisson process flows, and no backlog. We also model splitting/merging of flows between boundaries by relating each inflow variability to the outflow variabilities of neighboring restrictions, specifically as $v_i = \sum_{i=1}^{n} w_j g_{ji}$ where $g_{ji} > 0$ implies that there is a flow from restriction j to restriction i. We notice that some restriction strengths a_i can be designed while others are fixed.

The design problem of interest is to set the parameters a_i to get desirable variability in downstream flows or regional counts, while maintaining small backlogs (so that total counts in upstream regions do not exceed thresholds, and aircraft are not subject to long delays). Specifically, a natural performance measure is one that captures the total impact of the control on the aircraft counts in a couple critical regions, by combining the variabilities of flows in the regions with the backlogs in the regions caused by restrictions. Many such measures are well-approximated as being linear or quadratic in the backlogs B_i and variabilities w_i.

The design problem posed above can straightforwardly be rewritten as the following linear algebraic design problem:

Problem 3. Design a diagonal matrix Z so as to minimize a cost measure that is quadratic in the entries of $I - Z$ or Z^{-1} and in $\mathbf{w}$ (e.g., $(Z^{-1}\mathbf{1})^T(Z^{-1}\mathbf{1}) + \mathbf{w}^T\mathbf{w}$) subject to the constraints that

1) the system of n equations $(I - Z\widehat{G})\mathbf{w} = Z\widehat{\lambda}$ is satisfied;

2) each diagonal entry of Z is constrained to be in $[0, 1]$.

Here, the **topology matrix** $\widehat{G}$ can be simply computed from the routing parameters g_{ji}, the **input vector** $\widehat{\lambda}$ depends on the boundary flow rates λ_i, and the **actuation matrix** Z contains the designable restriction strengths a_i. We stress here that this design problem is also a decentralized one, in that each restriction impacts only local flows and yet a global objective is pursued.

6.2.3 Other Applications

Controlling Autonomous Vehicle Teams

Autonomous vehicle teams must move in formation for such diverse purposes as sweeping for mines, exploring underwater geologic formations, or studying the Martian landscape. Physical constraints, cost requirements, and security concerns often necessitate that the vehicles only make decentralized measurements, i.e. each vehicle can only partially observe the entire network's state through some set of (in general non-absolute) position and velocity observations. The decentralized controller synthesis task then is to design the force inputs to each vehicle from its observations. Assuming a static (memoryless) feedback control paradigm, a high-velocity-gain controller can generally be used (see [29]). In this case, the high-performance design problem can be abstracted to the decentralized design problem of selecting diagonal matrix K to place the eigenvalues of KG at desirable locations, where G describes the agents' position observation topology. Analogous problems also arise in designing e.g. distributed estimation algorithms for sensor fusion.

Designing Preconditioners for Numerical Analysis

Preconditioners are used to speed up convergence of linear iterative algorithms, such as can be used to solve the system of linear equation $Gx = b$ [169]. Diagonal preconditioners are especially common because the extra computation required for their implementation is small. Essentially, a preconditioner's performance can be measured in terms of a **condition number** (typically either $\frac{|\lambda_{max}(KG)|}{|\lambda_{min}(KG)|}$ or $\frac{|\sigma_{max}(KG)|}{|\sigma_{min}(KG)|}$, where $\lambda()$ and $\sigma()$ are the eigenvalues and singular values, respectively, and K is the preconditioner). Thus, an optimal diagonal preconditioner is a diagonal matrix K that minimizes one of the condition numbers, subject to a constraint on the signs of the eigenvalues. Finding the optimum is a decentralized design problem, in that we are optimizing gains (weights) that have "local" influence (change one row of G). The methodologies developed here can give insight into the structure of the optimal diagonal preconditioner.

6.3 The Decentralized Design Problem

The controller design tasks introduced in Section 6.2 are all of the following form: a scalar cost $c(D + KG)$ must be minimized with respect to the diagonal **actuation matrices** D and K, where the **topology matrix** $G \in R^{n \times n}$ captures the interactions among components in a network, and the diagonal entries of D and K are subject to one or more constraints. We view this design problem as canonical but widely applicable, and hence seek methods for solving it.

Let us first elaborate on each aspect of this design problem, and distinguish the problem from those previously addressed in the matrix-analysis literature.

The Topology Matrix G represents the strengths and polarities of interactions among network components, such as traffic flow densities between boundaries in the airspace or infection-spread probabilities for networked computers. The topology matrix may in general be an arbitrary real, square matrix, but for many applications is known to be more structured, for instance nonnegative, symmetric, and/or positive definite [74]. We notice that the network's interaction topology can be illustrated using a **graph** with n nodes, and with weighted and directed edges defined from the topology matrix.

The Actuation Matrices D and K directly or indirectly specify the control effort provided to each component in the network, for instance the quarantining/vaccination efforts provided to each district to prevent biological virus spread or the actuation of autonomous vehicles in a team. We stress here that D and K capture distributed control efforts in that they locally or partially change the structure of G. The additive matrix D describes entirely local actuations or topology changes in the network (for instance, use of a computer virus detection and removal program). Meanwhile, the multiplicative actuation matrix K describes actuations or topology changes that proportionally impact all components connected to a particular component (for instance, a restriction in the air traffic system, which proportionally reduces flows to all downstream restrictions). The entries in the diagonal matrices D and K may be variously constrained; particularly common are instances where the entries are constrained to a simplex. Such a constraint results for instance when the total resource allocated for virus control is lower-bounded, and the resource permitted in each region is bounded in an interval. We note that the special case where $D = 0$, and hence the topology matrix has the form KG, is relevant in several application areas.

The Cost Function c() is application-specific. However, the cost is very often associated with a dynamics defined on the **actuated network topology** $D + KG$, as in all the examples in Section 6.2. In fact, the cost functions for the examples in Section 6.2 are all based on a linear (or linearized) dynamics defined on the network. That is, the components in the network each have dynamically-varying states x_i associated with them (e.g., numbers of aircraft or probabilities of infection), such that $\mathbf{x}^{\mathbf{T}} \overset{\triangle}{=} [x_1\ x_2\ ...\ x_n]$ is governed by the differential equation $\dot{\mathbf{x}} = (D + KG)\mathbf{x} + \mathbf{w}$ or $\mathbf{x}[k+1] = (D + KG)\mathbf{x}[k] + \mathbf{w}$, where $\mathbf{w}(t)$ specifies external inputs to each component. Since these dynamics (whether representing virus spread or vehicle movement) are of fundamental interest for the applications, it is no surprise that the optimization cost is often defined based on the solution of the above linear differential/difference equations. Classically, the transient solutions of these equations are written as a sum of response terms each governed by an *eigenvalue* of the matrix $D + KG$, and hence the *stability* and *settling rate* of the dynamics depend on the eigenvalue locations in the complex plane. This motivates design of actuation matrices to optimize a dominant eigenvalue-based parameter (often the least negative among the eigenvalues' real parts in the continuous-time case, and the largest among the eigenvalues' magnitudes in discrete-time). Alternatively, the steady-state solutions to above linear equations for a constant persistent input $\mathbf{w}(t) = \overline{\mathbf{w}}$ (given by $-(D+KG)^{-1}\overline{\mathbf{w}}$ and $(I-(D+KG))^{-1}\overline{\mathbf{w}}$ respectively, assuming stability) may define the cost[†]. It is worth noting that, even if dynamics are not of interest, the eigenvalues of $D+KG$ give much insight into the network's topological structure (see [73]) and hence eigenvalue design is commensurate with topology design.

Connection to Existing Matrix Algebra Methods This design problem unfortunately cannot be addressed by the existing matrix algebra techniques developed for complex networks. Spectral graph theory methods and methods for particular matrix classes largely focus on analyzing a given topology (matrix), rather than seeking high-performance topologies among a class of possibilities [73,74]. Meanwhile, the D-stability and diagonal Lyapunov stability tools give conditions under which all gains within a class achieve stability (essentially a robustness result), rather than identifying a single high-performance design.

6.4 Novel Methodology for Network Controller Design

Fundamentally, our control-theoretic method for solving the described network design problem is based on recognizing that the actuated topology matrix $D+KG$ has a special structure when the optimal actuation matrices $D = D^*$ and $K = K^*$ are chosen. That is, optimal or near-optimal actuations serve to alter the topology matrix to a particular form; our approach is to identify this

[†] The careful reader will note that the asymptotic cost for the air traffic example is slightly different in form. The difference results because the inflow variabilities can be easily eliminated from the original expressions for asymptotics in this case. Either form can serve as a starting point for design.

form, which in turn permits us to compute the optimal actuation explicitly with a finite search. We have applied this three-step methodology to design both D and K for various cost functions, including those discussed in Section 6.2. In order to make the presentation accessible to a broad audience and also in the interest of space, we only include an overview of the methodology and a brief description of the results here. We kindly refer the readers to [1,2,29] for details. Specifically, we achieve the design through the following steps:

1. Finding the Structure of the Optimally Actuated Topology We apply standard constrained optimization techniques (i.e., Lagrange multiplier techniques [71]) to obtain structural characteristics of the optimally-actuated topology matrix $D^* + K^*G$. Specifically, by taking derivatives of the *Lagrangian* (which is formed from the cost function and constraints, we can obtain a set of equations that are necessarily satisfied by the optimization parameters (in our case the diagonal entries of D and K). Here, we are very often concerned with costs based on dynamics and especially eigenvalues, and hence taking derivatives with respect to the the optimization parameters involves invoking *eigenvalue sensitivity results* [70]. Doing so, we obtain structural conditions on the *eigenvectors* of $D^* + K^*G$, i.e. when the optimizing actuations are used. This special structure implies to us that critical dynamics of the optimized network are excited by, and propagate in, certain spatial patterns. Identifying these common structural and dynamic characteristics of optimally-actuated topologies is one of the key contributions of this research.

Let us list the structure of the optimally-actuated topology, for several example costs and constraints:

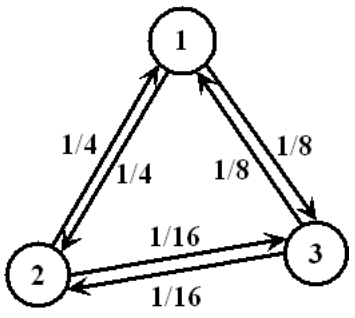
- Consider designing D to minimize the **dominant eigenvalue** of $D + KG$, where each gain D_i is constrained to an interval and $\sum D_i$ is also constrained. For the optimizing actuation

$D = D^*$, the following is true for each $i \in 1, \ldots, n$: either D_i^* is at a limiting value, or the ith **participation factor** (the product of the left and right eigenvectors' ith components[‡]) of the dominant eigenvalue of D^*+KG is equal to a fixed constant. That is, the optimizing actuation serves to equalize the participation factor of each network component (from actuation to response) in the dominant-eigenvalue dynamics, to the extent permitted by the constraints (Figure 6.1). We refer the readers to Theorem 2.1 in Chapter 2 for a formal statement of this result and also the proof. Briefly, the result is developed as follows: for an optimizing actuation, from Lagrange multiplier techniques we obtain that either the sensitivity of the dominant eigenvalue to each parameters is 0, or the parameter hits the constraint boundary. Working from the fact that the dominant eigenvalue's sensitivity to each parameter (that is not at a boundary) is 0, we obtain from the left- and right- eigenvector equations (with a little algebra) that the corresponding participation factors are identical.

- For optimal diagonal preconditioners K^*, the ith participation factors of the maximum and minimum eigenvalues of K^*G are equal in absolute value (i.e., they can differ only in sign). The development of this result is analogous to the one for the dominant eigenvalue of $D+KG$, see Theorem 3 in [29] for details.

- For the asymptotic cost $\mathbf{1}^T(I-(K+G))^{-1}\bar{\mathbf{w}}$ subject to a simplex constraint, the product of the asymptotic state $\overline{x}_i$ with the sum of the ith column of $(I-(K+G))^{-1}$ is equal for all i. The methodology for deriving this result is again very similar to that for the above costs, with only the exception that the sensitivity of the quadratic cost rather than of an eigenvalue is used.

[‡] Participation factors were originally introduced in the study of electric power networks, to codify how the dynamics of a particular mode (eigenvalue) of a system are impacted by each network component, see [175].



$$G = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{4} & 0 & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{16} & 0 \end{bmatrix},$$

$D_i \in [0,1]$, $\sum D_i \geq \Gamma$

Optimal Designs:

1) $\Gamma = 1.5$: $D_1^* = 0.4167$, $D_2^* = 0.4792$, $D_3^* = 0.6042$, $\lambda_{max}^* = 0.7917$ and $v_{max}^* = [1\ 1\ 1]^T$. (Individual constraints not reached, all components participate equally.)

2) $\Gamma = 2.9$: $D_1{}^* = 0.928$, $D_2{}^* = 0.972$, $D_3{}^* = 1$, $\lambda^* = 1.2661$, and $v_{max}^* = [1\ 1\ 0.7047]^T$. (Components 1 and 2 participate equally, Component 3 at constraint.)

Fig. 6.1: The structure of the optimally-actuated topology is illustrated, for the cost $\lambda_{max}(D+G)$.

2. Computing the Optimal Actuations By invoking these structural characteristics of the optimally-actuated topology, it turns out that we can limit ourselves to solving a finite (albeit in general large) set of systems of equations to find the optimum actuations D^* and K^*. Thus, we can give explicit formulae for the optimizing actuations, or develop finite search algorithms for finding them. We note that these reformulations often require use of basic eigenanalysis or of algebraic graph theory notions such as the Courant-Fisher Theorem [73]. For instance, for optimization of $\lambda_{max}(D+G)$ with respect to D, the structural result described above allows us to find D^* by choosing each D_i as 0, 1 or $0 < D_i < 1$. By placing sets of D_i at 0 and 1 respectively and then solving for the remaining gains based on the structural result, we find that the optimal actuations can be computed by solving at most 3^n systems of equations (see Theorem 2.4 in the Chapter 2).

3. Simplifying Computation for Special Classes of Topologies For special classes of topology matrices (e.g., nonnegative, diagonally-symmetrizable, and/or positive definite [72,74]), the dimension of the computation of D^* or K^* can often be reduced even further. These simplifications result because the special topology guarantees convexity of the optimization (so local optimality yields global optimality), and/or because the eigenvectors of $D+KG$ are further structured in these cases. The additional structure also often permits simple interpretation of the optimizing actuations.

For instance, again consider optimizing the dominant eigenvalue of $D+KG$ with respect to D, subject to the simplex constraints. When G is nonnegative and diagonally-symmetrizable (which are reasonable assumptions in describing e.g. virus spread), the iterative algorithm for finding D^* in many cases requires at most n solutions of systems of equations, rather than 3^n. Specifically, the optimal actuation can be found by first finding the optimum when the constraints on individual D_i are relaxed. Then, if too much actuation is required for some i, these D_i can simply be set to their limiting value and the optimum recomputed (with the process repeated if other gains exceed their bounds), see Theorem 2.6 in the Chapter 2. An eigenvector majorization result allows us to justify that such a simple algorithm can find the optimum. Recently, we have been able to obtain similar results on majorization of eigenvector components for arbitrary irreducible nonnegative matrices (not just diagonally-symmetrizable ones), which permit extension of this algorithm to such topologies [8].

In terms of simplifying interpretations, for positive and symmetric topologies, one sees that the requirement of equalized participation factors corresponds to seeking equalization of row sums, or individual components' impacts, as best as is permitted by the constraints (Lemma 2.5 in Chapter 2). Further, the proof of Theorem 2.6 also clarifies that, for nonnegative topologies, a network component that cannot fully participate due to its resource limits is supported through extra resource allocation at its neighbors. This interpretation is, for instance, valuable in virus-spreading applications.

6.5 Examples

6.5.1 Control of the Hong Kong SARS Epidemic

The article [24] developed a multi-group model for the propagation of SARS in Hong Kong's 18 districts, and identified the homogeneous control needed to reduce the basic reproduction ratio R_0 to 1. Our approach allows us to find the optimal heterogeneous control that uses the same total resource amount (see also [1]); this control reduces the basic reproduction ratio to 0.64. Equivalently, it is easily shown that $R_0 = 1$ can be achieved even when the total control resource is reduced to 79% of the one with equal allocation. We kindly ask the reader to see Figure 2.2 and Table 2.2 in Chapter 2 for the details.

These studies show that an epidemic's spread can be stopped more quickly with the same total control resources, through heterogeneous allocation. This intelligent allocation takes advantage of the spatial structure of the population by placing more control resources in the highly connected parts of the network such as Districts 5 and 7 in Hong Kong, and fewer resources in isolated parts such as District 1 (see Figure 2.2 and Table 2.2). In this way, the limited control resources are best able to reduce the rate at which the epidemic diminishes. Such a control would significantly reduce the impact on people's daily lives in some districts (which have less control resources allocated) and overall, while still stopping the virus spread quickly.

For illustration, we also consider how the heterogeneous resource allocation changes when the local mixing rate h_{ii} is increased (compared to the mixing rate of neighboring districts h_{ij}). As expected, increasing the local mixing rate makes the resource allocation more homogeneous (Table 2.2).

6.5.2 Traffic Flow Management Design

We also use our methodology to design *en route* restrictions for air traffic flow management (see illustration of the restriction model in Figure 3.13 of Chapter 3). Specifically, we consider restriction design for several congested regions in an air traffic network, with merging and splitting flows (Figure 3.13). Flow management is required to reduce the variability in regional aircraft counts (so as to prevent capacity violations), but without causing excessive backlog and delay. As discussed in Section 6.2, the flow management problem can be abstracted to the decentralized design problem studied here. We can thus use the design methodology to optimally choose restriction strengths in this example, as shown in Figure 3.13. In agreement with detailed modeling-based and operational studies [2], our approach shows that restrictions are most effective when placed upstream on flows that impact multiple congested regions. The additional value provided by our approach is the ability to design restriction strategies for arbitrary flow topologies, and the possibility of optimizing/improving these strategies.

7. ON THE STRUCTURE OF GRAPH EDGE DESIGNS THAT OPTIMIZE THE ALGEBRAIC CONNECTIVITY

We take a structural approach to the problem of designing the edge weights in an undirected graph subject to an upper bound on their total, so as to maximize the algebraic connectivity. Specifically, we first characterize the eigenvector(s) associated with the algebraic connectivity at the optimum, using optimization machinery together with eigenvalue sensitivity notions. Using these characterizations, we obtain an alternative finite-search algorithm for finding the optimal design in tree graphs that is quadratic in the number of vertices, and further address update of the design upon addition of a new vertex. We also obtain a suite of results concerning the topological and eigen-structure of optimal designs for bipartite and general graphs. In turn, we obtain a lower-bound on the optimal algebraic connectivity in terms of the graph's diameter, and also describe how our structural insights can inform and be meshed with numerical solution techniques. Finally, an example concerning flow-network design is presented.

7.1 Introduction

The burgeoning importance of large-scale dynamical networks in our day-to-day lives has brought about a keen interest in graph theory in the engineering community. While the interface between graph theory and dynamical-network analysis has been widely studied, the problem of *designing* networks and their controllers to exploit a their topological structure remains challenging. It is our perspective that such network and controller designs are of critical importance, and so we have engaged in a major effort to develop a systematic graph-based methodology for design [1,5,10,29]. Here, we enrich our methodology to address the particular problem of designing the edge weights in a graph, so as to optimize an associated network's dynamics.

Edge-weight design in graphs is needed for a range of network applications, including sensor network algorithm-design, assembly of autonomous vehicle teams, virus-spreading control, and optimization of numerical sampling tools, among others [5,29,80]. Of particular interest, Boyd and his co-workers have identified and given a common framework for several important edge-weight design problems [80,178–184], and in turn have used a semi-definite programming (SDP) methodology to find optimal edge-weight designs. While the SDP methodology does provide solutions to the edge-weight design problems given some (non-trivial) regularity conditions, it does not directly yield insight into graphical and dynamical properties of high-performance designs; such insights are critical both because they help to characterize the behaviors of the optimally designed systems, and because they allow us to identify/construct good designs even when precise optimization is not possible. Here, we use a methodology that meshes optimization, spectral graph theory, and eigenvalue sensitivity notions to obtain structural results concerning optimal edge designs. Our approach enriches and informs the existing numerical (SDP-based) characterizations of the optimal edge design.

The methodology for edge-weight design introduced here is closely connected to the techniques for static controller design that we introduced in [1,5,29]. In [29], we motivated and addressed an optimal node-design or scaling problem, in particular addressing the design of diagonal matrix K to optimize a performance measure defined from the matrix KG. In [1,5], we went one step further and showed that the approach can be used to solve some optimal decentralized design problems with constraints, i.e., the design of a diagonal matrix D or K to minimize the dominant eigenvalue of $D+KG$ subject to constraints on D/K. That problem is common in the decentralized control of infrastructure network dynamics such as epidemic spread or air traffic flow. We have also given a complementary methodology for designing *dynamic* decentralized controllers using a multiple-derivative and multiple-delay paradigm, in [10].

Our efforts here also constitute a contribution to algebraic graph theory research. While the bulk of the literature in this domain is focused on analyzing particular graphs rather than designing them, Fiedler has addressed an optimal edge-weight design problem. Fiedler's important work provides structural understanding of an optimal edge-weight design (in particular, for an eigenvalue design of tree-graph Laplacian matrices) [185]. Also of interest, a few recent efforts have obtain structural results for other graph design problems (e.g., the fastest mixing Markov chain problem), but for limited classes of graphs such as paths [178, 179]. Meanwhile, other efforts in the algebraic graph theory community have sought structural insight into optimal edge-weight designs, starting from the SDP formulation [186–189]. From this algebraic graph-theory perspective, our efforts serve to 1) further the structural analysis given in [178–180], 2) achieve design for a much broader class of edge-weight design problems and graphs, and 3) clarify that structural insights into the optimal design in fact yield and/or permit refinement of good algorithms for edge-weight design.

In this chapter, we study one canonical graph-edge design problem: namely, that of selecting the edge weights subject to an upper bound on their total, so as to maximize the graph's *algebraic connectivity* (i.e., the second-smallest eigenvalue of the Laplacian matrix associated with the graph). The problem is motivatived and introduced in Section 7.2. We then give several structural characterizations of the eigenvector associated with the optimal algebraic connectivity (Section 7.3); these characterizations summarize and extend the early work of Fiedler [185]. We use this structural interpretation to give an atternate design for tree graphs that can be constructed in $\mathcal{O}(n^2)$ operations, and also identify an $\mathcal{O}(n)$ algorithm for updating the design upon addtion of new vertex (Section 7.4). Next we also specify several properties of the optimal solution for more general classes of graphs, with a particular focus on recognizing the presence of repeated-eigenvalue solutions as well as multiple optimal designs (Section 7.5). We also describe how the structural approach here can inform and be meshed with numerical (e.g., semi-definite programming-based) tools for design. Finally, we illustrate our development with an example on flow-network design (Section 7.6).

The structural approach to graph-edge design studied here can be adapted to numerous other design problems of interest. We are in the process of applying the graph-edge design methodology for two other canonical problems—namely, 1) optimization of the dominant eigenvalue of (both symmetric and asymmetric) positive matrices defined on a graph, and 2) optimization of the mixing rate of a Markov chain (i.e., optimization of the subdominant eigenvalue of a stochastic matrix defined on a graph). In the interest of space, these results will be presented in future work.

7.2 Review and Problem Formulation

We consider the problem of designing the edge weights in a graph, to maximize the **algebraic connectivity** or **Fiedler eigenvalue** of the graph, i.e. the second-smallest eigenvalue of the associated Laplacian matrix.

Formally, let us consider a non-negatively weighted, undirected graph G with vertex set $V = \{1, \ldots, n\}$, edge set E (where edges are specified as unordered pairs of vertices), and nonnegative weight $k_{ij} = k_{ji}$ associated with each edge $\{i, j\} \in E$. We recall that the **Laplacian matrix** L of the graph G is defined as follows: $L = D - K$, where $k_{ij} \overset{\triangle}{=} 0$ for $\{i, j\} \notin E$, $K \overset{\triangle}{=} [k_{ij}]$, and D is a diagonal matrix with ith diagonal entry given by $\sum_{j=1}^{n} k_{ij}$. We note that the Laplacian matrix L is a symmetric positive semidefinite matrix, and so has real eigenvalues with the minimum one equal to 0. The second-smallest eigenvalue of the Laplacian matrix, known as the **algebraic connectivity** (or, more informally, the Fiedler eigenvalue in honor of the extensive study of it by M. Fiedler), is of wide interest to both graph theorists and dynamical-network analysts.

Here, we aim to assign the nonnegative edge weights k_{ij} for $\{i, j\} \in E$ so as to maximize the algebraic connectivity λ_2, subject to an upper bound Γ on the total edge weight: $\sum_{\{i,j\}\in E} k_{ij} \leq \Gamma$. We refer to this edge-weight design problem as the **Laplacian edge design problem** and refer to E as the designable edge set. We call a design that achieves the maximum an **optimal edge design**, and use the notation k_{ij}^* for the edge weights in such a design; analogously, we use the notation λ_2^* for the **optimal algebraic connectivity** (i.e., the algebraic connectivity for the optimal edge design). We refer to eigenvectors associated with the algebraic connectivity at the optimum as **optimized eigenvectors**. In the case where the optimal eigenvalue is non-repeated and use the notation $\mathbf{x}^*$ for the unique (to within a scale factor) eigenvector. When the optimal eigenvalue is repeated m times, we denote a basis for the corresponding eigenspace as $\mathbf{x}^*(1), \ldots, \mathbf{x}^*(m)$, and use the notation $\mathbf{x}^*$ for vectors in the span of $\mathbf{x}^*(1), \ldots, \mathbf{x}^*(m)$ We also find it convenient to refer to the set of edges in a design that have strictly positive weights as the **non-zero edge set**, and to use the notation E_d (respectively, E_d^* for an optimal design) for this set. We notice that $E_d^* \in E$.

The Laplacian edge design problem posed above was first studied by Fiedler in [185], wherein he coined the term **absolute algebraic connectivity**[*] for the optimal λ_2. In particular, the article [185] gives a structural characterization of the eigenvector of the Laplacian L associated

[*] To be precise, Fiedler uses the term for the case where $\Sigma_{i,j\in E} k_{ij} \leq n$, so we use different terminology here; however, notice that the more general problem trivially reduces to this one through scaling.

with the optimal algebraic connectivity when it is nonrepeated, and in turn finds the optimal edge design for tree graphs (including ones with repeated algebraic connectivity) in terms of the *variance* of the graph. More recently, Boyd and his co-workers have used semi-definite programming methods to obtain the absolute algebraic connectivity for general graphs, in the process obtaining a set of optimal edge weights. Also of interest, the article [185] exposes that the absolute algebraic connectivity can be found as a solution to an embedding problem, and in turn relates absolute algebraic connectivity to the graph separators.

The structural approach that we pursue here is very closely connected with the eigenvector structure-based approach of Fiedler. However, our focus here is not only on characterizing the absolute algebraic connectivity but also explicitly constructing and characterizing the optimal edge design (or designs). To this end, we clarify that a polynomial-time (in fact, $\mathcal{O}(n^2)$) algorithm can be used to find the optimal edge design for trees, as an alternative to the design strategy given in Fiedler. We also characterize the topological structure of the optimal design in this case and show how the design can be obtained with $\mathcal{O}(n)$ effort upon addition of a new vertex. Further, we obtain several characterizations of the family of optimal edge designs, as well as bounds on the absolute algebraic connectivity, for bipartite and general graphs.

7.3 The Structure of the Optimized Eigenvector

Let us review and extend the structural characterization of the optimized eigenvectors given in [185]; Specifically, we first review the result from Fiedler's work on the optimized eigenvector, in the case where the optimal algebraic connectivity is assumed non-repeated.

Theorem 7.1. *Consider the Laplacian edge design problem. For an optimal edge design such that the algebraic connectivity is a non-repeated, the following condition holds: for each* $\{i, j\} \in E$, *either 1) $k^*_{ij} = 0$ or 2) $|x^*_i - x^*_j| = u$ and $k^*_{ij} > 0$, where u is a positive constant.*

In words, the theorem states that the absolute difference in eigenvector components along each edge in E used in the optimal design is identical, as long as the algebraic connectivity is non-repeated. We have proved the result using the standard Lagrange multiplier machinery together with eigenvalue sensitivity ideas, as an alternative to the majorization proof in [185]. The proof methodology highlights the analagy of our graph-design methodology with our previous results on optimal scaling and static decentralized controller design [1, 5, 29].

The above result fully characterizes the optimized eigenvector, in the case where the optimal algebraic connectivity is a nonrepeated eigenvalue of the Laplacian matrix. We note that this optimized eigenvector has numerous other structural properties that are common to all eigenvectors associated with the algebraic connectivity of a Laplacian, see e.g. [73] for details. We also note (as was also noted in [185]) that the above structural result implies that the non-zero edge set of the optimum forms a *bipartite graph*, if indeed the algebraic connectivity is not repeated. We will show in the following subsections that the above structural result facilitates design and provides a variety of insights into the optimal design.

As our later discussion will clarify, the optimal solution commonly has repeated algebraic connectivity, and so characterizations of the optimized eigenvectors associated with a repeated algebraic connectivity are needed. Broadly, characterizing the opitmized eigenvectors for the repeated-eigenvalue case is complicated because of the difficulty in finding sensitivities of repeated eigenvalues to perturbations, see e.g. [190]. Here, we review the explicit characterization of the eigenvectors given by Fiedler for the case that the designable edge set forms a tree [185]. We then develop a check for whether a repeated-eigenvalue solution is an optimal one, for general graphs.

Let us begin with the tree case.

Theorem 7.2. *Consider the optimal edge design problem for a connected tree graph (i.e., for the case that the designable edge set E specifies a connected tree). In this case, the unique optimal edge design satisfies one of the following three conditions:*

1) the optimal algebraic connectivity is non-repeated, the optimized eigenvector $\mathbf{x}^*$ has no zero components, $|x_i^* - x_j^*|$ is identical for each $\{i, j\} \in E$, and the optimized eigenvector's components are strictly increasing along the path from any vertex s to any vertex t in the designable edge graph such that $x_s^* < 0$ and $x_t^* > 0$.

2) the optimal algebraic connectivity is non-repeated, the optimized eigenvector $\mathbf{x}^*$ has a single zero component at a vertex of degree 2 in the designable edge graph, $|x_i^* - x_j^*|$ is identical for each $\{i, j\} \in E$, and the optimized eigenvector's components are strictly increasing along the path from any vertex s to any vertex t in the designable edge graph such that $x_s^* \leq 0$ and $x_t^* > 0$.

3) the optimal algebraic connectivity is repeated z times. Also, all (eigen)vectors

$$\mathbf{x}^* \in Span(\mathbf{x}^*(1), \cdots, \mathbf{x}^*(z))$$

have a zero component at a particular vertex i with degree $z+1$. Further, for a path from vertex i to any vertex t, the sequences $x_i^*, \cdots, x_t^*$ is monotonic and the differences of the eigenvector components across each edge in the path are identical.

We refer the reader to [185] for the proof.

Next, we give a condition for checking whether or not a repeated-eigenvalue solution is optimal.

Theorem 7.3. *Consider an edge weight assignment with $\sum_{\{i,j\} \in E} k_{ij} = \Gamma$, which has algebraic connectivity repeated z times, and coresponding eigenvectors $\mathbf{x}^*(1), \cdots, \mathbf{x}^*(z)$. This assignment* *is an optimal edge design, if and only if the following condition holds: for any set of numbers Δ_{ij}, $\{i,j\} \in E$ such that $\sum_{\{i,j\}\in E} \Delta_{ij} = 0$ and $\Delta_{ij} \geq 0$ if $k_{ij} = 0$, there exists $\mathbf{x}^* \neq 0 \in$ $Span(\mathbf{x}^*(1), \cdots, \mathbf{x}^*(z))$ such that $\sum_{\{i,j\}\in E} \Delta_{ij}(x_i^* - x_j^*)^2 \leq 0$.*

Proof: An edge-design is optimal if and only if any permissible (small) perturbation of the design does not increase the Fiedler eigenvalue. Thus, let us compare the Fiedler eigenvalue of L (the Laplacian when the edge weights k_{ij} are considered) with the Fiedler eigenvalue of $\widehat{L}$ (the Laplacian when edge weights $k_{ij}+\Delta_{ij}$ are considered). We note that $\widehat{L}$ can be written as $L+\delta L$, where $||\delta L||$ is small when the perturbation Δ_{ij} is assumed small. To obtain conditions for optimality, we need to determine the sensitivity of the Fiedler eigenvalue to the edge-weight perturbations Δ_{ij}, here in the case that the Fiedler eigenvalue is repeated. To do so, we apply an existing result of the sensitivity of repeated eigenvalues, that fundamentally is based on obtaining the sensitivities through solution of a lower-order eigen-problem [191]. Specifically, say that the matrix L has Fiedler eigenvalue $\lambda_1(L)$ repeated m times, with an eigenspace whose basis is specified by the columns of the $n \times m$ matrix V. Now consider the eigenvalue of $\lambda_1(\widehat{L})$. The result [191] indicates that the change in the m repeated Fiedler eigenvalues of L is approximated to the first order by the m eigenvalues of $V^T \delta L V$. That is, $\widehat{L}$ has m eigenvalues that are given by $\lambda_1(L)$ plus each of the m eigenvalues of $V^T \delta L V$ plus a further $\mathcal{O}(||\delta L||^2)$ error. Thus, further invoking convexity, we obtain that the graph design k_{ij} is optimal, if $V^T \delta L V$ has at least one non-positive eigenvalue for every perturbation δL corresponding to a permissible perturbation of the edge weights. With just a little algebra.

We note that Theorem 7.3 gives a full structural characterization of the optimal design, albeit in an implicit form. We also note that Theorem 7.3 reduces to Theorem 7.1 when $z = 1$, i.e. the algebraic connectivity is non-repeated.

7.4 An Explicit Design for Tree Graphs

The structural characterizations of the optimized eigenvector(s) presented in Section 7.3 immediately permit us to develop finite-dimensional search algorithms for finding the optimal edge design. For tree graphs, it turns out that we can obtain the optimal design exactly with $\mathcal{O}(n^2)$ operations. In this section, we give the algorithm for finding the optimal edge design for tree graphs, present some simple qualitative insights into the pattern of the optimal edge weights, and show how the design can be updated upon addition of a new vertex in $\mathcal{O}(n)$ time. The algorithm and further results are obtained naturally from the structural characterization of the eigenvector given in Section 7.3 together with the eigenvector equation, and hence we present the algorithm here without proof.

We note that the article [185] already has obtained the optimal edge weights for tree graphs, with the results phrased in terms of the *variance* of the graph (with the motivation the the designable edge graph's structure is thus connected to the optimal design). Our algorithm for the optimum is an alternative to the one of Fiedler, that to some extent facilitates distributed computation of the design and leads to the further presented results.

Before presenting the algorithm, we find it useful to define some terminology. First, consider an edge $\{i, j\}$ in the tree. We refer to the two partitions formed upon removal of the edge as the **partitions induced by edge** $\{i, j\}$, and use the notation $S_i(\{i, j\})$ and $S_j(\{i, j\})$ for the partition including i and the partition including j, respectively. Similarly, we refer to the partitions formed upon removal of a vertex i in the tree as the **vertex-partitions induced by the vertex** i. We use the notation $S_1(i), \ldots, S_m(i)$ for the (in general m) partitions formed. Finally, we use the notation $D(i, j)$ for the distance (number of edges) between vertex i and vertex j.

We are now ready to present the algorithm. The algorithm has two steps. The first concerns finding the **critical edge** or **critical vertex**, i.e. the edge such that the optimized eigenvector has components of different signs at the two ends, or else the vertex for which the eigenvector component is null. The second step is the computation of the optimal design

Algorithm:

Step 1: Finding the Critical Edge or Vertex. Search through the edges in the graph, until a critical edge or vertex is found. In particular, for each edge $\{i,j\} \in E$, find $C(\{i,j\}) = \frac{1}{n}(\sum_{k \in S_j(\{i,j\})} D(i,k) - \sum_{k \in S_i(\{i,j\})} D(i,k))$. If $0 < C(\{i,j\}) < 1$, then $\{i,j\}$ is the critical edge. If none of the edges is critical, find $C(\{i\}) = \frac{1}{n}\sum_{k=1,\cdots,n} D^2(i,k)$. Then the vertex i for which $C(\{i\})$ is minimized is the critical one (and in fact $C(\{i\}) \leq C(\{j\})$-1 for all j in this case). We notice that finding the distances and hence $C(\{i,j\})$ or $C(\{i\})$ requires $\mathcal{O}(n)$ additions/multiplications, and so Step 1 has maximum computational complexity $\mathcal{O}(n^2)$.

Step 2: Finding the Optimal Edge Design. Let us consider two cases, namely the case where we have a critical edge and that where we have a critical vertex.

Critical-Edge Case: Say that the edge $\{i,j\}$ is the critical one. Then construct the n-component vector $\widehat{\mathbf{x}}$ that has kth entry given by $D(i,k) - C(\{i,j\})$. Then $\mathbf{x}^* = \frac{\widehat{\mathbf{x}}}{||\widehat{\mathbf{x}}||_2}$ is the optimized eigenvector (normalized to unit length). Also, the optimal algebraic connectivity is $\lambda^* = \Gamma(x_i^* - x_j^*)^2$. Finally, the optimal edge weights can be found recursively, as follows. For edges $\{a,b\}$ such that a is a leaf, $k^*_{a,b} = \frac{\lambda^* x_a^*}{x_a^* - x_b^*}$; these optimal edge weights can immediately be calculated. Once they have been calculated, notice that there exists at least one non-leaf edge $\{a,b\}$ whose optimal weights has not been computed, and for which all the optimal edge weights in the partition $S_a(\{a,b\})$ have been computed. For this edge, $k^*_{a,b}$ can be computed as $k^*_{a,b} = \frac{\lambda^* x^*_a - \sum_{q\in\mathcal{N}(a)} k^*_{q,a}(x^*_q - x^*_a)}{x^*_a - x^*_b}$, where $\mathcal{N}(a)$ contains the neighbors of vertex a except for b. After computing this optimal edge weight, we see that again one of the edges whose optimal weight remains to be computed has an associated partition for which all optimal weights have been computed; hence, recursively, all the optimal edge weights can be computed. It is easy to check that this computation is $\mathcal{O}(n)$.

Critical Vertex Case: Say that the vertex i is the critical one. Construct a vector $\mathbf{y}$ with kth entry given by $D(i,k)$. For each edge $\{a,b\}$ in the graph, find the **scaled weight** $\widehat{k}_{a,b}$ of the edge. Do this as follows: first, for each edge $\{a,b\}$ such that a is a leaf of the tree, find $\widehat{k}_{a,b} = \frac{y_a}{y_a - y_b}$. After the weights for the leaves have been found, notice that in each vertex-partition induced by i, at least one of the edges whose weight remains to be found has associated (edge) partition which 1) is within the vertex partition and 2) has all scaled weights determined. For any such edge $\{a,b\}$ (with a connected to the partition with known weights), the scaled weight is computed as $\widehat{k}_{a,b} = \frac{y_a - \sum_{q\in\mathcal{N}(a)} \widehat{k}_{q,a}(y_q - y_a)}{y_a - y_b}$. In this way, scaled weights can recursively be computed for all edges. Next, the optimal edge weight for each edge $\{a,b\}$ can be computed as $k^*_{a,b} = \frac{\Gamma \widehat{k}_{a,b}}{T}$, where $T = \sum_{\{a,b\}\in E} \widehat{k}_{a,b}$. Also, the optimal eigenvalue is given by $\lambda^* = \frac{\Gamma}{T}$. Finally, the optimized eigenvectors form a vector space of dimension $m-1$, where m is the number of vertex-partitions induced by i. In particular, any vector with kth entry given by $c_j y_k$, where j is the induced partition which contains vertex k, that has zero sum is an optimized eigenvector. It is easy to check that this computation is $\mathcal{O}(n)$. $\square$

We note that the computation of the critical edge or vertex is identical to the one given in [185]. Meanwhile, the edge-weight computation that we use contrasts from that in [185], in that individual weights are found recursively from neighbors' weights. We notice that this approach make clear that only a limited set of global information (e.g. the critical edge location) need be transmitted to permit local computation of the optimal weights.

The algorithm is computationally $\mathcal{O}(n^2)$. We notice that when we add new vertices to an existing tree, we do not need to recalculate the critical edge or vertex from scratch. Specifically, when a single node is added to an existing tree, Step 1 can be simplified to have only constant computational time. This simplification is made possible by the fact that upon tree expansion, the critical edge/vertex $\{i, j\}$ moves in a special pattern. We show this result in Theorem 7.4. For convenience, let us denote the tree constructed from tree $T = T(V, E)$ by connecting new vertex q to vertex $p \in V$ as $\tilde{T} = T(\tilde{V}, \tilde{E})$, where $\tilde{V} = \{V, q\}$ and $\tilde{E} = \{V, \{p, q\}\}$.

Theorem 7.4. *Consider a tree graph $T = T(V, E)$ and say that a vertex q is added to the graph through connecting to vertex $p \in V$. The critical edge or vertex of tree $\tilde{T}$ can be determined as follows:*

1) Suppose tree T has a critical edge $\{i, j\}$, and $q \in S_i(\{i, j\})$, then tree $\tilde{T}$ either has a critical edge $\{i, j\}$ or has a critical vertex i.

2) Suppose tree T has a critical vertex i, and $q \in S_k(i)$, then tree $\tilde{T}$ either has a critical vertex i or has a critical edge $\{i, j\}$, where $j \in S_k(i)$ and $\{i, j\} \in E$.

Proof: We prove this theorem using the concept of **absolute center** proposed in [185], which is defined as the point M that minimizes $S(M) = \sum_{k \in v} d^2(M, k)$. Clearly, the absolute center is either the critical vertex or lies on the critical edge. In order to prove the theorem, we only need to show that with the addition of a vertex, the following two facts hold: 1) the absolute center can not move in the opposite direction aginst where the extra vertex is added; and 2) the absolute canter can not move across the closest vertex.

The first statement can be easily proved using a simple contradiction argument, directly based on the definition of the absolute center. Now let us prove the second statement. Let us consider the case that T has a critical vertex i, and the distance between the vertex q and i is l_q+1. Clearly, we have $l_q \leq \frac{n-1}{2}$. We could see easily that for the tree $\tilde{T}$, we have $S(j)-S(i) \geq n+l_q^2-(l_q+1)^2 = n-1-2l_q \geq 0$, since for the origial tree, $S(j)-S(i) \geq n$ holds according to Theorem 3.3 in [185]. Hence the absolute center of $\tilde{T}$ can not cross j. Combining the first and the second statements, we see that $\tilde{T}$ either has a critical vertex i or has a critical edge $\{i,j\}$, where $j \in S_k(i)$ and $\{i,j\} \in E$. When T has a critical edge, the proof is similar and hence is omitted here.

The theorem informs that by adding one node to an existing tree, the critical vertex/edge of the expanded tree can be easily found. The critical vertex/edge in the expanded tree either stays the same, or moves along the edge in the direction of the added node without crossing the nearest vertex. The precise location of the critical vertex/edge can be obtained in constant time through modifying $C(\{i,j\})$ of tree T.

We also note that the edge weights in the optimal solution have some interesting dependence on the visual graph structure. We present these simple results in Theorem 7.5 and Theorem 7.6.

Theorem 7.5. *Consider any path from the critical edge/vertex to a leaf in a connected tree graph. The optimal edge weights along the path decrease monotonically.*

Theorem 7.6. *Consider the set of edges that connect to a leaf in a connected tree graph. The magnitudes of optimal edge weights in the set are ordered according to, and in fact are linear proportional to, the corresponding leaves' distances to the critical edge/vertex.*

For a connected tree graph, the above two theorems give us necessary conditions for an edge weight design to be the optimum. In the circumstance that the optimal edge weights are hard to obtain, we can resort to the theorems to obtain a solution that is close to the optimum through designing the edge weights, by walking from the leaves to the critical vertex/edge.

7.5 Some Structural Results for Bipartite and General Graphs

From the structural results in Section 7.3, we also can obtain a finite-search algorithm for finding the optimal design, in the case where the optimal algebraic connectivity is not repeated. Briefly, the finite-dimensional algorithm works as follows: for each possible cutset of the graph, it turns out that one can assign a unique *potential optimal eigenvector* such that the cutset separates eigenvector components with different signs. In turn, an edge weight assignment that achieves this eigenvector can be identified, if one exists, and the optimality of the potential solution can be checked. A direct implementation of such a finite-search algorithm (which is deeply related to our algorithms for scaling design, see [1,5,29]) is computationally intensive as compared to the standard numerical methods, and so we omit the details.

A more important consequence of our direct approach to the Laplacian edge design problem is that it yields significant structural insight into the optimal design for general (non-tree) graphs. Here, we summarize some interesting insights into the optimal designs for more general classes of graphs. In particular, we 1) characterize the edge-utilization structure and aspects of the eigen-structure of the optimal, focusing especially on the observation that many optimal edge assignments are present and on the consequences of this observation; 2) lower-bound the minimum eigenvalue in terms of graph properties and use this to improve existing bounds for design performance; and 3) show how the direct approach can inform, and be meshed with, LMI-based approaches.

In presenting results for non-tree designable edge sets, we shall often find it convenient to distinguish between bipartite and non-bipartite graphs. Let us thus recall that a **bipartite graph** is one in which the vertices can be divided into two sets, such that every edge connects a vertex in one set with a vertex in the other. We shall also find it convenient at times to classify graphs in terms of whether they admit an optimal design without a repeated algebraic connectivity.

7.5.1 Edge-Utilization Structure and Eigenstructure

Let us begin with analysis of the edge-utilization structure and eigen-structure for the optimal. A critical observation that underlies the results in this subsection is that the Laplacian edge design problem (almost) always admits multiple optimal edge designs. Let us formalize this concept, in the following theorem:

Theorem 7.7. *Consider a Laplacian edge-design problem that has at least one optimal design with non-repeated algebraic connectivity. The optimal edge design for the problem is unique if and only if the designable edge set forms a tree graph.*

Proof: This result can be proved straightforwardly by using the eigenvalue/eigenvector equation at the optimum together with perturbation arguments. Suppose we start with a non-tree optimal design with non-repeated algebraic connectivity. We can always modify the edge weights by small values to obtain another optimal design, while maintaining the same total edge weight. This process is possible because of the freedom in design given by the extra edges together with the special eigenvector structure at the optimum. Hence this optimal design is not unique. The only unique design occurs when the designable edge set forms a tree, see our earlier results and [185].

This result can be proved straightforwardly by using the eigenvalue/eigenvector equation at the optimum together with perturbation arguments. Thus, we see that for non-tree designable edge sets, the Laplacian edge-design problem admits a family of optimal edge-weight selections. The method of proof shows that, in fact, multiple edge designs exist even though the optimizing eigenvector may be unique.

Observing that many optimal edge designs are possible, we may be motivated to seek for designs that have particular edge-utilization characteristics. The next result clarifies that, at least for bipartite graphs, optimal designs can be obtained that have the identical structural properties as the graph of designable edges, e.g. identical minimum cutsets, node-degree properties, etc. Designs with these characteristics may be preferable e.g. because of their desirable fault-tolerance properties.

Theorem 7.8. *Consider a Laplacian edge-design problem with designable edge set that forms a bipartite graph. If there is at least one optimal design for which the algebraic connectivity is non-repeated, then there is an optimal design for which all the designable edges are assigned non-zero weights.*

Proof: Consider any design for which the Fiedler eigenvalue is non-repeated. The designable edge graph has a spanning tree with non-zero edge weights. This observation, together with the eigenvector-component-difference result given in [185] and the fact that the graph is bipartite, yields that the difference in the components of optimized eigenvector $\mathbf{v}^*$ across each edge in the designable edge graph is identical. Now consider incrementing each edge weight that is not in the mentioned spanning tree by a small amount ϵ. Let us next re-solve for the edge weights in the spanning tree so that λ^* remains an eigenvalue, and the corresponding eigenvector is $\mathbf{v}^*$. From the fact that the differences in eigenvector components are identical, we automatically recover that the sum of the edge weights in the new solution is also Γ. For sufficiently small ϵ, we see from perturbation notions that all the edge weights are strictly positive, and further that λ^* remains the Fiedler eigenvalue. Thus, we recover that there is an optimal design that uses all edges.

Using a very similar argument, one can also obtain a more restricted result on edge utilization for general graphs:

Theorem 7.9. *Consider a Laplacian edge-design problem. If there is at least one optimal design for which the algebraic connectivity is non-repeated, then there is an optimal design that 1) is bipartite and 2) would become non-bipartite if any other edge from the designable edge set E were made non-zero.*

Also, the locations of the eigenvalues in the optimal solution—and in particular the possibility for repeated algebraic connectivity—is important because it informs on e.g. the sign patterns of eigenvector components (and hence associated dynamics) and impacts use of numerical tools for design.

In the following theorem, we characterize the presence of solutions to the Laplacian edge design problem with repeated eigenvalues:

Theorem 7.10. *Each Laplacian edge design problem either has an optimal edge design such that the non-zero edge weight set E_d forms a tree, or has an optimal edge design such that the algebraic connectivity is repeated.*

Proof: We prove this theorem using a purturbation argument. Suppose we start with a non-tree optimal design with non-repeated algebriac connectivity λ^*. Now let us iteratively reduce the weight of a edge that is not in a spanning tree, while modifing the other edge weights to maintain one eigenvalue at λ^*. This process either leads to an optimal tree solution, or to a solution that λ^* is not the algebraic connectivity, e.g., λ^* is the third dominant eigenvalue. Clearly for the second case, in the middle of the iteration process, there exists a design has repreated algebraic connectivity.

This theorem, which also is proved by studying the family of possible designs achieving the optimum, clarifies the very common presence of repeated-eigenvalue optima. This common occurrence of repeated-eigenvalue solutions is important from the perspective of using numerical optimization tools (such as SDP-based methods, see e.g. [80]): at feasible solutions with repeated eigenvalues, the derivative of the Lagrangian with respect to the design parameters becomes ill-defined; the fact that the optima themselves have such a structure suggests that appropriate regularization may be needed in using the numerical techniques.

In concluding the discussion on the edge-utilization structure and eigenstructure of optimal designs, we recall that the optimizing eigenvector has a very special structure, in the case where the algebraic connectivity is nonrepeated, see also [185]. In particular, as we stated in positing the finite-search algorithm, we can uniquely determine the optimizing eigenvector (obviously, to within a scaling factor) once the cutset that separates positive-valued and negative-valued components in the eigenvector is known. This conclusion is a natural generalization of the eigenvector construction for trees (see Theorem 7.2), but does not provide low-order algorithms for finding the optimum, and so we omit the details. What is interesting is that this insight into eigenvector structure (or, alternatively, the check for optimality given in [185]) immediately yields a conclusion about whether or not an the optimal edge design can be a tree:

Theorem 7.11. *Consider a Laplacian edge-design problem, where the minimum edge- and vertex-cutsets of the designable edge graph are both at least* 2*. Then the non-zero edge set for any optimal edge design does not form a tree.*

Proof: Let us prove the result by contradiction. Assume that the optimal edge design was a tree. Then there is either a vertex whose corresponding optimized Fiedler eigenvector component is nil, or an edge for which the optimized eigenvector components are of different signs at its two ends. Consider the edge case (which is in fact the non-repeated-eigenvalue case). Let us view this edge as separating two partitions of the tree. From the condition of the theorem, we know that there is at least one designable edge connecting vertices in the two partitions. Finally, from monotonicity of eigenvector components, we know that the difference in eigenvector components across this edge is more than the difference between two vertices in the tree, hence the design cannot be optimal. A very similar argument holds in the nil-vertex case, with the observation that there are edges between the multiple partitions formed being used.

We notice that this result gives a more precise characterization of the edge-utilization structure, in particular one where the structures of *all* optimal designs are characterized, for a broad class of designable edge graphs.

7.5.2 Graph-Theoretic Bounds on Performance

Our solution methodology for the Laplacian edge-design problem also permits development of graph-theoretic bounds on the optimal algebraic connectivity. Such bounds lend insight into the design methodology, because they permit comparison of the optimal design with uniform-weight designs and allow evaluation of performance for particular graph classes (e.g., regular meshes, small-world networks, etc). Here, as an example, we introduce a lower-bound on the optimal algebraic connectivity, and discuss application of this bound.

Lower bounding the algebraic connectivity is often of particular interest, because such a bound implies an upper bound on the *settling time* of an associated dynamics, e.g., a distributed agreement protocol in a sensor network or a *formation* dynamics in an autonomous-vehicle team (e.g., [25,141]). Here, we are interested in lower-bounding the algebraic connectivity, in the case where the optimal edge design is used. We can simply bound the algebraic connectivity in terms of the diameter of graph specified by the designable edge-set, as follows:

Theorem 7.12. *Consider a Laplacian edge design problem that has resource bound* Γ*. Then the optimal Fiedler eigenvalue is lower-bounded by* $\frac{4\Gamma}{nD^2}$*, where* D *is the diameter of the designable edge graph (i.e. of the unweighted graph in* n*- vertices with edge set* E*).*

The proof of this lower-bound follows from considering a modified Laplacian edge design problem for a spanning tree of the original designable graph, and using the eigenvector-structure result given in Theorem 7.2. We omit the details. A particularly interesting instance of the above theorem is in the case where the upper bound Γ is set equal to the number of vertices n; this instance permits us to study the scaling of the optimal eigenvalue with respect to the number of vertices in the designable graph, when the total available resource (Γ) is proportional to the number of vertices. In particular, we find that optimal Fiedler eigenvalue is lower-bounded by $\frac{4}{D^2}$ in this case. Thus, we see that the optimal eigenvalue remains relatively large as the number of vertices in the graph increases, as long as the diameter remains small. This result immediately implies that the optimal edge design for small-world graphs, which have small diameter (see [157, 192, 193]), achieve fast settling.

It is worth noting that an even tighter lower bound, phrased in terms of the variance of spanning trees of the graph, follows immediately from [185]. However, application of the variance-based result is more complex, because of the difficulty both in its computation and in choosing the appropriate spanning tree.

7.5.3 Meshing our Design with Numerical Methods

Our direct methodology for addressing the Laplacian edge-design problem can also inform existing numerical solution techniques. To review, Boyd and co-workers have used *semi-definite programming* (SDP) techniques to solve the Laplacian edge-design problem and other similar design problems (e.g., [80]). In our complementary work on network scaling problems [29], we have also noted that simple gradient descent-type algorithms can be used to obtain optima. These numerical methods typically yield fast solutions to the design problems, which complement the structural insights that can be obtained through our direct approach. Here, let us briefly discuss one way in which our approach can inform, and be meshed with, the numerical solution strategies.

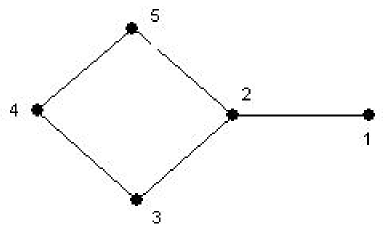
In particular, we stress again that our methodology shows that many optimal designs can be obtained for non-tree graphs, not only the single optimum provided by the numerical methods. Thus, our direct methodology can naturally be interfaced with the numerical techniques to identify a family of optimal edge designs, as follows:

1) The numerical technique can be used to obtain with high fidelity one optimal edge design and corresponding optimal algebraic connectivity and optimized eigenvector. We notice that our direct analysis explicitly specifies the optimized eigenvector once the cutset separating vertices with positive- and negative- valued components is identified; thus, the exact optimized eigenvector can be obtained after a finite number of iterations of the numerical algorithm.

2) Once the optimized eigenvector $\mathbf{x}^*$ and eigenvalue λ^* have been obtained, the eigenvector equation can be used to find the space of edge designs for which $\mathbf{x}^*$ and λ^* are an eigenvector and eigenvalue, respectively. We notice that only some of these designs have λ^* as the algebraic connectivity (rather than as one of the higher eigenvalues); it is straightforward to check whether λ^* is the optimal eigenvalue from the eigen-analysis.

Let us illustrate the method for finding a family of designs in an example:

We consider the Laplacian edge design problem for the 5-vertex designable edge graph shown in Figure 7.1, where the upper bound is assumed to be $\Gamma = 52$. From the above theorems,

a) 

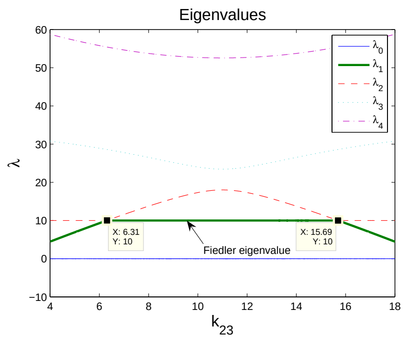
b) 

Fig. 7.1: Example demonstrating multiple optimal edge designs: a) 5-vertex designable edge graph, b) Laplacian eigenvalues over the range of the optimal designs

we recover that the vector $\mathbf{x} = [\frac{16}{10}, \frac{6}{10}, \frac{-4}{10}, \frac{-14}{10}, \frac{-4}{10}]'$ is the optimal eigenvector (if there is an optimum with non-repeated eigenvalue), and that the corresponding optimal eigenvalue is 10. Eigenanalysis can then be used to find the necessary conditions on the edge weights at the optimum:

$$\begin{cases} k_{12}^* = 16 \\ k_{23}^* + k_{25}^* = 22 \\ k_{23}^* - k_{34}^* = 4 \\ k_{34}^* + k_{45}^* = 14 \end{cases} \tag{7.1}$$

We note that the family of solution to these necessary-condition equations is parameterized by a single edge weight. All these solutions have the $\mathbf{x}$ as an eigenvector, with corresponding eigenvalue at 10. However, the eigenvalue is the algebraic connectivity for only some of the designs. In particular, from Figure 1, which shows the dependence of the all the eigenvalues of the design on the edge weight between vertices 2 and 3, we see that the optimum is reach whenever this edge weight is between 6.31 and 15.69.

7.6 Illustrative Example: A Well-Designed Flow Network

In numerous domains, items/materials transport through a network toward an equilibrium state—for instance, water may flow among reservoirs until their depths balance; or a vehicular transportation network may spread goods, new ideas, or (on a less positive note) viruses through a community; or a network of biochemical interactions may allow units in the brain to match their sleep states (e.g., [1, 4, 5]). In these various applications, the *settling time* of the network—i.e., the time required for the network to reach its equilibrium state—is often of interest. When the settling time is to be designed (or has been designed through an evolutionary process) subject to transportation resource constraints, a network design problem must be addressed.

A canonical model for the evolution of nodal quantities in a flow network toward equilibrium is described by the differential equation $\dot{\mathbf{x}} = -L\mathbf{x}$, where $\mathbf{x}$ lists the evolving quantities at the nodes, and L is a Laplacian matrix associated with a specified graph. In the case where we can allocate flow resources between connected nodes subject to a total resource limit, we recover the Laplacian edge design problem. As an illustration of our design methodology, let us address the Laplacian edge design problem for a canonical flow network with tree graph structure.

In particular, we consider design for the 40-vertex tree shown in Figure 7.2, subject to a resource bound $\Gamma = 50$. The egde marked "max" turns out to be the critical edge, and hence the algebraic connectivity is non-repeated. The optimal edge-weight allocation is illustrated in the figure, with the thickness of a line representing the edge weight. we find that the the optimal resource allocation yields a settling dynamics governed by the (negative of) the optimized eigenvalue $\lambda^* = 0.1086$, as compared to a governing eigenvalue of 0.0423 for a uniform resource allocation of 50 units.

Although the above the design could also be achieved through numerical methods, we stress that our design methodology informs us on the structure of the optimal resource allocation and

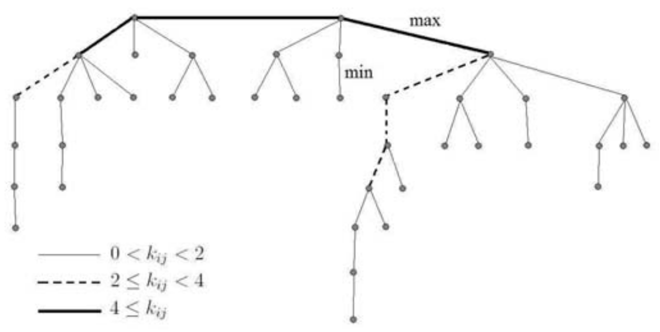


Fig. 7.2: Optimal edge design for a 40-vertex tree. The edges with maximum and minimum weight are shown, and the weights are delineated with varying line thicknesses.

associated dynamics in several ways. Here are some observations that are illustrated by the flow-network example:

- Our structural design methodology provides information on the distribution (pattern) of the optimal edge weights in the tree. In particular, we see from Theorem 7.5 that the largest weights are close to the critical edge/vertex, and specifically fall on the the longest paths to the critical edge/vertex. Meanwhile, the weak edge weights are at or near the tree's leaves, with the strength proportional to the leaves' distance to the critical edge/vertex, e.g., the weakest is on the shortest paths to the critical edge/vertex as observed in the example (see Figure 2). We again stress that the characterization of edge weight parameters permits us to obtain good designs even when the precise construction of the optimum is infeasible.

- Our design method clarifies the structure of the optimized eigenvector associated with the algebraic connectivity. Specifically, we know that the eigenvector components associated with each graph vertex are structured as follows: they decrease/increase by equal intervals along each path from the critical edge or vertex. That is, the eigenvector has a linear shape along each graph edge. This characterization gives insight into the initial states in the flow network that result in slowest settling, when the optimal design is used. In particular, let us compare the settling to equilibrium for different initial nodal quantities (that are normalized to unit spread). Among such initial quantities, the one in which nodal quantities grow linearly along each branch away from the critical edge settle most slowly[†]. Such an initial condition can be visualized for the particular given example.

- We can lower-bound the performance of the optimal design. Specifically, noting that the example network has diameter $D = 13$, we immediately find from Theorem 7.12 that $\lambda^* \geq 0.0296$.

[†] Notice that the vector is non-unique in the repeated-eigenvalue case.

8. ON TIME-SCALE DESIGNS FOR NETWORKS

We motivate the problem of designing a subset of the edge weights in a graph, to shape the spectrum of an associated linear time-invariant dynamics. We address a canonical design problem of this form by applying time-scale assignment methods, and give graph-theoretic characterizations of the designed dynamics.

8.1 Introduction

Controller design for the purpose of time-scale assignment is a cornerstone of classical and modern control theory. In such controllers, high and/or low gains of various scales are used to assign the eigenvalues of a linear time-invariant (LTI) plant along one or more asymptotic time scales, e.g. [139, 194–196]. Time-scale assignment has proved critical for a family of stabilization and performance-design tasks. In particular, multiple time-scales are fundamentally needed for disturbance rejection (e.g. [197]), and further permit systematic solution of such varied problems as stabilization/regulation under actuator saturation (e.g. [198]), loop transfer recovery (e.g. [196]), and decentralized controller design [10], among others. The ability to assign eigenvalues along desired time-scales is fundamentally related to the *linear-system structure*, i.e. to the zero dynamics and infinite-zero structure of the plant. As designs for large-scale systems and networks are increasingly needed, however, it is becoming more and more important that time-scale assignment capabilities be related to the *topological* (graph) structure of the system. To clarify this connection, the zero- and infinite-zero- structure—and hence the time-scale design properties—must be characterized in terms of the topological structure.

In a complementary direction, the common presence of multiple time-scale dynamics in existing large-scale infrastructure networks has been explained, and the time-scale structure *has* been related to the topological structure of the network (e.g., [138,199]). This characterization—which originated in the electric power systems community under the heading of **slow coherency** [138] and was further generalized through the definition of **synchrony** [199]—is based on the premise that large-scale networks naturally have groups of components that are strongly connected to each other but only weakly tied to the remainder of the network. The special topological structure yields 1) slow dynamics that are global but coherent or synchronic within each tightly-connected group, and 2) fast dynamics that are localized to individual groups. This recognition of the typical time-scale structure of networks is valuable for a family of infrastructure-network analyses, including for model reduction and partitioning (e.g., [138, 199, 200]). However, the graph structure-based time-scale characterizations are only for existing networks, and the idea of *designing* desirable time scales by exploiting the graph structure has not been addressed. Such design is of significant interest, because it can permit shaping of the network dynamics, including specifically the modification of existing coherency structures.

The purpose of this this work is to marry the efforts on time-scale assignment with the graph-structural characterization of time-scales in large-scale network analysis. That is, we motivate the problem of *designing time-scales in large scale systems by exploiting their topological structure*, and in turn initiate research in this direction by addressing a canonical design problem of wide interest. Precisely, we identify several controller design and graph-edge design problems in networks, for which time-scale designs that exploit the network's topology are needed. The problems that we identify originate from such diverse fields as virus-spreading control, drug design, traffic flow management, and sensor networking. We then fully address the time-scale design for an example problem motivated by these applications, namely that of designing some edge weights in a graph (while others remain fixed) to shape a dynamics defined by the associated **Laplacian** matrix (see [73] for background on the Laplacian matrix and its spectrum). Specifically, viewing this *partial graph design* problem as a (decentralized) controller design problem, we characterize the infinite-zero structure and finite-zero dynamics of the plant in terms of the topologies of the fixed and designable graph edges. In turn, we propose a high-gain methodology for the partial graph design, and characterize the spectrum upon design in terms of the graph topologies. We thus tie the performance of time scale-based designs to the topological structure, and (in a complementary direction) characterize the network dynamics over a range of edge-weight values.

In that we are obtaining designs for networks that exploit their graph structure, our efforts here also contribute to the nascent research on high-performance network design [1, 5, 10, 29, 80]. These recent studies are focused on designing network controllers or connections (edges) to optimize dynamic measures, typically using optimization machinery together with algebraic graph-theory notions. These design problems are complex, and only the simplest cases have been addressed—for example, designing all the transition probabilities in a Markov chain to achieve fast mixing, or selecting static controllers for a network of autonomous agents with single-integrator dynamics [29, 80]. Our efforts here expose that time-scale designs can be used for a much wider class of design problems, in particular ones where the network has a complex existing structure and only some local features (whether edge properties or local controllers) can be altered. For these partial design problems, the dynamics from the point of the designable features exhibit a rich structure that is deeply tied to the existing network topology; a graph-based approach to time-scale design is critical for shaping these dynamics.

The remainder of the chapter is organized as follows. In Section 8.2, we motivate partial graph design problems in several applications. In Section 8.3, we address a canonical partial graph design problem, and also present an example illustrating our design.

8.2 Motivation

We motivate the partial graph design problem, and hence the time-scale assignment methodology that exploits topological structure, from several application domains.

8.2.1 Design in Autonomous Vehicle Coordination and Sensor Networking

In recent years, the development of distributed algorithms/controllers for autonomous vehicle coordination and sensor networking applications has been of wide interest in the controls community (e.g., [19, 25]). Network controller/algorithms have been developed for myriad tasks (including formation, agreement, and distributed partitioning), but these various tools have in common that they permit simple and highly-limited agents to coordinate by exploiting network interactions. Although many network tasks have been studied, however, design of *high-performance* controllers/algorithms (that achieve fast settling, and robustness to disturbances and variations) is in its early stages (e.g. [29, 80]). These few efforts have focused on designing the entire communication/observation network, or else local controllers at all the network nodes, to optimize a scalar performance measure (e.g., a settling rate or condition number) [1, 5, 29]. Building on these works, we have recently demonstrated that pole placement can be achieved using a multiple-delay and multiple-derivative control scheme at all network nodes [10].

In many cases, only partial design of the network interactions and controllers is possible. For instance, only a few nodes in a large-scale mobile sensor network may be amenable to modification, due to limitations in resources or access. Similarly, sensor networks that are operated by multiple players perhaps only can be updated in parts, only newly-added sensors/vehicles in a network may be amenable to modification, or only certain communication links may admit higher bandwidth/fidelity. In these cases, design of a subset of the network edges (specifically, protocol strengths or weights) and nodes (specifically, controller gains) is of critical interest. This is precisely the partial graph design problem.

In a complementary direction, we often need to characterize the dynamics of autonomous-vehicle teams or sensor-network algorithms upon perturbation of network parameters, due to e.g. communication failures or environmental variation. Such characterizations also require us to study network dynamics as edge weights or controller gains are varied, and so partial graph design informs the analysis.

A wide variety of agent, network, and controller models are used in sensor networking and autonomous vehicle control applications, and so a range of partial graph design problems can be posed. Here, let us abstractly introduce only one canonical design problem, which for instance is representative of consensus-algorithm and velocity-coordination design, e.g. [25,29]. Specifically, let us consider the dynamics $\dot{\mathbf{x}} = -L\mathbf{x}$, where $\mathbf{x}$ represents the agents' states (e.g., velocities, opinions), and L is the Laplacian matrix associated with a weighted graph Γ that has vertex set V and edge set E, and weight $k_{ij} > 0$ for each $\{i, j\} \in E$. Let the edge set E consist of a subset E_f with edges having fixed weights and a subset $E_d = E - E_f$ with edges whose weights can be designed. The design problem of interest is to select the weights k_{ij} for $\{i, j\} \in E_d$, so as to shape the dynamics $\dot{\mathbf{x}} = -L\mathbf{x}$ (in particular, to decrease the settling time of the dynamic response while limiting the impact of initial-condition- and external- disturbances).

8.2.2 Epidemic Spread Control

Epidemic spread is a typical large-scale network application that demonstrates strong topological structure-dependent and time-scale behavior (e.g. [1, 49]). Since viruses are transmitted through the close contact of susceptibles with infectives, the distribution of the **contact rates** (and hence **transmission rates**) among a population [49]—or in the other words, the population structure—crucially impacts epidemic spread. Moreover, slow coherency is apparent since a population is naturally composed of rather highly connected groups (that usually coincide with geopolitical regions) with much lower contact rate in between.

Several control strategies (e.g., vaccination, fast hospitalization, and restriction on traveling) function to reduce the transmission rates among/within regions. By heterogeneously distributing these control resources (e.g., reducing the transmission rates by different amounts across the network) to exploit the topological structures, we can efficiently reduce the spread of virus with limited control resources [1]. In our previous work [1], we developed an optimal heterogeneous resource allocation by taking into account the topological structure. One specific control problem that we explored in that chapter was to optimally scale the outflow transmission rate of each region so as to minimize the virus spreading rate, subject to resource constraints.

In designing optimal resource allocations, an important practical concern is that the nominal transmission rates in some parts of a network are costly or impossible to change. For instance, we may reduce the transmission rate across regions by checking travelers' health conditions and quarantining the ones with susceptible symptoms; however some regions may not have such medical equipment. Also, reducing traffic flows between some regions is infeasible because of the close economic ties between them. In these circumstances, the transmission rates between certain regions are fixed while others are free to design. This motivates the partial graph design problem.

As a simple example, we consider the multi-group Susceptible-Infected-Susceptible (SIS) epidemic model [1]. At early stages of spread, the dynamics of the number of infectives in each region i can be described by

$$\begin{bmatrix} \dot{I}_1 \\ \dot{I}_2 \\ \vdots \\ \dot{I}_N \end{bmatrix} = \begin{bmatrix} \beta_{11}N_1 - v_1 & \beta_{21}N_1 & ... & \beta_{n1}N_1 \\ \beta_{12}N_2 & \beta_{22}N_2 - v_2 & ... & \beta_{n2}N_2 \\ \vdots & \vdots & \cdots & \vdots \\ \beta_{1n}N_2 & \beta_{2n}N_n & ... & \beta_{nn}N_n - v_n \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_N \end{bmatrix}$$

, where I_i is the number of infectives in region i, N_i is its population, v_i is the local recovery rate, and β_{ij} is the transmission rate from region i to region j. For edges $\{i, j\}$ in a set E_f,the weights β_{ij} are fixed. A typical design problem is to design β_{ij} for $(i, j) \in E - E_f$ so as to minimize the dominant eigenvalue of the system matrix, perhaps subject to constraints on the total resource, (e.g., $\sum \beta_{ij} \geq \Gamma$) and individual constraints on the designable transmission rates. Beyond designing the dominant eigenvalue, spectral assignment for the purpose of shaping the spread response and disturbance rejection may be needed.

8.2.3 Multi-Target Drug Design

One task in drug design is to modify targets (e.g., compounds, reactions, or enzymes) in a microbiological network, to affect the network in a desired way (e.g., stop the production of some compounds while minimizing the interference to the remaining compounds in the network) [201]. Multi-target drug design is believed to be more favorable to single-target design in that it efficiently provides more reliable dynamics [202]. This motivates study of a partial graph edge design problem, where the designable edges represent the interactions that are potential targets. Let us give an example formulation of the drug targeting design problem as a partial graph design problem.

Let us consider a simplified model $\dot{\mathbf{x}} = \sigma(A\mathbf{x})$ of a *gene regulatory network*, where x_i denotes the the activity of gene i, A_{ij} denotes the strength of interaction from gene i to gene j (see [203]), and the standard saturation function $\sigma(\cdot)$ represents that the activity of each gene is constrained.

Also we denote the set of interactions that are potential targets as E_d. In terms of controlling the asymptotic behavior, the optimal edge design problem can for example be formulated as: designing the interaction strengths for edges $\{i, j\} \in E_d$ such that 1) the system model is stable in the sense of Lyapunov, and 2) a quadratic function $\sum_{i=1}^{n} \epsilon_i(\bar{x}_i - x_i^*)^2$ is minimized, where $\bar{x}_i$ is the asymptotic value of x_i, x_i^* is the desired asymptotic activity of gene i, and ϵ_i is a weighting factor. When transient dynamics are also considered, the optimal edge design problem can for example be formulated as: designing interaction strengths for edges $\{i, j\} \in E_d$ such that a quadratic function $\sum_{i=1}^{n} \epsilon_i(\bar{x}_i - x_i^*)^2 + \sum_{i=1}^{n} g_i \int_{t=t_0}^{to+T} (x_i(t) - x_i^*)^2$ is minimized, where T is a settling time, and g_i is also a weighting factor. We note that these design problems also fundamentally require spectrum assignment through edge-weight design.

Other than the selective control problems like drug target design, evaluation tasks such as: 1) predicting dynamics with partial knowledge of the network and 2) inversely detecting certain pathways based on the exhibited system behavior, are important in biological network applications. These problems arise commonly because of the difficulty in detecting/quantifying all reaction pathways experimentally. In these cases, characterizing dynamics over the range of possible interaction strengths is important. These complementary problems also thus benefit from the analysis of partial graph edge design.

8.2.4 *Changing Coherency Topologies in Electric Power Networks*

The slow-coherency theory has long been used in electric power system analysis, for such varied tasks as model reduction, partitioning, and identification of critical failures (e.g., [138]). Recently, DeMarco and coworkers have argued that identification of coherency groups (partitions) is also critical for evaluation of generator market power, because such groups are constrained in active-power transfer with the remaining network and so the generators within them achieve similar *locational marginal prices* [204].

Broadly, coherency of generators in the electric power system is defined in terms of the generator's rotor angles. In particular, coherent groups are typically determined from a linearized model of the swing dynamics around a nominal power-flow (steady-state) solution. The linearized dynamics turn out to be defined by the Laplacian of a graph, where the edge weights in the graph are related to the line susceptances and the nominal angle differences between the buses. The coherency groups are identified from this Laplacian.

We contend that a variety of evaluation and design tasks can be informed through characterization of linearized swing dynamics over a range of graph edge weights. Most prominently, changes in the operating point of the model will result in changes in weights in the linearized model, and hence will affect the dynamics and coherency structure of the model. For instance, increasing load demand in parts of the system may force lines toward their active-power limits even if they have large susceptance, and hence change the coherency structure (and in turn market power, etc). Similarly, line susceptances may vary due to environmental conditions, or may be uncertain. Thus, it is valuable to evaluate the network's performance—and more specifically, the spectrum associated with the linearized dynamics—as some of the edge weights in the associated graph change. This evaluation over a range of possible graph edge weights is the partial graph design problem studied here. In a complementary direction, critical transmission problems (e.g., hyper-sensitivity to line failures, or the presence of generators with extreme market power) and inclusion of new generation may require construction of new lines. Design of further transmission can similarly be viewed as a partial graph design problem. We note that partial graph design for the purpose of disturbance rejection is of particular interest in the power system application.

8.3 A Canonical Partial Graph Design Problem

While the partial network design problems introduced in Section 8.2 vary in their specifics, the fundamental design aim is common: we seek to design the strengths of some interconnections in a network, or else decentralized controllers at some network nodes, so as to shape the network's dynamics. Furthermore, we aim to obtain designs and design characterizations that are phrased in terms of the graph structure of the network.

To this end, we here pursue a canonical partial network design problem, specifically the problem of designing a subset of the edges' weights in a graph to shape the dynamics defined from an associated Laplacian matrix. We introduce the design problem in Section 8.3.1. We then reformulate the problem as a decentralized controller design problem (Section 8.3.2). Based on the reformulation, we take two steps to address the partial network design problem: 1) we relate the linear system structure (finite- and infinite-zero structure) of the open-loop plant in the analogous control problem to network's topology, in particular a *fixed-edge graph* (i.e., comprising edges that cannot be designed) and a *designable-edge graph* (comprising edges whose weights can be selected) in the original problem (Section 8.3.3); 2) by bringing to bear the time-scale-based design methodology [195, 196], we address the partial network design problem from the controller design viewpoint (Section 8.3.4) and so characterize the closed-loop spectrum. In this way, we both obtain and characterize designs in terms of the network's topology. Finally, we give an example (Section 8.3.5).

8.3.1 Formulation

We focus on designing the weights of a subset of the edges in a graph, to shape the spectrum of an associated Laplacian matrix and hence to shape dynamics defined thereof. This design problem for Laplacians is directly applicable to two of the applications in Section 8.2, namely the sensor networking and electric power applications. We stress, however, that the methodologies for this particular design problem naturally can be adapted to the various other controller and network-interconnection design problems posed in the introduction *including one with asymmetric topologies.*

Precisely, we consider a weighted and undirected graph Γ with n vertices, labeled $1, \dots, n$. We specify the edges in the graph through two disjoint sets each containing pairs of distinct vertices, which we term the **fixed edge set** E_f and the **designable edge set** E_d. Specifically, for each pair of vertices $\{i, j\} \in E_f$, the graph Γ has an edge between vertex i and j with fixed weight $k_{ij} > 0$. Meanwhile, for each pair $\{i, j\} \in E_d$, the graph has an edge between vertex i and vertex j with weight k_{ij} that can be set to a desired nonnegative value. For pairs $\{i, j\}$ that are neither in E_d or E_f, we shall say that there is not an edge between vertex i and vertex j, and for convenience we set the weight k_{ij} to 0. We also find it convenient to label and order the edges in E_d with the positive integers $1, \dots, |E_d|$, and refer to the weight of the edge $m \in \{1, \dots, |E_d|\}$ as k_m.

We aim to design the edge weights k_{ij} for $\{i, j\} \in E_d$, so as to shape a dynamics defined from the weighted Laplacian matrix associated with the graph Γ (e.g. [73]). Let us recall that the Laplacian matrix associated with the graph Γ, which we denote as $L(\Gamma)$, is defined as follows: $[L(\Gamma)]$ is an $n \times n$ matrix with entries given by $[L(\Gamma)]_{ij} = [L(\Gamma)]_{ji} = -k_{ij}$ for all $i \neq j$, and $[L(\Gamma)]_{ii} = -\sum_{j \neq i} [L(\Gamma)]_{ij}$ for all i. Our goal is to design the edge weights $k_{ij} \in E_d$, to shape the spectrum of $L(\Gamma)$ (i.e., to assign its eigenvalues and eigenvectors), or equivalently to shape the dynamics of such differential equations as $\dot{\mathbf{x}} = L\mathbf{x}$, $\dot{\mathbf{x}} = -L\mathbf{x}$, or $\ddot{\mathbf{x}} = -L\mathbf{x}$ (see the motivation in Section 8.2).

We refer to the above design problem in its entirety as the **partial graph design problem**. For convenience, we refer to the graph Γ in the case where the designable edge weights are set to zero as the **fixed-edge graph**, and use the notation Γ_F for it. We use the notation $L(\Gamma_F)$ for the corresponding Laplacian. We also form a **designable-edge graph** Γ_D by removing the fixed edges from Γ; we define the Laplacian matrix $L(\Gamma_D)$ for the designable graph in the standard way.

8.3.2 Reformulation as a Decentralized Controller Design

Our aim is to set the designable edge weights in graph Γ to shape the dynamics of $\dot{\mathbf{x}} = L(\Gamma)\mathbf{x}$, or in other words to assign the spectrum of $L(\Gamma)$. Here, we show that the problem of shaping the dynamics can be reformulated as a linear static decentralized controller design problem. Through designing the gains of the decentralized controller, we in turn are able to assign the spectrum of $L(\Gamma)$.

To present the reformulation, we find it convenient to use the notation $\mathbf{q}_i^j$ for the n-component vector with ith entry equal to 1, jth entry equal to -1, and remaining entries null. In this notation, the Laplacian $L(\Gamma)$ can be rewritten as

$$L(\Gamma) = \sum_{\{i,j\}\in E} k_{ij}\mathbf{q}_i^j{\mathbf{q}_i^j}^T = \sum_{\{i,j\}\in E_f} k_{ij}\mathbf{q}_i^j{\mathbf{q}_i^j}^T + \sum_{\{i,j\}\in E_d} k_{ij}\mathbf{q}_i^j{\mathbf{q}_i^j}^T.$$

To clarify that the design of $L(\Gamma)$ is a decentralized controller design problem, let us define the following matrices:

- We let $A = \left[\sum_{\{i,j\}\in E_f} k_{ij}\mathbf{q}_i^j{\mathbf{q}_i^j}^T\right]$.
- For each edge $m = 1, \ldots, |E_d|$ in the designable edge set, let B_m equal q_i^j, where $\{i, j\}$ are the two ends of the edge. Also, we let $C_m = B_m^T$.
- Recall that we denote the weight of edge m in the designable-edge graph by k_m.

In this notation, we immediately recover the following expression for $L(\Gamma)$:

$$L(\Gamma) = A + \sum_{m=1}^{|E_d|} B_m k_m C_m$$

. Noting that the weights k_m are the parameters in $L(\Gamma)$ amenable to design, we see that the partial graph design problem is the following decentralized controller design problem: a decentralized LTI plant with state matrix A has $|E_d|$ channels, where channel m has observation matrix C_m and actuation matrix B_m; static nonnegative linear feedback gains k_m for all m such that $m \in \{1, ..., |E_d|\}$ must be developed at the $|E_d|$ channels, to shape the dynamics $\dot{\mathbf{x}} = L(\Gamma)\mathbf{x}$, or equivalently the spectrum of $L(\Gamma)$.

For convenience, let us stack each channel's observations and inputs into matrices, as follows: $B \triangleq \begin{bmatrix} B_1 & \dots & B_{|E_d|} \end{bmatrix}$, and $C \triangleq \begin{bmatrix} C_1 \\ \vdots \\ C_{|E_d|} \end{bmatrix}$. In this notation, the decentralized system's closed-loop dynamics can be written as $\dot{\mathbf{x}} = A\mathbf{x} + BKC\mathbf{x}$, where K is an $|E_d| \times |E_d|$ diagonal matrix whose diagonal entries are the designable edge weights. Thus, we see that the partial graph design problem can be viewed as an $|E_d|$ channel static decentralized controller design for the plant (C, A, B), for the purpose of spectrum assignment. In the remainder of this section, we obtain and characterize solutions to this design problem, that are based on the topology of the fixed- and designable- edge graphs.

8.3.3 Topological Characterization of the Plant Dynamics

Time-scale assignment for LTI plants requires characterization of the *linear-system-structure*, i.e. the infinite-zero- and finite-zero dynamics of the plant. The **special coordinate basis** (SCB) for linear systems provides a representation of the linear system structure that particularly facilitates time-scale design [205]. Thus, here we obtain the SCB for the plant model formulated in Section 8.3.2, as a step toward time-scale assignment through partial graph design. We note that the SCB, and the linear system structure that it captures, were developed for centralized control; however, this work as well as our recent efforts [10] indicate that the SCB facilitates decentralized controller design also.

The special structure of the partial graph design problem permits us to characterize the linear system structure of the plant (C, A, B) in terms of the graph topology. We begin with a preliminary remark on the plant's open-loop poles:

Remark: The open-loop poles of the plant (C, A, B) are the eigenvalues of the matrix $L(\Gamma_F)$.

Next, we present a sequence of results that together specify the finite- and infinite-zero dynamics of the plant (C, A, B) in terms of the fixed- and designable- edge graph topologies. For convenience, we do so (Theorems 8.1, 8.2, 8.3) in the case where the designable graph Γ_D is is a **z-forest**, i.e. a collection of z **trees** or connected acyclic graphs. From the perspective of obtaining the linear system structure, we can limit ourselves to the case where the designable graph is acyclic WLOG, since cyclic designable graphs yield redundant observation and input, i.e. the matrices B and C are not full rank. In particular, one can always view the control as using a subset of observations and inputs and hence define the finite- and infinite- zero dynamics thereof, while the redundant observations and inputs are simply considered unused. Thus, the results that we obtain for the linear-system structure for the z-forest case trivially translate to the general case, and so we focus on the z-forest case for notational simplicity. It is worth making one further (and rather intricate) observation, however: redundancy in B and C is of no value to design in the centralized setting, hence the centralized infinite-zero and finite-zero structure decomposition always ignores this case. Interestingly, however, the use of non-tree designs fundamentally affects other aspects of performance in the decentralized setting. With this contrast in mind, we give a brief discussion of the cyclic designable graph design in Section 8.3.4. As an aside, we note that acyclic designable graphs are typical in many of the applications of our theory, for instance for design of a single interaction (edge), or for design of the interconnections between newly added nodes and existing ones.

In characterizing the finite- and infinite-zero dynamics, let us first specify the dimensions of each, and in the process clarify that the plant has a **uniform-rank** structure (see e.g. [206]):

Theorem 8.1. *When Γ_D is a z-forest with $|E_d|$ edges in total, the plant (C, A, B) is square-invertible and uniform-rank-*1*, with finite- invariant zero dynamics of dimension $n - |E_d|$.*

Proof: Notice that the first Markov parameter CB is an $|E_d| \times |E_d|$ matrix of full rank. Thus, we immediately recover that the plant is uniform-rank-1 and square invertible. Since the plant has $|E_d|$ inputs and outputs, the total dimension of the infinite-zero dynamics is $|E_d|$, and so the finite invariant zero dynamics has dimension $n - |E_d|$.

The uniform-rank-1 structure of the plant immediately permits us, through simple transformation of the standard representation for uniform-rank systems (see e.g. [196] for details), to phrase the dynamics of the plant (C, A, B) as follows:

$$\dot{\mathbf{y}} = P_1\mathbf{y} + C_a\mathbf{x}_a + Q\mathbf{u} \tag{8.1}$$

$$\dot{\mathbf{x}}_a = A_a\mathbf{x}_a + B_a\mathbf{y},$$

where $\mathbf{x}_a \in R^{n-m}$ represents the state of the plant's finite-invariant zero dynamics, and the matrices P_1, C_a, Q, A_a, and B_a are obtained through the state transformation. Next, we focus on characterizing the parameters of the SCB representation in terms of the fixed- and designable- edge graphs' topologies, to permit high-gain design and design characterization in terms of the graph topology.

Our characterization of the infinite-zero and finite-zero dynamics is in two steps: first (Theorem 8.2), we define a state vector for the finite-zero dynamics, and so specify the finite- and infinite-zero dynamics formally in terms of the plant model (C, A, B) (but in a form that facilitates connection with the graph topology). Next (Theorem 8.3), we give an explicit graph-theoretic construction of the state matrix associated with the finite-zero dynamics. Before presenting these results, we require some further notations[*]:

—We call x_i the state variable **associated** with vertex i in the graph.

—We find it convenient to partition the vertices based on the graph Γ_D. In particular, we partition the vertex set in such a way that two vertices are in the same partition if and only if there is a path between them in Γ_D. Notice that the groups of vertices that form connected subgraphs in Γ_D, as well as the remaining isolated vertices, are the partitions. In total, there are $n - |E_d|$ partitions, which we label $S_1, \dots, S_{n-|E_d|}$.

—We define the **superstate** $\overline{x}_i$ associated with each partition $i = 1, \dots, n - |E_d|$, as $\overline{x}_i = \frac{1}{|S_i|}\sum_{j \in S_i} x_i$. We notice that a vector containing the super-states can be computed as a linear combination of the state vector $\mathbf{x}$, say as $\widehat{C}\mathbf{x}$.

We are now ready to specify a state for the zero dynamics, and hence obtain the SCB representation formally in terms of the plant model.

Theorem 8.2. *The superstates $\overline{x}_i$, $i = 1, \dots, n - |E_d|$, together form a state for the zero dynamics*

[*] We note that the following definitions are for arbitrary designable-edge graphs, not only forests.

of the plant. In these coordinates, the SCB representation of the plant is

$$\dot{\mathbf{y}} = CA \begin{bmatrix} C \\ \widehat{C} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix} + CB\mathbf{u} \tag{8.2}$$

$$\dot{\mathbf{x}}_a = \widehat{C}A \begin{bmatrix} C \\ \widehat{C} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix}.$$

Proof:

Since the plant is uniform-rank-1 with $|E_d|$ outputs and $n - |E_d|$ zeros, the plant dynamics can be expressed through state transformation with the two equations $\dot{\mathbf{y}} = P_1\mathbf{y} + C_a\mathbf{x}_a + Q\mathbf{u}$ and $\dot{\mathbf{x}}_a = A_a\mathbf{x}_a + B_a\mathbf{y}$, where $\mathbf{x}_a \in R^{n-|E_d|}$ represents the state of the plant's zero dynamics. We claim that $\mathbf{x}_a = \begin{bmatrix} \overline{x}_1 \\ \vdots \\ \overline{x}_{|E_d|} \end{bmatrix}$ serves as a state for the zero dynamics.

To prove the claim, we need to show that the plant dynamics can be written in the form (8.1) when $\mathbf{x}_a$ is defined as above. To do this, it is sufficient to show that the whole plant state $\mathbf{x}$ can be recovered from $\mathbf{x}_a$ and $\mathbf{y}$ (so that $\dot{\mathbf{y}}$ can be written in terms of $\mathbf{y}$, $\mathbf{x}_a$, and $\mathbf{u}$), and that $\dot{\mathbf{x}}_a$ does not depend on the input (and so can be written as a linear combination of $\mathbf{x}_a$ and $\mathbf{y}$).

To show that the whole state $\mathbf{x}$ can be recovered from $\mathbf{x}_a$ and $\mathbf{y}$, notice that $\begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix} = \begin{bmatrix} C \\ \widehat{C} \end{bmatrix} \mathbf{x}$, where $\widehat{C} \in R^{(n-m)\times(m)}$ computes $\mathbf{x}_a$ from $\mathbf{x}$ according to the definition of the superstate. We need only show that $\begin{bmatrix} C \\ \widehat{C} \end{bmatrix}$ has full rank to show that $\mathbf{x}$ can be recovered from $\mathbf{y}$ and $\mathbf{x}_a$. From the fact that Γ_D is a z-forest, we immediately see that the m rows of C are linearly independent. Now consider appending C sequentially with each row of $\widehat{C}$. The augmented row is clearly linearly independent to previously-added rows of $\widehat{C}$, since the non-zero entries of the new row are at different locations than those of the previously-added ones. Also, note that the newly-added row is linearly independent of the rows of C, since any linear combination of these rows has entries that sum to zero within each partition. Thus, all rows of $\begin{bmatrix} C \\ \widehat{C} \end{bmatrix}$ are linearly independent, and hence $\mathbf{x}$ can be recovered from $\begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix}$.

We also must show that $\dot{\mathbf{x}}_a$ can be written as only a linear combination of $\mathbf{y}$ and $\mathbf{x}_a$. To do so, notice that $\dot{\mathbf{x}}_a = \widehat{C}\dot{\mathbf{x}} = \widehat{C}(A\mathbf{x} + B\mathbf{u}) = \widehat{C}A\mathbf{x} + \widehat{C}B\mathbf{u}$. Noting that the two entries in each column of B corresponding to a partition sum to zero, we recover that $\dot{\mathbf{x}}_a = \widehat{C}\mathbf{x} = \widehat{C}A \begin{bmatrix} C \\ \widehat{C} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix}$. We have thus shown that $\mathbf{x}_a$ as defined in theorem statement specifies a state for the finite-zero dynamics.

The SCB form follows automatically: the expression for $\dot{\mathbf{x}}_a$ has already been found, while the expression for $\dot{\mathbf{y}}$ can be obtained from the original plant model together with the computation of $\mathbf{x}$ in terms of $\begin{bmatrix} \mathbf{y} \\ \mathbf{x}_a \end{bmatrix}$.

While Theorem 8.2 formally specifies the SCB of the plant, the representation is not explicitly connected to the fixed- and designable- graphs' topologies. In fact, all the parameter matrices in the SCB representation can be described explicitly in terms of the graph topologies. In the interest of space, we only characterize the state matrix of the zero dynamics, which is of critical importance in the high-gain design, in this way. We first require a bit further notation: Consider two distinct partitions S_i and S_j, where $i, j \in 1, \ldots, n - |E_d|$. We use the notation $\overline{k}(S_i, S_j)$ for the sum of the (fixed-graph) edge weights between partitions i and j, i.e. $\overline{k}(S_i, S_j) = \sum_{l \in S_i, m \in S_j} k_{lm}$. We refer to $\overline{k}(S_i, S_j)$ as the **aggregate weight** between partitions i and j.

We are now ready to present the structural result:

Theorem 8.3. *Consider the partial edge design problem when the designable-edge graph is a z-forest, and consider the state matrix of the zero dynamics A_a in the SCB representation (8.1). The entry at row i and column j of A_a is given by $\frac{\overline{k}(S_i,S_j)}{|S_i|}$, for $i \neq j$. The diagonal entries are the negative of the sum of the off-diagonal entries, i.e. they make the row sums zero.*

Proof:

We seek to simplify the expression for the zero dynamics given in Theorem 8.2 (Equation 8.2), so as to give an explicit graph-theoretic characterization of its state matrix. To this end, let us denote the matrix $\begin{bmatrix} C \\ \\ \widehat{C} \end{bmatrix}$ by $\begin{bmatrix} Z_1 & Z_2 \end{bmatrix}$, where Z_1 has dimension $n \times |E_d|$ and Z_2 has dimension $n \times (n - |E_d|)$. From Theorem 8.2, we see that the state matrix of the zero dynamics is $A_a = \widehat{C}AZ_2$.

Let us characterize the matrices $\widehat{C}$ and Z_2 in terms of the graph structure, and hence obtain a graph-theoretic characterization of A_a. To do this, notice first that row i of the matrix $\widehat{C}$ has entries as follows: the jth entry is equal to $\frac{1}{|S_i|}$ if $j \in S_i$, and is equal to zero otherwise. Now consider the matrix Z_2. From its definition, we see that $\begin{bmatrix} C \\ \\ \widehat{C} \end{bmatrix} Z_2 = \begin{bmatrix} \mathbf{0}_{|E_d| \times (n-|E_d|)} \\ \\ \mathbf{I}_{(n-|E_d|) \times (n-|E_d|)} \end{bmatrix}$. We claim from this expression that Z_2 has the following form: the jth entry in column i is equal to 1 if $j \in S_i$, and is equal to zero otherwise. To check this, notice that each row in C sums to zero and has non-zero entries within a single partition. Thus, we obtain that $CZ_2 = 0$ for Z_2 as defined. Meanwhile, we see that the entries in each row i of $\widehat{C}$ are contained within partition i and sum to 1, and so $\widehat{C}Z_2 = I$. From the characterization of C and $\widehat{C}$, and noticing that $A = L(\Gamma_F)$, we recover the result of the theorem with just a little algebra.

Let us make a couple observations about the structural result given in Theorem 8.3. First, we note that the state matrix of the zero dynamics is a Laplacian matrix with each row i inversely scaled by the number of vertices in partition i. More specifically, the zero dynamics is defined by a

Laplacian matrix of a graph with the aggregate weights specifying the edge values, together with a diagonal scaling matrix. Precisely, let us define the **zero graph**[†] Γ_Z as a weighted and undirected graph with $n - |E_d|$ vertices, with the edge between vertex i and vertex j having weight $\overline{k}(S_i, S_j)$. We refer to the Laplacian of the graph (defined in the standard way) as $L(\Gamma_Z)$. Further, let us define the **size-scaling matrix** D as an $|E_d| \times |E_d|$ diagonal matrix with entries given by $|S_i|$. In this notation, the state matrix of the zero dynamics is given by $D^{-1}L(\Gamma_Z)$. From this expression, we see that the plant (C, A, B) has at least one zero at the origin, with the number at the origin given by the number of components in the graph Γ_Z. It is easy to check that the zeros at the origin are both *input- and output- decoupling zeros*. Meanwhile, the remaining zeros are strictly positive, and are transmission zeros. Since the zeros are the eigenvalues of a scaled Laplacian matrix, algebraic graph theory tools can be brought to bear to characterize the zeros (e.g. [73]).

8.3.4 Time-Scale Design

A high-gain controller architecture permits systematic assignment of the closed-loop poles of an LTI plant along asymptotic time scales, see the literature on **asymptotic time-scale and eigenstructure assignment**, or ATEA, design [196]. Specifically, given a plant's infinite-zero structure (which is clarified by the SCB), one can specify a family of high-gain controllers that allow placement of certain eigenvalues at one or more desirable *fast* time scales, while the remaining *slow* eigenvalues approach the finite invariant zeros of the plant. These multiple time-scale designs are widely used, including for design of stabilizing controllers [195], almost-disturbance-decoupling [197], and (as we have recently clarified) decentralized controller design [10]. As a further refinement, high-and-low-gain methods can be used for e.g. stabilization under saturation [198].

[†] We again note that the definition is in force for any partial edge design problem, not only one with designable-edge graph that is a z-forest.

High-gain and high-and-low-gain methods are apt for the partial graph design problem. Specifically, in the various applications, large or small edge weights—which correspond to high- and low-gains in the controller design reformulation—are often naturally assigned: e.g., algorithm weights can be set to desired large or small values, or the rates of chemical reactions can be drastically changed in the context of drug design. For the complementary analysis tasks (e.g., characterization of settling rates over a range of uncertain weights), the design methods are also valuable because they specify the dynamics of the network over the range of possible parameter values and identify the extremes.

Here, we give a first set of results concerning high-gain approaches for the partial graph design problem[‡]. Our focus here is on designing the initial-condition response of the plant and in particular the closed-loop spectrum, and also characterizing these designs in terms of the graph topology. We present our results in two steps, first specifying the high-gain design and its spectrum, and second discussing the spectrum over the range of possible edge weights by using the results from the design. Finally, we mention several generalizations and connections.

Let us begin by specifying and characterizing the high-gain design. Because of the special uniform-rank-1 structure of the equivalent controller design problem (as indicated in the SCB representation of the plant), we immediately recover from the ATEA design literature that identically scaling up all decentralized controller gains (equivalently, all designable edge weights) achieves a two-time-scale design. Specifically, we apply the following parameterized design: for each edge $\{i, j\} \in E_d$, we choose the edge weight as $k_{ij} = \alpha\widehat{k_{ij}}$ where each **nominal edge weight** $\widehat{k_{ij}}$ can be any fixed positive value, and the parameter α $(0 < \alpha < \infty)$ provides an identical scaling to each weight. We use the notation $K(\alpha)$ for a family of edge designs of this form, and call these a

[‡] We note that the results in this section hold for arbitrary designable-edge graphs, not only forests.

high-gain edge design.

The following theorem specifies the two-time-scale structure resulting from use of a high-gain edge design:

Theorem 8.4. *The spectrum of $L(\Gamma)$ upon application of a high-gain edge design $K(\alpha)$ with arbitrary nominal edge weights is as follows: for $\alpha \to \infty$, $L(\Gamma)$ has 1) $|E_d|$ eigenvalues that approach $+\infty$, and in particular are within $\mathcal{O}(1)$ of the non-zero eigenvalues of $L(\Gamma_D)$; and 2) $n - |E_d|$ eigenvalues that approach (i.e., are within $\mathcal{O}(\frac{1}{\alpha})$ of) the $n - |E_d|$ eigenvalues of $D^{-1}L(\Gamma_Z)$.*

This theorem clarifies that *any* high-gain edge design drives $|E_d|$ (fast) eigenvalues of $L(\Gamma)$ arbitrarily far right in the complex plane, while moving the other (slow) eigenvalues toward those of the scaled zero graph (which are the in fact the zeros of the plant model (C, A, B)). The result follows immediately from consideration of the ATEA design methodology [196] together with the SCB representation from Section 8.3.3 and simple algebraic graph theory notions, and so we omit the proof.

When a high-gain edge-design is used, we can also characterize the eigenvectors of $L(\Gamma)$. Briefly, we find the following, for sufficiently large α. 1) The eigenvectors associated with the slow eigenvalues have (approximately) identical entries corresponding to vertices in the same partition; these entries are matched with the entries of the eigenvector of $D^{-1}L(\Gamma_Z)$. 2) The eigenvectors associated with the fast eigenvalues are each concentrated in the vertices corresponding to a single connected subgraph in Γ_D.

We stress that the above characterizations of the graph design hold for *any* high-gain edge design, i.e. regardless of the choice of the nominal edge weights. Let us briefly discuss how one can choose among possible nominal designs. In doing so, first let us note a fundamental difference between centralized and decentralized high-gain feedback: a centralized static output feedback can be used to place the $|E_d|$ fast eigenvalues at arbitrary locations, while in the decentralized setting the fast eigenvalues (i.e., the eigenvalues of $L(\Gamma_D)$) cannot be assigned arbitrarily. In fact, the problem of assigning the eigenvalues of $L(\Gamma_D)$ is that of designing *all* the edge weights in a specified graph to place the eigenvalues of the associated Laplacian at desirable locations, see the previous works [6, 80] for numerical/analytical solutions to this problem. It is worth noting that our capability for assigning the fast eigenvalues is highly dependent on whether or not Γ_D is a tree, demonstrating that the (decentralized) partial graph design problem depends intricately on the structure of Γ_D even though the linear system structure does not.

Next, we note that the high-gain design also provides insight into the Laplacian's spectrum, over a range of possible designable edge weights. Such insight is valuable for the complementary task of evaluating a network's dynamics, when some of the interconnection strengths are subject to variation. To this end, we show in the following theorem that the eigenvalues of $L(\Gamma)$ are bounded by those of the fixed-edge graph and zero-graph Laplacians.

Theorem 8.5. *Consider a partial graph design problem. Denote the eigenvalues of $L(\Gamma_F)$ as $0 = \widehat{\lambda_0} \leq \widehat{\lambda_1} \leq \ldots \leq \widehat{\lambda_{n-1}}$, denote the eigenvalues of $D^{-1}L(\Gamma_Z)$ as $0 = \overline{\lambda_0} \leq \overline{\lambda_1} \leq \ldots \leq \overline{\lambda_{|E_d|-1}}$, and denote the eigenvalues of $L(\Gamma)$ by $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}$. For any assignment of nonnegative edge weights in the designable graph, we have $\lambda_i \geq \widehat{\lambda_i}$, $i = 0, \ldots, n-1$ and $\lambda_i \leq \overline{\lambda_i}$, $i = 0, 1, \ldots, |E_d| - 1$.*

In other words, the ith eigenvalue of the graph Laplacian is between the eigenvalues of the fixed-edge graph's Laplacain and the scaled zero graph's Laplacian. The result follows automatically from Theorem 8.4 along with the property that a Laplacian's eigenvalues increase monotonically with the edge weights, so we omit the details.

We have thus developed and characterized the performance of high-gain partial-graph designs.

The fundamental contribution of this design methodology is two-fold: 1) it shows how a network's topology can be exploited to assign an associated dynamic's spectrum along time scales, and 2) it characterizes the performance of the design explicitly in terms of the network's graph topology.

Let us conclude our discussion of the high-gain design, by remarking on several generalizations and connections:

1) The canonical problem studied here provides a simple illustration of the time-scale design strategy, but the strategy applies to other problems. We especially stress that asymmetric partial graph designs can also be addressed.

2) High-gain design can break existing slow-coherency structure (time-scale separation) only when edges between two subnetworks that were initially weakly linked can be designed (and hence the nodes become part of the same partition in our terminology). Our methodology clarifies that such a design not only eliminates slow eigenvalues associated with the coherency structure, but also loses the disturbance-localization properties that result from coherency.

3) Analogous results can be obtained for low-gain designs.

8.3.5 An Illustrative Example

We study a 30-node graph (Figure 8.1). The fixed-edge graph in this graph (identified by the thin blue lines) has edge weights inversely proportional to the length of the line in the plot. The fixed-edge graph has three completely decoupled subgraphs (A, B,and C), and subgraph A itself is composed of two weakly-coupled subgraphs (A_1, A_2). The designable graph (marked by the thick red lines) combines A and B into a single graph, and reduces weak coupling between the subgraphs A_1 and A_2. We are concerned with assigning the spectrum of the graph's Laplacian.

Now let us apply the time-scale design. From Theorem 8.1, the equivalent plant for this example has a zero dynamics of dimension $R^{25\times 25}$. Twenty-three state variables in the zero dynamics are identical to the state variables of the open-loop plant, while the other two state variables are the averages of the state variables in its partition, e.g., one state variable is the average of the states of vertices 8, 19, 22 and 26. The state matrix of the zero dynamics can be easily characterized in terms of the graph topologies using Theorem 8.3.

The above structural decomposition provides us with insights into the spectrum of the Laplacian matrix upon high-gain design. The spectrum (or, equivalently, associated dynamics) obtained through modifying these edge weights is constrained by the inherent structure of the zero graph. In this example, as we scale up the weights in the designable edge graph from nominal values, 5 eigenvalues of $L(\Gamma)$ move towards ∞. The other 25 increase monotonically, and approach but can never surpass the zeros, which are in fact the eigenvalues of the scaled zero-graph's Laplacian. From the zero graph, we infer that: 1) one eigenvalue moves from zero to a non-zero value, 2) the original slow-coherent behavior is eliminated/reduced since the zero graph has no edge-cutset of size 1, and 3) many larger eigenvalues change little since they are specified by strongly-connected subgraphs in the fixed-edge graph that are also present in the zero graph. A plot of the 5 smallest non-zero eigenvalues of the Laplacian as we scale the strengths of the 5 designable edges verifies these observations (Figure 8.1).

8.4 Connections and Future Directions

The control of dynamical networks has been widely studied during the last few years, so it is worthwhile to briefly connect our results with the existing literature in retrospect. To position our work in this context, let us first note that a great majority of the recent network controls literature has focused on introducing/modeling various algorithmic and control tasks and associated

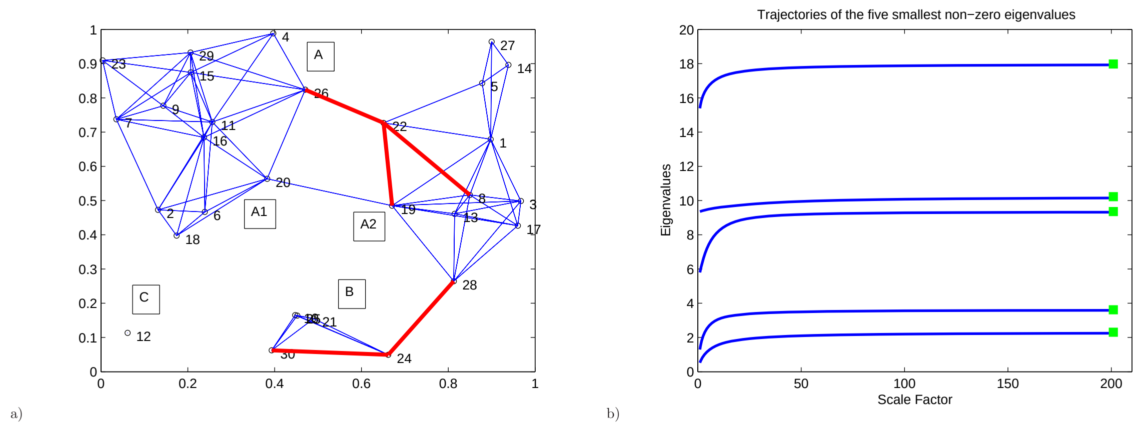


Fig. 8.1: a) 30-node graph: thin blue lines specify the fixed graph, while the thick red lines specify the designable graph. b) the 5 smallest non-zero eigenvalues of the graph Laplacian as we scale the strengths of the designable communication links.

algorithms/controllers (see [26, 141, 207, 208] and the references contained therein). In contrast, only very few of the recent works on network controls have addressed *controller and graph design*, i.e. specification of controller structure and gains, algorithm weights, or graph edge weights so that a network's dynamics meets performance criteria (see e.g. [1, 5, 6, 10, 29, 80]). Our perspective is that the design of network controllers and graphs is the principal outstanding task in the network controls arena, and so our focus has been in this direction.

The particular contributions made by this work are 1) solution of the important *partial* network design problem and 2) illustration of time-scale methods as a key tool for spectrum design. Both contribution significantly enhance the existing results on graph and network-controller design. Specifically, we note that the existing results on dynamical network design can be classified into graph-edge design problems [6, 80] and nodal controller/resource design problems [1, 29]. In the nodal controller design direction, our group has addressed several dominant eigenvalue or Fiedler eigenvalue *optimization* tasks, through design of static controllers (under constraint) at all nodes in the network. As a subsequent step, we have considered designing *dynamical* controllers for networks of double-integrator agents as well as hardwired networks, so as to achieve not only stabilization but effective placement of closed-loop eigenvalues [10]. In the graph-edge design direction, the article [80] and our work [6] have taken complementary (respectively, numerical and structural) approaches to the problem of designing *all* edge weights in a graph to *optimize* a Fiedler eigenvalue measure for an associated Laplacian matrix. However, in a wide range of network control applications (e.g., sensor networking, virus-spread control, or electric power systems ones), only some graph edges or controllers at some nodes can be designed. This chapter has initiated study of such partial graph/controller design problems, and so significantly enhances the scope of the graph-design efforts. Our study of partial graph design also elucidates that Fiedler eigenvalue optimization as a design goal is both unwieldy (due to non-convexity) and incomplete (since the entire spectrum may need design, and also since a family of good solutions rather than a single optimal one may be needed). What our time-scale design methodology provides is a systematic technique for obtaining a family of good partial graph designs, and for comparing them. Our results suggest that, in analogy with centralized controller designs, time-scale notions are fundamental to graph and network controller design.

We stress that the results presented here are only a beginning to research on partial graph design. The following are particular outstanding tasks that we plan to address in our future work:

1) The time-scale-based graph design introduced here directly applies even when the graph topology is non-symmetric and the matrix of interest is not a Laplacian. However, the eigenvalue majorization results obtained from the time-scale design (Theorem 8.5) are specific to the symmetric Laplacian case, since they depend on monotonicity properties of the Laplacian-matrix eigenvalues (and hence of the dynamical system's invariant zeros). We will seek to

characterize the system's invariant zeros for other common topology matrices (e.g, nonnegative matrices), and so further characterize the asymptotic design.

2) Although our focus here has been on eigenvalue assignment, the time-scale design approach naturally also permits us to shape the eigenvectors of a graph's Laplacian. A thorough study of eigenvector assignment using time-scale notions requires a full graph-theoretic characterization of the closed-loop zero dynamics (not only its state matrix). We will pursue this characterization in our future work.

3) Multiple time-scales are fundamentally needed for disturbance rejection [197], and so our design approach is promising for achieving external stability (disturbance rejection) in network applications. We expect to explicitly study the external stability of the models introduced here, using a time-scale approach.

9. MAJORIZATIONS FOR THE DOMINANT EIGENVECTOR OF A NONNEGATIVE MATRIX

Motivated by network controller design applications, we develop several majorization results for the dominant eigenvector of an irreducible nonnegative matrix.

9.1 Introduction

Recent efforts on network control and design have made clear that graph-theoretic characterization of *eigenvector* components is critical, for both undirected and directed graphs (which correspond to symmetric and asymmetric topology matrices). However, graph-theoretic characterization of eigenvalues and especially eigenvectors of asymmetric matrices is very limited. Here, we study the structure of the eigenvector associated with the dominant eigenvalue, for the broad class of irreducible nonnegative matrices (see e.g. [74]). More precisely, we characterize the dependence of the dominant-eigenvector components on individual entries in the matrix (equivalently, edge weights in an associated graph), as well as on row scalings (Section 9.2). Motivated by network design tasks in particular, we also briefly study the dependence of dominant eigenvector components on simultaneous modifications in multiple rows (Section 9.3).

Our particular motivation for studying the dominant eigenvectors of positive matrices stems from our efforts in decentralized controller design [1, 29]. From another viewpoint, this work also contributes to the extensive research on positive dynamical systems and the associated nonnegative matrices (see e.g [74, 209]), by further characterizing the eigenvector associated with the dominant eigenvalue.

9.2 Majorizations for Single-Row Incrementations

We are concerned in this section with understanding how the dominant-eigenvector components of a nonnegative matrix depend on individual matrix entries, and on scalings of single rows in the matrix. In fact, we find it most convenient to study the dependence in the case that a single row of the matrix is incremented in an arbitrary fashion, and hence to obtain results for single-entry changes and row scalings as special cases.

Precisely, let us consider an $n \times n$ real irreducible nonnegative matrix $G \overset{\triangle}{=} [g_{ij}]$. We consider incrementing a single row of G, say (WLOG) the first row, by a vector $\mathbf{a}^T \overset{\triangle}{=} \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix}$, where each $a_i \geq 0$ and $\mathbf{a} \neq \mathbf{0}$. That is, we study $\widehat{G} = G + \mathbf{e}_1 \mathbf{a}^T$, where $\mathbf{e}_1$ is an 0–1 indicator vector with first entry equal to 1.

We notice that $\widehat{G}$ is also an irreducible nonnegative matrix. Thus, G and $\widehat{G}$ each has a real positive eigenvalue (denoted λ and $\widehat{\lambda}$, respectively) that is non-repeated and has magnitude at least as large as each of its other eigenvalues. The eigenvector $\mathbf{v}$ (respectively $\widehat{\mathbf{v}}$) associated with G ($\widehat{G}$) is strictly positive entrywise. For the purpose of this note, we refer to $\mathbf{v}$ ($\widehat{\mathbf{v}}$) as the **dominant eigenvector** of G ($\widehat{G}$). Also, we use the notation v_i (respectively, $\widehat{v}_i$) for the ith component of $\mathbf{v}$ (respectively $\widehat{\mathbf{v}}$).

Our purpose is to compare the components of the dominant eigenvectors $\mathbf{v}$ and $\widehat{\mathbf{v}}$. We find that incrementing the first row increases the first eigenvector component relative to each remaining component:

Theorem 9.1. Consider the dominant eigenvectors $\mathbf{v}$ and $\widehat{\mathbf{v}}$. Then $\frac{\widehat{v}_1}{\widehat{v}_j} > \frac{v_1}{v_j}$ and for $j = 2, \dots, n$.

We thus recover that, when the eigenvector is normalized to unit length, $\widehat{v_1} > v_1$.

The above theorem shows that, when a particular row of a positive matrix is incremented (in an arbitrary way), the associated component of the dominant eigenvector increases relative to the other components. This results automatically specializes to two cases of particular interest, namely 1) the incrementation of a single entry in the matrix (which corresponds to incrementing an edge weight in an associated graph), and 2) the scaling of a row in the matrix (which corresponds to scaling the "influence" of a node in the graph, e.g. by changing a controller gain):

Corollary 9.2. Consider incrementing the entry at row i and column j of an irreducible nonnegative matrix G. Then the ratio of the ith component of the dominant right eigenvector of G to each other component strictly increases. Similarly, the ratio of the jth component of the dominant left eigenvector to each other component strictly increases.

Corollary 9.3. Consider scaling the ith row of an irreducible nonnegative matrix G by a factor $\alpha > 1$ (respectively $0 < \alpha < 1$). Then the ratio of the ith component of the dominant right eigenvector of G to each other component strictly increases (respectively, strictly decreases).

The result for row-scaling of positive matrices also permits us characterize the eigenvectors of nonsingular irreducible M *matrices* (see [74]) upon row-scaling, using the fact that inverses of irreducible M matrices are nonnegative (in fact, strictly positive) matrices:

Corollary 9.4. Consider scaling the ith row of a nonsingular and irreducible M matrix by a constant $\alpha > 1$ (respectively, $\alpha < 1$), and consider the left eigenvector associated with the eigenvalue of minimum magnitude that is real. The ratio of the ith component of this eigenvector to each other component strictly decreases (respectively, increases) upon row-scaling.

Let us take a moment to briefly interpret the above results for a couple applications, to illustrate their use. In the interest of space, we shall only discuss these examples at a conceptual level.

1) Velocity-control problems in autonomous-vehicle-coordination applications can often be abstracted that of designing a diagonal gain matrix K so as to optimize the dynamics $\dot{\mathbf{x}} = -KH\mathbf{x}$, where H is an M-matrix (see e.g. [29]). The above analysis clarifies that increasing the gain for a particular vehicle not only speeds up the slow mode of the system, but reduces the excitation caused by the initial conditions of that vehicle (since the *left* eigenvector component is depressed).

2) In virus-spreading-control applications, reducing the flow of infectives from one region i to another j has the effect of reducing the impact of the infectives in region i on other regions and reducing the size of the infected population in j (in addition to slowing the spread of the infection in general).

9.3 Majorizations for Multiple-Row Incrementations

In decentralized controller design tasks, it turns out that understanding the dependence of eigenvector components upon scaling of multiple rows or incrementation of multiple diagonal entries is important [1, 29]. With these applications in mind, here we briefly characterize the dependence of dominant eigenvector components of nonnegative matrices on diagonal entries of the matrices (noting that similar results hold for other multi-row incrementations). We develop the majorization results in two steps: first, we consider *designing* incrementations of multiple diagonal entries to achieve certain ratios among the dominant eigenvector components. Second, we use this result to study arbitrary incrementations of multiple diagonal entries.

First, here is the design result:

Theorem 9.5. Consider an irreducible nonnegative matrix G, and say (WLOG) that we can increment the first $m < n$ diagonal entries by amounts $k_1, \ldots, k_m$, respectively, to obtain $\widehat{G}$. Then, for each $k_1 > 0$, we can find $k_2 > 0, \ldots, k_m > 0$ so that 1) $\frac{\widehat{v}_i}{\widehat{v}_j} = \frac{v_i}{v_j}$ for $i = 1, \ldots, m$, $j = 1, \ldots, m$, and 2) $\frac{\widehat{v}_i}{\widehat{v}_j} > \frac{v_i}{v_j}$ for $i = 1, \ldots, m$, $j = m+1, \ldots, n$. Furthermore, $k_2, \ldots, k_m$ increase monotonically with increasing k_1.

In words, this theorem states that there is a way to increment m diagonal entries of a nonnegative matrix (for any fixed increase in the first component) so that the corresponding m components of the dominant eigenvector remain the same to within a scaling, while the ratios of these components to the others increase.

Finally, the above design result yields a majorization of eigenvector components for arbitrary diagonal incrementations, as formalized in the following theorem:

Theorem 9.6. Consider an irreducible nonnegative matrix G, and say (WLOG) that we increment the first $m < n$ diagonal entries by amounts $k_1, \ldots, k_m$, respectively, to obtain $\widehat{G}$. Then, there is $i \in 1, \ldots, m$ such that $\frac{\widehat{v}_i}{\widehat{v}_j} > \frac{v_i}{v_j}$ for all $j = m+1, \ldots, n$.

This theorem states that, when multiple diagonal entries of an irreducible nonnegative matrix are incremented, at least one corresponding components in the dominant eigenvector becomes larger in relation to all the components corresponding to non-incremented entries. This majorization result can proved by iteratively applying the design result (Theorem 9.5), and at the final stage applying the result of Theorem 9.6 to obtain the majorization.

10. AN EXPLICIT FORMULA FOR DIFFERENCES BETWEEN LAPLACIAN-EIGENVECTOR COMPONENTS USING COALESCED GRAPHS

We obtain an explicit formula for the absolute difference between two eigenvector components for a weighted graph's Laplacian matrix, in terms of the the Laplacian's eigenvalues as well as the eigenvalues of matrices associated with certain coalesced graphs. We then briefly illustrate two uses of this formula, in analyzing graph modifications.

10.1 Introduction

The eigenvalues of the Laplacian matrix associated with a weighted graph have been extensively characterized in terms of topological features of the graph. Eigenvectors, as other important statistics of the Laplacian matrix, provide significant further information about graph properties and network dynamics defined on a graph. Hence, the characterizations of eigenvectors are needed for a range of decentralized controls and dynamical-network analysis/design applications, including e.g., network partitioning [210], synchronization design [211], and optimal network resource allocation [1]. Despite such need, graph theoretic studies of the Laplacian's eigenvectors are sparse (see [73, 212, 213] for some reviews of these literature), and do not provide exact general characterizations of eigenvector-component values in terms of graph constructs for arbitrary graphs. Of relevance to our work, Merris in [214] obtained some exact results regarding Laplacian eigenvalues/eigenvectors under certain special graph topology changes (including upon coalescing of some particular vertices in a graph, e.g. ones that have identical components in an eigenvector).

Motivated by problems in such areas as network identification, graph partitioning, and secure controller design, we have been seeking explicit characterizations of various Laplacian eigenvector characteristics (including of eigenvector component values and differences). In this technical communique, we obtain that certain important eigenvector characteristics can be computed explicitly in terms of the Laplacian's eigenvalues, as well as the eigenvalues of Laplacian-type matrices of graphs formed by coalescing vertices in the original graph. Specifically, we give explicit expressions for differences between the eigenvector components associated with two vertices (which are widely used in e.g. graph partitioning and controller design applications) in terms of these eigenvalues. We also show two preliminary uses of the result. First, we obtain bounds on Laplacian eigenvalues upon modification of the graph in terms of the original eigenvectors, and hence we better characterize the spectra of certain designed networks. Second, we characterize the dependences of eigenvalues on edge weights solely in terms of the eigenvalues of the original and coalesced graphs.

We stress that our characterization is in force for an arbitrary graph's Laplacian matrix, and for any unique eigenvector of the matrix. In this sense, our work exposes that, generally, eigenvectors of a Laplacian matrix are precisely specified by the matrix's eigenvalues together with the eigenvalues of particular coalesced graphs' Laplacians.

10.2 Main Result

Let us consider an undirected and weighted graph G with n vertices (labeled $1, \dots, n$), m edges (each comprising a pair of distinct vertices), and positive weight $k_{ij} = k_{ji}$ associated with each pair $\{i, j\}$ that is an edge. We define the symmetric $n \times n$ Laplacian matrix L of the graph in the standard way, i.e. as follows: the off-diagonal entry l_{ij} is set equal to $-k_{ij}$ if $\{i, j\}$ is an edge, and is set to 0 otherwise. Each diagonal entry is selected so that the rows of the matrix sum to 0, i.e. $l_{ii} = -\sum_{j=1, j\neq i}^{n} l_{ij}$.

We recall that the Laplacian matrix L is a symmetric positive semi-definite matrix, with the number of zero eigenvalues equal to the number of maximal connected subgraphs of G. From here on, let us assume that G is connected. In this case, L has a non-repeated zero eigenvalue with corresponding eigenvector equal to $\mathbf{1}$ (the vector of all ones). The remaining $n-1$ eigenvalues are strictly positive and simple, though they may be repeated. Our primary aim here is to characterize the corresponding eigenvectors (in particular, to compute differences between eigenvector components) in terms of the eigenvalues of L and of certain derived graphs' Laplacians. For ease of presentation, we will address the case where the eigenvector of interest corresponds to a non-repeated eigenvalue, but also briefly summarize the result for the more general case (where the eigenspaces associated with repeated eigenvalues need to be considered).

Our main result is phrased in terms of the eigenvalues of Laplacian-type matrices formed when vertices in the original graph are coalesced, and so we require several definitions regarding coalesced graphs and associated Laplacian-type matrices. To this end, let us consider the graphs formed when one pair of vertices in a graph G are coalesced into a single vertex. We use the notation $\tilde{G}(i,j)$ to represent the $n-1$ vertex graph formed through coalescing vertices i and j in G. Specifically, $n-2$ of the vertices in $\tilde{G}(i,j)$ represent the $n-2$ vertices other than i and j in G. For each distinct pair x and y of these vertices, we associate an edge in $\tilde{G}(i,j)$ with the pair if $\{x,y\}$ is an edge of G, and assume that the weight of the edge $\tilde{k}_{xy} = \tilde{k}_{yx}$ is the same as in G (i.e., equal to k_{xy}). Also, let us denote the remaining vertex in $\tilde{G}(i,j)$, which represents the aggregation of vertices i and j in G, as $\overline{ij}$. We associate an edge with vertices $\overline{ij}$ and x ($x \neq \overline{ij}$), if there is an edge between x and i or an edge between x and j in G. We let the weight of this edge equal k_{ix} (respectively k_{jx}) if G only has an edge between i and x (respectively, j and x), and set the weight of this edge to $k_{ix} + k_{jx}$ if both edges are present in G. We refer to the graph $\tilde{G}(i,j)$ as the **(i.j)-coalesced graph**.

Finally, we use the notation $L(G)$ for the Laplacian matrix associated with the original graph G, and use the notation $L(\tilde{G}(i,j))$ for the Laplacian matrix associated with the graph where i and j are coalesced. (For convenience, we assume that the coalesced vertex is the one of lowest ordinality in constructing the Laplacian matrix, so that the first row and column of the Laplacian correspond to the coalesced vertex. Also, we find it convenient to define a **scaled Laplacian matrix** $\overline{L}(\tilde{G}(i,j))$ for coalesced graphs: this matrix is formed from the Laplacian matrix by scaling the first row of $L(\tilde{G}(i,j))$, i.e. the row corresponding to the coalesced vertex, by $\frac{1}{2}$ (and leaving the remainder of the matrix $\tilde{G}(i,j)$ unchanged).

We are now ready to present the main result, which relates the differences between eigenvector components for a graph's Laplacian matrix to eigenvalues of this matrix and those of its coalesced graphs' Laplacians:

Theorem 10.1. Consider a graph G. Let us label the eigenvalues of the Laplacian matrix $L(G)$ as $\lambda_1, \ldots, \lambda_n$, where $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Consider any non-repeated and non-zero eigenvalue λ_q of $L(G)$, and denote the corresponding eigenvector as $\mathbf{v}_q$ (where $\mathbf{v}_q$ is normalized to unit length, i.e. $\mathbf{v}_q^T \mathbf{v}_q = 1$). The absolute difference between the ith and jth entries of $\mathbf{v}_q$ can be computed as

$$|v_{qi} - v_{qj}| = \sqrt{2 \frac{\prod_{z=2}^{n-1} (\mu_z - \lambda_q)}{\prod_{z=2, z \neq q}^{n} (\lambda_z - \lambda_q)}}, \tag{10.1}$$

where $0 = \mu_1 < \mu_2 \leq \ldots \leq \mu_{n-1}$ are the eigenvalues of the scaled Laplacian for the (i.j)-coalesced graph $\overline{L}(\tilde{G}(i,j))$.

Proof

For ease of presentation, let us consider $i = 1$ and $j = 2$. This is done WLOG, since the vertices in the graph can simply be relabeled in this way. The outline of the proof is as follows: we will consider the response of a linear time-invariant dynamical system defined from the Laplacian matrix above, and compute the response in two ways. Equivalencing the two forms will yield the desired result.

As a preliminary notational step, let us give the Jordan decomposition of $L(G)$. Since $L(G)$ is symmetric, it can be written in the form

$L(G) = \begin{bmatrix} \mathbf{1} & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 = 0 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}^T$, where each column of the matrix $\mathcal{V} = \begin{bmatrix} \mathbf{1} & \mathbf{v}_2 & \dots & \mathbf{v}_n \end{bmatrix}$ is an eigenvector of $L(G)$ normalized to unit length. Since the eigenvalue λ_q is assumed non-repeated, the qth column of $\mathcal{V}$ is the unique normalized eigenvector associated with λ_q, which we wish to characterize.

Now, let us consider the impulse response of the system

$$\dot{\mathbf{x}} = L(G)\mathbf{x} + \mathbf{e}_{12}u$$
$$y = \mathbf{e}_{12}^T\mathbf{x}, \tag{10.2}$$

where $\mathbf{e}_{12}$ is an n-component vector whose 1st component is 1, whose 2nd component is -1, and which is zero otherwise. It follows immediately from the Jordan form of $L(G)$ together with classical linear systems analysis that this impulse response is

$y(t) = \sum_{z=2}^{n} e^{\lambda_z t}(v_{z1} - v_{z2})^2$, $t \geq 0$. From this expression, we see that the coefficient of the exponential $e^{\lambda_q t}$ is $(v_{q1} - v_{q2})^2$; we shall obtain an alternate characterization of this coefficient in terms of the original and scaled Laplacian's eigenvalues, and hence obtain the desired results.

We can alternately find the response $y(t)$ by finding the transfer function of the system, and writing the transfer function in pole-residue form. We will relate the poles and zeros (and hence the transfer function) to the eigenvalues of $L(G)$ and $\overline{L}(\tilde{G}(1,2))$, hence obtaining an expression for the coefficient of $e^{\lambda_q t}$ in the dynamical response in terms of these eigenvalues and proving the equivalence. We first note that the characteristic function, and hence denominator of the transfer function (prior to any pole-zero cancellation), is $\prod_{z=1}^{n}(s-\lambda_z)$. We also must find the numerator of the transfer function. We do this in two steps: first, we demonstrate an equivalent feedback representation of the system (10.2). We then use this equivalent representation to determine the numerator.

The equivalent representation that we obtain is based on a more general reformulation of linear-time-invariant systems known as the *special coordinate basis*, that identifies the *invariant zeros* of the dynamics (we ask the reader to see [196,205] for further background). To present the equivalent representation, we find it convenient to define an $n-1$ component **zero state vector** $\mathbf{x}_a = \begin{bmatrix} \frac{x_1+x_2}{2} \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$.

We claim that that the dynamics of the system (10.2) can equivalently be written in terms of the zero state vector, as follows:

$$\begin{aligned} \dot{y} &= ay + \mathbf{c}^T\mathbf{x}_a + 2u \\ \dot{\mathbf{x}}_a &= \overline{L}(\tilde{G}(1,2))\mathbf{x}_a + \mathbf{b}y, \end{aligned} \tag{10.3}$$

where the specific values of a, $\mathbf{b}$, and $\mathbf{c}$ are not important to our development. To verify this claim, we note the following invertible relationship between the state vector $\mathbf{x}$ and the vector $\begin{bmatrix} y \\ \mathbf{x}_a \end{bmatrix}$:

$$\begin{bmatrix} y \\ \mathbf{x}_a \end{bmatrix} = \begin{bmatrix} 1 & -1 & \\ \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} \mathbf{x}$$

and

$$\mathbf{x} = \begin{bmatrix} \frac{1}{2} & 1 & \\ -\frac{1}{2} & 1 & \\ & & I_{n-2} \end{bmatrix} \begin{bmatrix} y \\ \mathbf{x}_a \end{bmatrix} \tag{10.4}$$

Using these expressions together with the definitions of y and $\mathbf{x}_a$, we obtain

$$\begin{aligned} \dot{y} &= \mathbf{e}_{12}^T \dot{\mathbf{x}} = \mathbf{e}_{12}^T L(G)\mathbf{x} + \mathbf{e}_{12}^T \mathbf{e}_{12} u \\ &= \mathbf{e}_{12}^T L(G) \begin{bmatrix} \frac{1}{2} \\ \frac{-1}{2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} y + \mathbf{e}_{ij}^T L(G) \begin{bmatrix} 1 & \\ 1 & \\ & I_{n-2} \end{bmatrix} \mathbf{x}_a + 2u. \end{aligned}$$

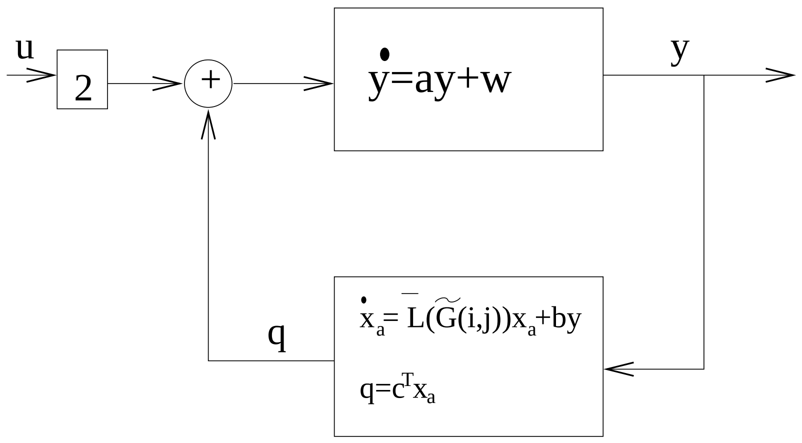


Fig. 10.1: A feedback representation of the system (2)

Thus, we have verified that $\dot{y}$ can be written in the form shown in (10.3). Similarly, we find that

$$\dot{\mathbf{x}}_a = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} \dot{\mathbf{x}} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} (L(G)\mathbf{x} + \mathbf{e}_{12}u)$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} L(G) \begin{bmatrix} 1 & \\ 1 & \\ & I_{n-2} \end{bmatrix} \mathbf{x}_a + \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} L(G) \begin{bmatrix} \frac{1}{2} \\ \frac{-1}{2} \\ 0 \\ \vdots \\ 0 \end{bmatrix} y.$$

Noting that $\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & \\ & & I_{n-2} \end{bmatrix} L(G) \begin{bmatrix} 1 & \\ 1 & \\ & I_{n-2} \end{bmatrix} = \overline{L}(\tilde{G}(1,2))$, we see that $\dot{\mathbf{x}}_a$ has the form given in (10.3). Thus, we have shown that the system (10.2) can be written in the form (10.3).

As a second step, let us use the form (10.3) to characterize the transfer function of the system (10.2). To do so, note the special form (10.3) clarifies that a projection of the state vector (in particular, the $(n-1)$-component vector $\mathbf{x}_a$) evolves without any direct influence from the input u, and without directly impacting the output y. That is, we see that the system can be viewed in the feedback form shown in Figure 1, i.e. one with a first-order dynamics in the forward path and an order-$(n-1)$ dynamics in the feedback path. Let us characterize the transfer function $H(s)$ from the feedback representation given in Figure 1. To do so, let us consider the transfer functions of the forward and feedback paths. We see that the forward path transfer function is $H_A(s) = \frac{1}{s-a}$. Meanwhile, using that the state matrix of the the feedback-path transfer function is $\overline{L}(\tilde{G}(1,2))$, we see that the transfer function (prior to any pole-zero cancellations) is $H_B(s) = \frac{r(s)}{s(s-\mu_2)\ldots(s-\mu_{n-1})}$, where $r(s)$ is a polynomial of degree less than $n-1$. From these path transfer functions, we obtain that

$$H(s) = 2\frac{H_A(s)}{1-H_A(s)H_B(s)} = \frac{2s(s-\mu_2)\ldots(s-\mu_{n-1})}{s(s-a)(s-\mu_2)\ldots(s-\mu_{n-1})+r(s)}.$$

Recalling that the denominator of the transfer function prior to pole-zero cancellation is the degree-n characteristic polynomial, we immediately see that the transfer function is $H(s) = \frac{2\prod_{z=2}^{n-1}(s-\mu_z)}{\prod_{z=2}^{n}(s-\lambda_z)}$.

It is classical that $H(s)$ can be written in the form $\sum_{z=2}^{n} \frac{A_z}{s-\lambda_z}$ through partial fraction decomposition, and hence that the impulse response can be written as $\sum_{z=2}^{n} A_z e^{\lambda_z t}$. Thus, we see that $(v_{qi} - v_{qj})^2 = A_z$, where A_z is obtained through partial fraction decomposition of $H(s) = \frac{2\prod_{z=2}^{n-1}(s-\mu_z)}{\prod_{z=2}^{n}(s-\lambda_z)}$. We thus shortly recover the result of the theorem. □

This theorem shows that eigenvector component differences for a Laplacian matrix with distinct eigenvalues can be explicitly computed in terms of these eigenvalues, as well as the eigenvalues of certain coalesced graphs' Laplacian matrices. These explicit relationships between eigenvalues and eigenvector components constitute an interesting interpretation of the role played by a graph's structure in its Laplacian's spectrum. Also, these relationships hold promise for providing bounds on eigenvector components in terms of the eigenvalues, and, conversely, for providing bounds on eigenvalues in terms of eigenvector components. Further, the result gives insight into optimal graph *designs* (for which certain Laplacian eigenvector components are specially structured, [1, 29]), and also possibly may provide graph-theoretic (specifically, Laplacian eigenvalue-based) interpretations for linear systems constructs (reachability, observability) in some dynamical-network models. Let us make a few remarks about the above result, and then give a two simple examples illustrating its potential uses.

Here are the remarks regarding the theorem:

1) Graph-theoretic characterizations of a feedback representation for linear systems, as used in the proof of Theorem 10.1, have also been developed in [7]. In that work, the characterizations were used in designing edge weights in a graph for the purpose of shaping an associated dynamics. We also stress that the feedback representation that we presented for the system (10.2), known as the *special coordinate basis* (SCB) representation, has been developed generally for linear time invariant systems and enjoys wide application in controls engineering [205]. The article [7] as well as this short note constitute a first effort to give graph-theoretic characterizations of the SCB.

2) When an eigenvalue of $L(G)$ is repeated, the corresponding eigenvectors form a subspace of R^n of dimension greater than 1. Thus, eigenvector components and their differences are not specified uniquely. However, it can be shown that there exists precisely one vector in this eigenspace (of normalized length) whose components can be explicitly calculated in terms of eigenvalues as in the theorem statement.

3) The proof of the theorem can be presented entirely using various canonical representations of the Laplacian matrix, i.e. without invoking the dynamical system (10.2). However, we feel

that such a proof is more clumsy and also less insightful than one that uses a dynamical-system construct.

Finally, let us give two examples that preliminarily illustrate application of the above theorem.

Example 1: Eigenvector-Based Bounds on Eigenvalue Augmentations

It is well-known that increasing an edge weight in a graph results increases (or at least does not decrease) all the corresponding Laplacian matrix's non-zero eigenvalues. This concept is valuable in numerous network engineering tasks wherein limited resources must be assigned to some links in the network, for example in traffic network design [7]. In the limiting case that a particular edge-weight is made large, it is well-known that the eigenvalues of the resulting graph's Laplacian monotonically approach those of the coalesced graph's scaled Laplacian, as defined above (see [7]). Thus, characterizing eigenvalues of the coalesced graphs' scaled Laplacians in terms of the original Laplacian matrix's eigenvector components can help to easily select edges (links) that are worthwhile to modify. Noting that the eigenvalues of each coalesced graph's scaled Laplacian are *interleaved* with the eigenvalues of the original graph's Laplacian, we straightforwardly recover from Theorem 10.1 the following bounds on $\mu_2(i,j)$, i.e. the second-smallest (or *Fiedler*) eigenvalue when vertices i and j are coalesced (assuming it is not repeated):

$$\lambda_2 + \frac{1}{2}(\lambda_3 - \lambda_2)(v_{2i} - v_{2j})^2 \leq \mu_2(i,j) \leq \lambda_2 + \frac{1}{2}(\lambda_n - \lambda_2)(v_{2i} - v_{2j})^2 \qquad (10.5)$$

We note that the presented upper bound can also be viewed as an upper bound on the Laplacian's Fiedler eigenvalue, upon any augmentation of the edge weight between vertices i and j.

Let us illustrate the bound using a small example. In particular, let us consider a 5 vertex line graph with identical (unity) weights for all four edges. The Fiedler eigenvalue of the graph's Laplacian is $\lambda_2 = 0.382$ in this case. If we apply the bound above for one of the leaf edges in the graph, we recover that $0.408 \leq \mu_2 \leq 0.467$. In contrast, for the interior edges, we find that $0.451 \leq \mu_2 \leq 0.606$. In fact, we find that μ_2 (or in other words the Fiedler eigenvalue when the edge weight is made large) equals 0.429 and 0.546, respectively, for the two cases. We note that the upper bounds provided are considerably less than the next-smallest eigenvalue, $\lambda_3 = 1.382$.

We also note that useful bounds can be found on other eigenvalues using the explicit expression for the eigenvector components; we omit the details.

Example 2: Finding Eigenvalue Dependencies on Edge Weights

The dependence of a Laplacian matrix's non-zero eigenvalues on an edge weight in the associated graph is often of interest. The eigenvalue-eigenvector relationship that we developed above shows that, surprisingly, this dependence can be determined from only the eigenvalues of Laplacians associated with two graphs: 1) the original graph, 2) the graph with the edge of interest condensed. In particular, noting that the sensitivity of each (non-repeated) eigenvalue λ_q to a particular edge weight k_{ij} is given by $\frac{d\lambda_q}{dk_{ij}} = (v_{qi} - v_{qj})^2$ and applying the above equivalence, we find that the dependence of the eigenvalues on k_{ij} can be found by solving the following set of differential equations*:

$$\frac{d\lambda_q}{dk_{ij}} = 2\frac{\prod_{z=2}^{n-1}(\mu_z - \lambda_q)}{\prod_{z=2,z\neq q}^{n}(\lambda_z - \lambda_q)},$$

$q = 1, \ldots, n-1$, where $\mu_2, \ldots, \mu_{n-1}$ are the (fixed) eigenvalues of the Laplacian of the graph with edge $\{i, j\}$ coalesced. We note that finding all the eigenvalue derivatives with respect to k_{ij} requires on the order of n^2 operations, and so the above differential equations are appealing for numerically approximating the dependence of the eigenvalues on k_{ij}.

* From the interlacing property, it is easy to see that the eigenvalues will remain non-repeated as k_{ij} is changed.

PART III: NETWORK CONTROL

In Part III, we develop novel dynamical controllers (memoried designs) to shape the dynamics of networks. Additionally, the tools permit shaping of dynamics under common limitations, such as saturation constraints and delays, and allow completion of a range of complex tasks in dynamical networks.

In Chapter 11, we introduce our novel decentralized controller design in the context of a double-integrator network. The controller is constructed by first designing a stabilizing (and high-performance) derivative controller with higher relative degree, and then implementing the controller using delay approximation. Chapter 12 studies an alternative implementation of the stabilizing derivative controller using lead compensators, and Chapter 13 studies the design of the multi-lead-compensator controller in the presence of input saturation.

11. A MULTIPLE-DERIVATIVE AND MULTIPLE-DELAY PARADIGM FOR DECENTRALIZED CONTROLLER DESIGN: INTRODUCTION USING THE CANONICAL DOUBLE-INTEGRATOR NETWORK

We are engaged in a major effort to design decentralized controllers for modern networks, that is fundamentally based on 1) applying feedback of multiple derivatives of local observations and 2) implementing these derivative feedbacks using multiple-delay controllers. Here, we fully motivate and introduce the design paradigm in the context of a canonical sensing-network model, namely a network of saturating double integrators with general sensing topology that is subject to measurement delays. In this context, we illustrate that our design paradigm yields practical high-performance (in particular, group pole-placement) decentralized controllers that exploit the network topology while distributing the complexity and actuation requirements among the agents.

11.1 Introduction

Decentralized feedback systems have long been of interest to the controls community [28, 66, 67]. In recent years, research in decentralized control has been re-invigorated by interest in such applications as cooperative control of autonomous vehicle teams, data fusion in sensor networks, and virus-spreading control, among others, (see the overviews [141,142,207,208], see also, e.g., [1,25,29]). In particular, the novel characteristics of these *sensing-agent networks* (networks of highly-limited autonomous agents with distributed communication/sensing capabilities [208]) has brought about a focus on understanding the role played by a network's topology in permitting stabilization and high-performance control. This focus has again made clear that very little is known about *designing* high-performance controllers for decentralized systems—even for the very specially structured sensing-agent networks—and hence new tools for design are badly needed.

We are engaged in a major effort to design stabilizing and high-performance yet practical controllers for decentralized systems, that is fundamentally based on 1) locally using feedback of multiple *derivatives* of the observation and 2) using multiple-delay control schemes to implement these multiple-derivative controllers. We show that this new methodology is capable of addressing many of the complexities that are common to modern decentralized systems (such as sensing-agent networks), including very general observation topologies, saturation nonlinearities, and inherent network delays. We shall describe aspects of this systematic methodology for design in several installments [14,215]. In this chapter, we fully motivate and introduce the design methodology using a canonical but very widely applicable sensing-agent network model, namely a network of double-integrator agents with general sensing/communicating topology (e.g. [25,143]). The complementary installments demonstrate application to uniform rank and more general decentralized plant models (including for modern infrastructure networks) [215], and flesh out the implementation of multiple-derivative feedback using multiple-delay controllers.

Given the long history of decentralized control, the reader may well wonder why new techniques are needed for decentralized controller design. In fact, the study of sensing-agent networks, as well as certain infrastructure networks such as air traffic management systems [2] and electric power systems [216], has made it clear that *a single agent cannot possibly provide the actuation or complexity required to control the whole network, and further the controllers must exploit the network topology to cooperatively achieve performance requirements.* Unfortunately, the bulk of the traditional decentralized control theory views the network as a disturbance that must be dominated by the local dynamics [66], and hence does not permit design of controllers that exploit the network topology.

The seminal work of Wang and Davison [28] does make the role played by the network explicit, in that it gives necessary and sufficient conditions for stabilization of decentralized systems based on *fixed modes* (see also, e.g., [67, 217, 218]). Their methodology is very much applicable to modern networks, and we have used it to address the foundational problem of determining whether a sensing-agent network can be stabilized [25]. Unfortunately, Wang and Davison's perturbation-based approach does not permit *constructive design* of practical high-performance or even stabilizing controllers. While several works have extended [28] toward allowing eigenvalue placement (and hence high performance) in addition to stabilization, these approaches essentially concentrate the complexity and extent of actuation/observation at a single agent, and hence also are unsuitable for our applications [66]. For these reasons, new tools for decentralized controller design are critically needed.

In this document, we develop a multiple-derivative and multiple-delay paradigm for controlling decentralized systems. Fundamentally, the derivatives of local observations provide the local controllers with information about the entire network's state, and so permit control. To develop practical implementations of these multiple-derivative controllers for modern (e.g. sensing-agent) networks, we pursue multiple-delay approximations for the multiple-derivative controllers (i.e., feedback controls where the actuation signals are combinations of multiple delayed observations), see [14] for further development of multiple-delay controllers. This use of delayed observations may at first seem surprising since delays often serve to destabilize feedback control systems [219], but it is also well known that properly-selected delays can be used effectively in control [13, 220]. This derivative/delay paradigm is a very natural one for decentralized systems, for which centralized notions of state estimation fail, and hence delays/derivatives provide the only known approach to finding the global state from local observations. What is surprising is that we can achieve not only stabilization but effective pole placement, while using only one more delayed observation (or one higher derivative) than is needed for centralized control. We thus are able to construct fully decentralized controllers with quite low complexity and distributed actuation effort. In this chapter, we conceptualize and illustrate the delay-based decentralized control paradigm, using as a canonical example the double-integrator-network model.

Decentralized systems, and in particular sensing-agent networks, are strongly impacted both by constraints on the agents and network limitations and variations. An essential advantage of our delay-based control methodology is its effectiveness even in the presence of these harsh constraints/limitations. Specifically, **actuator saturation** nonlinearities are ubiquitous in sensing-agent network applications [1, 25]. While controller design under saturation has been extensively studied for centralized systems [221], design under saturation for decentralized systems is wholely unknown (see [222] for partial *existence* conditions). In fact, our multiple-derivative/delay control scheme provides a natural avenue for design under actuation saturation. Further using low-gain ideas, we can naturally design multiple-delay controllers that stabilize networks with actuator saturation. Also, network communications/sensing are always subject to delays, and so controlling networks with inherent delays is critically important. Since our control strategy systematically uses delayed observations, it is eminently suited for networks with inherent delays. In particular, we show that networks with arbitrary and inhomogeneous delays can be stabilized with a low-gain controller. These results for networks with saturation and delay indicate the wide applicability of our methodology for practical controller design.

We stress here that the delay-based control methodology is applicable to general linear time-invariant decentralized control systems, and so the reader may wonder why we have chosen to introduce the methodology using only a canonical example. In fact, focusing on the double-integrator network permits a clearer and simpler presentation for two reasons: 1) it permits a full characterization of the derivative-based and hence delay-based controller's performance and implementation from first principles (Sections 11.2 and 11.3), without requiring the complicated *special coordinate basis* (see [205] and also [215]) , and 2) the time-scaling properties of the double-integrator network permits simple design of low-gain multiple-delay controllers (Section 11.4). We feel strongly that presenting results in this simpler context allows us to expose the conceptual underpinnings of derivative/delay-based decentralized control, and to clearly develop the (rather intricate) tools for analyzing multiple-delay controllers. We also note that sensing-agent networks, and in particular double-integrator networks, are of wide current interest [25, 208] and so deserve an explicit treatment.

11.2 Controlling the Double-Integrator Network

In this section, we illustrate our methodology of using multiple derivative feedbacks to achieve stabilization and high-performance control, in the context of a decentralized **double integrator network** (see [25]). In this simple network model, each agent has a double-integrator internal dynamics, and observes only a linear combination of the states of some agents. Although simple, this network model is widely applicable, including for various autonomous-vehicle control and sensor-networking tasks [25, 141, 143]. We present our controller design for this simple but very widely-applicable decentralized network, to highlight the conceptual foundation for the methodology. Specifically, we show that, using linear feedback of derivatives of observations up to order 2 for each agent, we can stabilize the double integrator network. Moreover, we can place groups of poles at arbitrary locations or in desirable ranges using high gain control. The fundamental concept underlying this design is that feedback of derivatives of the output up to the relative degree of the local plant (2 in our case) gives each agent enough information about the global state to permit high-performance control through, essentially, plant inversion; we notice that one more derivative is needed then for centralized control of the plant, see the literature on asymptotic time-scale and eigenstructure assignment (ATEA design) and our recent application of it to multiple-delay control of centralized plants [13,196]. In this section, we first present the controller design (Section 11.2.1), and then give a conceptual discussion of the design method and its characteristics (Section 11.2.2).

11.2.1 Multiple-Derivative Controller Design

Formally, consider a linear time-invariant (LTI) system consisting of n double-integrator agents, i.e. described by

$$\begin{aligned} \ddot{\mathbf{x}}(t) &= \mathbf{u}(t) \\ \mathbf{y}(t) &= G\mathbf{x}(t), \end{aligned} \tag{11.1}$$

where $\mathbf{x}(t) \in \mathcal{R}^n$ represents the positions of the n agents, $\mathbf{u} \in \mathcal{R}^n$ and $\mathbf{y} \in \mathcal{R}^n$ are the inputs and observations respectively, and matrix $G = \{G_{ij}\}_{n\times n}$. Note that each agent i has only one input u_i and makes only one observation y_i, which is a linear combination of the positions of other agents, i.e., $y_i = [G_{i1}, ..., G_{in}]\,\mathbf{x}$. Further, each agent i has its own local feedback control law, which constructs its input u_i from its local observation y_i. We refer to this system as a **double-integrator network** (see [25]).

The condition to stabilize such a network using a linear time-invariant controller is that G has full rank (from [25], based on Wang and Davison's classical existence result [28]). Unfortunately, the existence condition seemingly does not translate to a simple controller design: the system can not always be stabilized with a static decentralized feedback controller $u(t) = Ky(t)$ (K diagonal), nor can it always be stabilized with position and velocity feedback (i.e. using a control law of the form $\mathbf{u}(t) = K_1 G\mathbf{x}(t) + K_2 G\dot{\mathbf{x}}(t)$, K_1 and K_2 diagonal) [25]. However, by introducing one more derivative to the feedback control—specifically by using the control law $\mathbf{u}(t) = k_1k_3\mathbf{y}(t) + k_2k_3\dot{\mathbf{y}}(t) + k_3\ddot{\mathbf{y}}(t)$, where k_1, k_2, k_3 are some properly chosen scalars — it turns out that we suddenly gain the ability to achieve stabilization and high performance (in particular, a "group" pole placement, where groups of n poles are placed at desirable locations). It is valuable to note that all agents have the same gains k_1, k_2 and k_3: one does not even need to employ agent-specific gains for stabilization and high-performance control.

We can design this multiple-derivative-based control law using a simple algorithm. We first describe the algorithm, and then formally show that the two tasks (stabilization and high-performance control) can be completed using the multiple-derivative-based controller.

Algorithm The following is the algorithm for designing the multiple-delay-based controller:

1) Choose two constants, say k_1 and k_2, such that the roots of $\lambda^2 + k_1\lambda + k_2$ are at desirable locations.

2) Choose k_3 sufficiently large, and apply the control law

$$\mathbf{u} = k_1k_3\mathbf{y} + k_2k_3\dot{\mathbf{y}} + k_3\ddot{\mathbf{y}} \tag{11.2}$$

We will show that the algorithm yields not only stabilizing controllers, but ones with closed-loop poles near to the roots of $\lambda^2 + k_1\lambda + k_2$. The above algorithm for designing the derivative-based controller, and the justification that it achieves stabilization/performance goals, is essentially based on using the second derivative of the observation in feedback with high gain (k_3 large) to effectively permit local control of agents' states.

In Theorem 11.1, we state the main result concerning stabilization of the double integrator network using the multiple-derivative-based controller (Equation 11.2).

Theorem 11.1. *Consider the double-integrator network described in (11.1), where G is nonsingular. The network can be stabilized using the multiple-derivative control law (Equation 11.2) with k_1, k_2 and k_3 satisfying: $a_{i1} > 0$, $\frac{a_{i1}a_{i2}-a_{i3}}{a_{i1}} > 0$, $\frac{a_{i1}^2 a_{i4}+a_{i3}(a_{i3}-a_{i1}a_{i2})}{a_{i3}-a_{i1}a_{i2}} > 0$ and $a_{i4} > 0$ for all i, where $a_{i1} = -2Re(\frac{k_2k_3\lambda_i}{1-k_3\lambda_i})$, $a_{i2} = -2Re(\frac{k_1k_3\lambda_i}{1-k_3\lambda_i}) + \left|\frac{k_2k_3\lambda_i}{1-k_3\lambda_i}\right|^2$, $a_{i3} = 2Re(\frac{k_1k_3\lambda_i}{1-k_3\lambda_i})Re(\frac{k_2k_3\lambda_i}{1-k_3\lambda_i}) + 2Img(\frac{k_1k_3\lambda_i}{1-k_3\lambda_i})Img(\frac{k_2k_3\lambda_i}{1-k_3\lambda_i})$, $a_{i4} = \left|\frac{k_1k_3\lambda_i}{1-k_3\lambda_i}\right|^2$, and λ_i is the ith eigenvalue of G. As a special case, if G has real eigenvalues, the network can be stabilized with k_1, k_2 and k_3 satisfying: $k_1 > 0$, $k_2 > 0$ and $k_3 > \frac{1}{min(\lambda(G)>0)}$ (where $min(\lambda(G) > 0)$ denotes the minimum positive eigenvalue of G) .*

Proof: We study the closed-loop poles of the system using the control law (Equation 11.2). The state-space of the closed loop system thus is $\dot{\mathbf{X}} = A_c\mathbf{X}$, where $\mathbf{X} = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{x} \end{bmatrix}$ and

$$A_c = \begin{bmatrix} (I-k_3G)^{-1}k_2k_3G & (I-k_3G)^{-1}k_1k_3G \\ I & 0 \end{bmatrix}.$$

Denoting $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ as the right eigenvector of A_c, we have $A_c \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, which implies that $x_1 = \lambda x_2$, and $\lambda(I-k_3G)^{-1}k_2k_3Gx_2 + (I-k_3G)^{-1}k_1k_3Gx_2 = \lambda x_1$. The latter yields: $\lambda k_2k_3Gx_2 + k_1k_3Gx_2 = \lambda^2(I-k_3G)x_2$, or $(\lambda k_2k_3 + k_1k_3 + \lambda^2k_3)Gx_2 = \lambda^2x_2$. This means that x_2 must be an eigenvector of G with a eigenvalue, say λ_i. Hence, we have $(\lambda k_2k_3 + k_1k_3 + \lambda^2k_3)\lambda_i = \lambda^2$, or $\lambda^2 - \frac{k_2k_3\lambda_i}{1-k_3\lambda_i}\lambda - \frac{k_1k_3\lambda_i}{1-k_3\lambda_i} = 0$. The closed-loop poles are the roots of the characteristic equations $\lambda^2 - \frac{k_2k_3\lambda_i}{1-k_3\lambda_i}\lambda - \frac{k_1k_3\lambda_i}{1-k_3\lambda_i} = 0$, for all i. Placing the roots of $\lambda^2 - \frac{k_2\lambda_i}{1-k_3\lambda_i}\lambda - \frac{k_1\lambda_i}{1-k_3\lambda_i} = 0$ in the Open Left Half Plane (OLHP) is equivalent to placing the zeros of $(\lambda^2 - \frac{k_2k_3\lambda_i}{1-k_3\lambda_i}\lambda - \frac{k_1k_3\lambda_i}{1-k_3\lambda_i^*})(\lambda^2 - \frac{k_2k_3\lambda_i^*}{1-k_3\lambda_i^*}\lambda -$

$\frac{k_1 k_3 \lambda_i^*}{1-k_3\lambda_i^*}) = 0$ into the OLHP, since the latter has two conjugate complex root pairs, each pair specifying a root of the original equation. As a special case, when λ_i is real, the latter has two repeated roots corresponding to each root in the original equation. The Routh Criterion naturally leads to the conditions for stabilization. □

It is easy to check that the conditions on the gains in Theorem 11.1 can always be satisfied by choosing k_1 and k_2 positive and k_3 sufficiently large. Theorem 11.1 states that by introducing the derivatives of observations $\mathbf{y}(t), \dot{\mathbf{y}}(t)$ and $\dot{\mathbf{y}}(t)$ into the control law as in (11.2), the decentralized system can be stabilized whenever G has full rank. This result is significant in that it gives an explicit controller design for stabilization of a double integrator network, rather than only giving conditions for the existence of such a controller. We will see that this design far outperforms single-channel-based designs (Section 11.2.2).

Using derivatives in the control law also allows performance design, e.g., placing groups of closed-loop poles at pre-defined positions or within ranges.

Theorem 11.2. *Consider the double-integrator network described in (11.1), where G is nonsingular. The closed-loop poles of the network can be placed arbitrarily near to any two pre-defined locations x_A and x_B (on the real axis or as a conjugate pair) using the multiple-derivative controller (Equation 11.2), by setting k_3 sufficiently large and choosing k_1 and k_2 such that $k_2 = -(x_A + x_B)$ and $k_1 = x_A x_B$.*

Proof: From the proof of Theorem 11.1, we know that the closed-loop roots are the zeros of the characteristic equations $\lambda^2 - \frac{k_2 k_3 \lambda_i}{1-k_3\lambda_i}\lambda - \frac{k_1 k_3 \lambda_i}{1-k_3\lambda_i} = 0$, for all i. Hence, when k_3 is sufficiently large, the coefficients of the characteristic equation approach the coefficients of the quadratic equation $\lambda^2 + k_2\lambda + k_1 = 0$. From the continuous dependence of roots on parameters, the closed-loop poles thus approach the roots of this characteristic equation. The result follows with just a little algebra.

□

This theorem states that, by using sufficiently high gains, we can place all the closed-loop poles arbitrarily close to any two predefined locations, with n poles at each location. Moreover, one can see through the root locus that when we decrease k_3 from a high value, the poles that are originally close to each pre-defined location separate, with speeds dependent on k_1, k_2, k_3, and λ_i. Thus, by choosing proper k_1, k_2 and k_3, we can place the poles in specified ranges.□

Motivated by Theorem 11.2, we define **group pole placement** as the task of placing a group of poles of a decentralized system near (or within a range of) some pre-defined positions in the complex plane. Note that group pole placement is different from exact pole placement in that we only place sets of poles within a range or near a location, rather than design the exact location of each pole. However, group pole placement is a much stronger achievement than stabilization. Group pole placement design allows us to set such performance statistics as the dominant eigenvalue and dominant eigenvalue ratio [29], and further through an inverse-optimality argument the design can be shown to achieve phase-margin requirements [223]. The derivative-based controller permits us to achieve group pole placement, with the locations of the groups of poles and the closeness of the poles within each group depending on the values of the gains that we have chosen.

11.2.2 Discussion: Concepts and Comparisons

It is worthwhile to further discuss our multiple derivative controller, so as to better interpret how it works (Section 11.2.2) and compare it to existing approaches for decentralized control (Section 11.2.2).

A Structural Interpretation

Our design methodology is based on using multiple-derivative-control—and specifically using one higher derivative than is needed for centralized pole placement (see the work on ATEA design [196])—to achieve high-performance decentralized control. The use of derivative-based control for decentralized systems is sensible (broadly speaking), in that derivatives of linear-system outputs identify the global state and so should facilitate control at each channel of a decentralized system. What is surprising is that precisely as many derivatives as the relative degree of the local plant, or one more than is needed for centralized control, is sufficient to provide each agent with enough state information to permit stabilization and group pole placement. Here, we give some further conceptual discussion of this special characteristic of the multiple-derivative control.

Specifically, let us argue that our controller, which uses an "extra derivative", implicitly and distributedly provides each agent with the *local* state information and so permits simple control of n sets of "local" dynamics. To see why this is the case, notice that the positions can be found from the observations as $\mathbf{x} = G^{-1}\mathbf{y}$, and similarly the velocities can be found as $\dot{\mathbf{x}} = G^{-1}\dot{\mathbf{y}}$. Thus, if each agent can be given the statistics $\mathbf{h}_i^T\mathbf{y}$ and $\mathbf{h}_i^T\dot{\mathbf{y}}$, where $\mathbf{h}_i^T$ is the ith row of the matrix G^{-1}, then it has available the local position and velocity. However, the use of the extra derivative in the decentralized controller implicitly does exactly this. In particular, note that upon application of the multiple-derivative control, the closed-loop dynamics are $\ddot{\mathbf{x}} = (I-k_3G)^{-1}k_1k_3\mathbf{y}+(I-k_3G)^{-1}k_2k_3\dot{\mathbf{y}}$, where we have written the right-hand side in terms of the observation $\mathbf{y}$ rather than the state. Thus, $\ddot{\mathbf{x}} = k_1(\frac{1}{k_3} - G)^{-1}\mathbf{y} + k_2(\frac{1}{k_3} - G)^{-1}\dot{\mathbf{y}}$. For k_3 sufficiently large, $\ddot{x}_i \approx k_1\mathbf{h}_i^T\mathbf{y} + k_2\mathbf{h}_i^T\dot{\mathbf{y}} = k_1x_i + k_2\dot{x}_i$. That is, each agent is approximately feeding back its local state and derivative, i.e. locally using a proportional-derivative controller, to achieve performance requirements. Such local control of identical double-integrators is of course straightforward, and so we automatically infer the possibility for group pole placement. We notice that this approach is a fundamentally distributed one, in the sense that actions at each channel are together permitting computation of the local state and control using it.

The above discussion clarifies that, fundamentally, the derivative-based controller achieves high performance by distributedly providing each agent with local state information from the network observation. Several further points regarding this viewpoint are worthwhile:

a) We note the great difference of this approach to the traditional approach taken in decentralized control, where knowledge of local state is *assumed* and is used to dominate the network interactions rather than being meshed with them [66].

b) The above discussion clarifies that decentralized control is greatly simplified *whenever* the agents are provided with the statistics $\mathbf{h}_i^T\mathbf{y}$ and $\mathbf{h}_i^T\dot{\mathbf{y}}$, whether by derivative-based control or through another means. For instance, direct communication of appropriate observations so as to permit computation is an alternative.

c) This viewpoint motivates us to seek better characterizations of the matrix G^{-1}, in terms of the network topology codified in G. The relationship between the structure of the topology matrix G and that of its inverse is generally complicated. Even when the matrix G is sparse, G^{-1} may be dense, and hence we see that conceptually the statistics needed by each agent require observations from throughout the network (which are rather slickly provided to the agent through use of one higher derivative). These statistics are much simplified, or amenable to interesting interpretations, for special classes of topology matrices such as those with a slow-coherent structure. We leave it to future work to make precise the structure of G^{-1} for these special classes.

Comparison with the Dominant Channel Approach

We have argued that our design is fundamentally different from the existing approaches for decentralized pole placement [67], in that it distributes the complexity and actuation among the channels (agents) rather than concentrating them at one channel. Here, we make explicit the advantage in complexity and actuation provided by our approach. Precisely, we show through an example that high complexity and large actuation may be needed in a single channel when the existing methods are used, as compared to when the multiple derivative controller is used. We recall that in the existing approaches, the entire network dynamics are made controllable and observable from a single channel through static feedback, and then a linear dynamic controller is implemented at this channel for pole placement (using standard method for centralized systems); it is this dominant channel approach that we will compare to our multiple-delay-based design.

First, let us compare the controller complexities for dominant channel-based approach and the multiple-derivative design, in the context of the double integrator network. For the multiple-derivative design, each agent requires precisely three signals (y_i, $\dot{y}_i$, and $\ddot{y}_i$), which are linearly combined to generate the actuation signal, regardless of the network topology; these signals can be approximated arbitrarily well using either 3 variously-delayed observations or a lead compensator with two poles, see the implementation section (Section 11.3) for further details. On the other hand, as we shall show below, the dominant channel approach necessitates use of a dynamic controller of order $2n$ at a single agent for certain network topologies. The comparison is formalized in the following theorem:

Theorem 11.3. *Consider a double-integrator network with full graph matrix* $G = \begin{bmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ 1 & & & \end{bmatrix}$.

If the dominant channel approach for decentralized pole placement is used, then one agent requires a controller that has dynamic order $2n$. *In contrast, the multiple derivative design requires a linear feedback of the observation and its first two derivatives, which can be implemented using three delayed observations or else a dynamic controller of order* 2 *at each channel for group pole placement.*

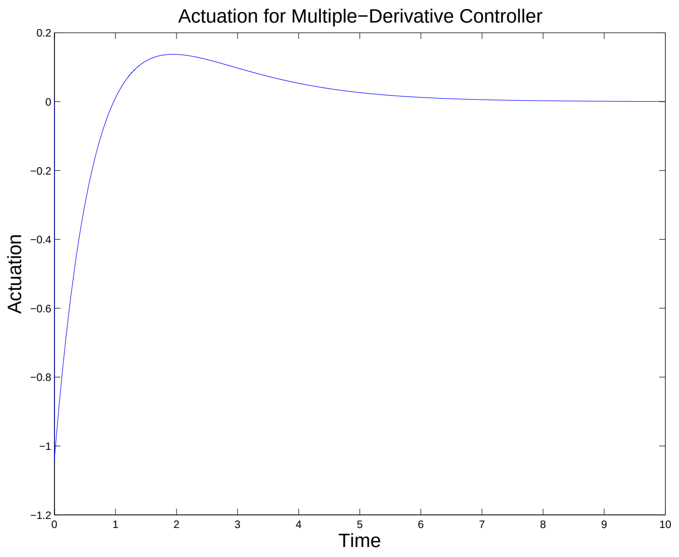
Proof: The result is automatic for the multiple-derivative design.

In the dominant channel approach, a static linear feedback is applied to each agent, and then this closed-loop system is controlled from any single channel (which we can choose WLOG to be the channel n, from symmetry). Specifically, let us consider applying the controls $u_i = p_i y_i$ for channels $1, \dots, n-1$ and the control $u_n = p_n y_n + w_n$, and then consider designing a further feedback in channel n (from y_n to w_n). With a little algebra, we find that the (SISO) transfer function from w_n to y_n for the example full graph matrix G is $\frac{1}{s^{2n}+P}$, where $P = p_1 p_2 \dots p_n$. We claim that feedback control of this plant for the purpose of pole placement requires a dynamic controller of order $2n$. To see why, consider applying a strictly proper linear dynamic controller $\frac{r(s)}{q(s)}$ to the plant. Then notice that the characteristic polynomial of the closed-loop system is $\gamma(s) = s^{2n}q(s) + Pq(s) + r(s)$. If the degree of $q(s)$ is m, notice that the only non-zero coefficients in the characteristic polynomial are those for the terms $s^{2n}, \dots, s^{2n+m}$, and $1, \dots, s^m$. Then we require $m \geq 2n-1$ to achieve stability, let alone pole placement. In fact, even a controller of order $m = 2n-1$ is not sufficient for pole placement since the ratio between the coefficients of s^{4n-1} and s^{2n-1} is fixed. Thus, a controller of order $2n$ is needed for pole placement. □

Let us also, through an example, compare the actuation needed by the agents for the multiple-derivative design and the dominant-single-channel approach. In particular, we consider a double-integrator network with $n = 5$ agents, and the cyclic graph G from Theorem 11.3. In this example, we assume that the agents are initially at position 1 and have velocity 0.

For the multiple-derivative design, we use parameter $k_1 = 1$, $k_2 = 2$, and $k_3 = 25$. We notice that this design aims to place all the eigenvalues at $s = -1$. It turns out that the gain k_3 has been chosen large enough that, in fact, the eigenvalues are to the left of $s = -0.8$. We note that the actuation signal for each agent is identical in this example. This common actuation signal is shown in Figure 11.1.

For the dominant single-channel approach, we initially apply unity static gains at each agent. We then place all the poles at -0.8 using dynamic feedback at (WLOG) agent 5, where we have conservatively chosen to place the poles at -0.8 to ensure that the comparison of the two controllers is fair. The actuation signal for agent 5 for this controller is shown in Figure 11.1. We notice that the magnitude of the required actuation signal is very large compared to that for the multiple-delay-based design (roughly, by a factor of 10^4 over a single agent's actuation in the multiple-delay design). Very similar results are obtained for other initial conditions, and for designs where randomly-chosen static gains are used rather than uniform ones. This need for large actuation is not surprising: in particular, a large effort is needed to move Agent 1 using the controller at Agent 5 (and in fact this also cause large swings in the location of Agent 1 in part of the transient).



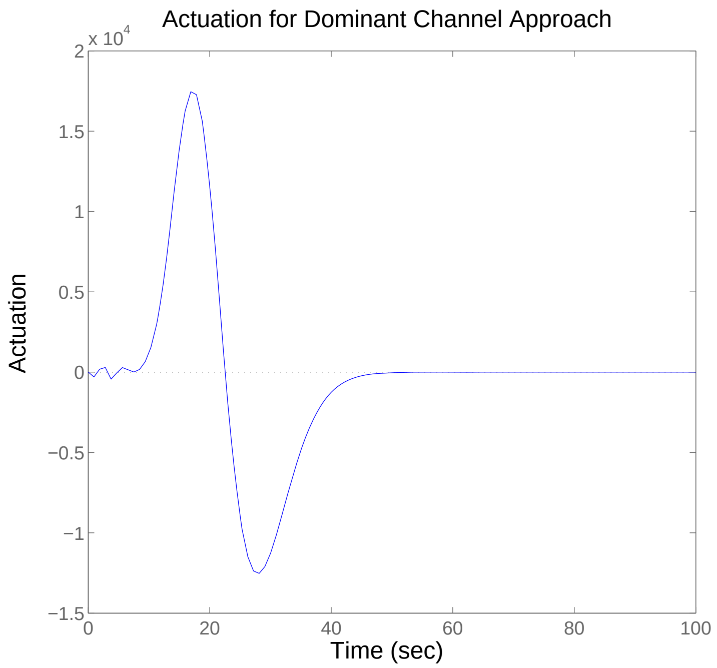


Fig. 11.1: *Top:* Actuation needed for multiple derivative control. *Bottom:* Actuation needed for control through a single dominant channel.

11.3 Implementation Issues: Overview

So far, using the double-integrator network as a canonical example, we have introduced a philosophy for decentralized control based on each channel (agent) using multiple derivatives of its local measurements. Here, we shall discuss the realistic implementation of such multiple-derivative controllers, with a particular focus on *multiple-delay control* (see e.g. [13, 14, 220] for background). In the interest of space, our development here is only an overview, with a focus on providing the user with working controllers for typical network topologies, discussing briefly the implementation for general topologies, and explaining to the reader the richness of this multiple-derivative-control task. Further details can be found in our related works [13, 14].

Fundamentally, our control scheme for the double integrator requires that each agent obtain and feed back the first two derivatives of its observations, or in other words the derivatives up to the relative degree of the local plant. Traditionally, derivatives of observations are obtained through either *1)* explicit measurement (e.g., measurement of vehicle velocities) or *2)* approximation of the derivative with a proper transfer function, e.g. through use of lead compensation or, less commonly, multiple-delay approximations [13, 14, 220]. Approximation of derivatives up to one less than the relative degree of a plant can be done systematically, so that a feedback system using these approximations has performance arbitrarily close to one actually using the derivatives. That is, the finite closed-loop poles upon use of the approximating controller can be made to approach the ones for the derivative-based control system, while all other poles* are driven to $-\infty$.

On the other hand, approximations of derivatives of order *equal* to the relative degree of the plant in the feedback can, if improperly used, produce highly unstable spurious dynamics (while in many other cases harmlessly replicating the derivative-based control). This possibility for instability

* Notice that the multiple-delay-controlled system is infinite-dimensional and has an infinite number of such poles.

essentially results from the effective delay that is imposed by any implementable approximation combined with the possible non-continuity of this derivative, at least at an initial time, see e.g. [224] for a treatment of the phenomenon. Luckily, for the double-integrator-network, even the most basic multiple-delay-based or lead-compensation-based control implementations are successful for many typical network topologies. In the cases where these basic schemes are not successful, a variety of alternatives are available, including *1)* clever selection of the approximation used by each agent, *2)* adaptation to eliminate potential unstable dynamics, and *3)* explicit use of the relationship between the highest derivative and the lower derivatives and input through communication of a few observations/inputs between agents. We have shown that the first of these alternatives is sufficient for addressing approximation for arbitrary network topologies [14], but each alternative may have certain advantages/disadvantages in implementation and deserves further study. It is this wealth of alternative approaches that yields a rich problem space in the arena of approximating multiple-derivative-based control. Here, let us describe the basic approximation scheme and delineate the network topologies to which it applies (without proof in the interest of space), and then ruminate on the alternatives.

As noted above, either lead compensation or multiple-delay-based designs can be used. Since networks are very often subject to intrinsic delays, designs using multiple delayed observations are naturally applicable to network tasks, and so we shall focus on multiple-delay-based controls. A basic approach to implement the multiple-derivative controller is to approximate derivatives with identical delays, e.g., the first derivative $\dot{y}(t)$ can be approximated as $\frac{y(t)-y(t-\tau)}{\tau}$, where τ is a small delay. Higher derivatives can similarly be approximated by interpolating the observation with a polynomial [13, 220]. Here, we consider using the decentralized control law $\mathbf{u}(t) = \alpha_1\mathbf{y}(t-\tau_1) + \alpha_2\mathbf{y}(t-\tau_2) + \alpha_3\mathbf{y}(t-\tau_3)$, where $0 \leq \tau_1 < \tau_2 < \tau_3$, α_1, α_2, and α_3 are some properly chosen scalars, to stabilize the double integer network. The controller parameters can be selected using the following simple algorithm.

1) Choose two constants, say k_1 and k_2, such that the roots of $\lambda^2 + k_1\lambda + k_2$ are at desirable locations.

2) Choose a set of times $0 \le \bar{\tau}_1 \le \bar{\tau}_2 \le \bar{\tau}_3$, which will specify the relative spacing in time between the delayed measurements used by the controller, see Equation 11.3 below. The delays $\bar{\tau}_1$, $\bar{\tau}_2$, and $\bar{\tau}_3$ need to be properly chosen so that a large k_3 does not introduce closed-loop poles in the ORHP (see [14] for the details).

3) Apply the control law:

$$\begin{aligned}\mathbf{u} = \frac{k_3}{\Delta}\left[k_1\left(\bar{\tau}_2\bar{\tau}_3^2 - \bar{\tau}_2^2\bar{\tau}_3\right) - k_2\frac{\bar{\tau}_2^2 - \bar{\tau}_3^2}{\epsilon} + 2\frac{\bar{\tau}_3 - \bar{\tau}_2}{\epsilon^2}\right]\mathbf{y}(t - \epsilon\bar{\tau}_1) + \\ \frac{k_3}{\Delta}\left[k_1\left(\bar{\tau}_3\bar{\tau}_1^2 - \bar{\tau}_3^2\bar{\tau}_1\right) - k_2\frac{\bar{\tau}_3^2 - \bar{\tau}_1^2}{\epsilon} + 2\frac{\bar{\tau}_1 - \bar{\tau}_3}{\epsilon^2}\right]\mathbf{y}(t - \epsilon\bar{\tau}_2) + \\ \frac{k_3}{\Delta}\left[k_1\left(\bar{\tau}_1\bar{\tau}_2^2 - \bar{\tau}_2\bar{\tau}_1^2\right) - k_2\frac{\bar{\tau}_1^2 - \bar{\tau}_2^2}{\epsilon} + 2\frac{\bar{\tau}_2 - \bar{\tau}_1}{\epsilon^2}\right]\mathbf{y}(t - \epsilon\bar{\tau}_3),\end{aligned} \tag{11.3}$$

where $\Delta = \begin{vmatrix} 1 & \bar{\tau}_1 & \bar{\tau}_1^2 \\ 1 & \bar{\tau}_2 & \bar{\tau}_2^2 \\ 1 & \bar{\tau}_3 & \bar{\tau}_3^2 \end{vmatrix}$, k_3 is chosen sufficiently large, and ϵ is a sufficiently small number.

This multiple-delay controller is based on the approximation of the multiple-derivative controller $\mathbf{u} = k_1k_3\mathbf{y} + k_2k_3\dot{\mathbf{y}} + k_3\ddot{\mathbf{y}}$, i.e. the multiple-derivative controller that we showed in Section 11.2 to achieve group pole placement. From the results in [14], and using the fact that G can always be pre-scaled by a diagonal gain matrix in decentralized control, we recover that the delay-based controller (Equation 11.3) with $\mathbf{y} = KG\mathbf{x}$ is equivalent in a pole-placement sense to the multiple-derivative control[†] whenever the eigenvalues of KG are in the OLHP. Matrices G whose eigenvalues can be placed in the OLHP by a diagonal scaling include all those that have a sequence of n nested principal

[†] Notice that we have excluded this pre-scaling up to this point for the sake of simplicity of presentation.

minors with full rank (e.g., [25]). Positive definite matrices such as grounded Laplacian matrices and diagonally-dominant matrices, which are common in many applications such as autonomous vehicle control and sensor network management ones, of course fall in this class (and require only pre-scaling by $-I$). Let us complete the discussion with a claim. We expect that derivative-based controllers can be implemented for *arbitrary* topologies simply by using inhomogeneous multiple-delay controllers.

We also note that multi-lead-compensator architecture can achieve stabilization and pole placement of a double integrator network with arbitrary topology [11].

11.4 Stabilization Under Constraint and Delay

Limitations, e.g., measurement delays and input saturation, are common in decentralized systems. These limitations pose further difficulty for stabilization. In the centralized setting, low-gain techniques have been long used for designing stabilizing controllers under saturation constraints and intrinsic delays [198, 225]. In the decentralized setting, Stoorvogel and coworkers recently obtained a check for the existence of a stabilizing control under actuator saturation [222], for plants without defective jw-axis eigenvalues. However, no effort has been devoted to designing stabilizing controllers for decentralized systems with these limitations. In this section, we will explore how to stabilize a double integrator network with measurement delay and/or input saturation using a multiple-delay controller. The scaling property associated with pure integrators facilitates the analysis, and so allows us to highlight the concepts underlying low-gain delay for decentralized systems with little technical complexity; we shall address the low-gain design more generally in future work.

It is worth stressing that the low-gain (scaling) arguments that we use here could have alternatively been used to design multiple-derivative controllers that operate in the presence of saturation and delay. However, it is critical that our implementation of the controllers—which may involve using certain high-gains, e.g. in approximating derivatives as delay differences—operate in the presence of derivatives and delays, and hence we find it more instructive to work directly with the multiple-delay controllers. For ease of presentation, we will show how the basic multiple-delay controller introduced in Section 11.3 can be modified to permit control under delay and actuator saturation, although similarly other multiple-delay-based or lead-compensation implementations can be adapted.

11.4.1 Design for Networks with Measurement Delay

Intrinsic delays in measurement pose one of the primary challenges in achieving control of network dynamics. It is well known that measurement delay can cause poor performance and even instability in linear control systems. However, as delays are inherent to our controller implementation, it is possible for us to account for this measurement delay,

Specifically, consider an LTI system consisting of n double integrators with measurement delay, i.e. described by

$$\begin{aligned} \ddot{\mathbf{x}}(t) &= \mathbf{u}(t) \qquad (11.4) \\ y_i(t) &= G_i\mathbf{x}(t-\tau_i), \end{aligned}$$

where $\mathbf{x}(t)$ represents the positions of the n agents and $\mathbf{u}(t)$ the inputs, $y_i(t)$ is the observation made by agent i, $G_i = \begin{bmatrix} G_{i1} & \dots & G_{in} \end{bmatrix}$, and $\tau_i \geq 0$ is the (known) time delay in the measurement made by agent i. Since the delays are assumed known, we can further delay the observations in appropriate channels for the purpose of feedback, so WLOG let us henceforth assume a common delay τ which is the maximum of $\tau_1, \dots, \tau_n$ in each channel.

In order to stabilize (11.4), let us use a modification of the delay-based decentralized controller given in Equation 11.3:

$$\begin{aligned}
\mathbf{u}(t) = {} & \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_2\bar\tau_3^2 - \bar\tau_2^2\bar\tau_3\right) - k_2\frac{\bar\tau_2^2-\bar\tau_3^2}{\epsilon} + 2\frac{\bar\tau_3-\bar\tau_2}{\epsilon^2}\right]\mathbf{y}(t) + \\
& \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_3\bar\tau_1^2 - \bar\tau_3^2\bar\tau_1\right) - k_2\frac{\bar\tau_3^2-\bar\tau_1^2}{\epsilon} + 2\frac{\bar\tau_1-\bar\tau_3}{\epsilon^2}\right]\mathbf{y}(t-\rho\epsilon\bar\tau_2+\tau) + \\
& \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_1\bar\tau_2^2 - \bar\tau_2\bar\tau_1^2\right) - k_2\frac{\bar\tau_1^2-\bar\tau_2^2}{\epsilon} + 2\frac{\bar\tau_2-\bar\tau_1}{\epsilon^2}\right]\mathbf{y}(t-\rho\epsilon\bar\tau_3+\tau),
\end{aligned} \tag{11.5}$$

where we choose ϵ, $\bar\tau_i > 0$ and k_i , $i = 1, 2, 3$ in the same way as for the basic multiple-delay controller, and then choose ρ such that $\rho = \frac{\tau}{\epsilon\bar\tau_1}$. We formalize the design of the control law for a double integrator network with measurement delay in Theorem 11.4:

Theorem 11.4. *Consider the double-integrator network with measurement delay described in (11.4), where G is nonsingular, and assume that the corresponding undelayed double-integrator can be stabilized using the basic multiple-delay controller (11.3). The network can be stabilized using the delay-based control law (Equation 11.5) by choosing sufficiently small ϵ, $\rho = \frac{\tau}{\epsilon\bar\tau_1}$, and k_1, k_2 and k_3 satisfying the conditions given in Theorem 11.1.*

Proof: It is easy to check that the closed-loop system is

$$\begin{aligned}
\ddot{\mathbf{x}}(t) = {} & \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_2\bar\tau_3^2 - \bar\tau_2^2\bar\tau_3\right) - k_2\frac{\bar\tau_2^2-\bar\tau_3^2}{\epsilon} + 2\frac{\bar\tau_3-\bar\tau_2}{\epsilon^2}\right]G\mathbf{x}(t-\rho\epsilon\bar\tau_1) \\
& + \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_3\bar\tau_1^2 - \bar\tau_3^2\bar\tau_1\right) - k_2\frac{\bar\tau_3^2-\bar\tau_1^2}{\epsilon} + 2\frac{\bar\tau_1-\bar\tau_3}{\epsilon^2}\right]G\mathbf{x}(t-\rho\epsilon\bar\tau_2) \\
& + \frac{k_3}{\rho^2\Delta}\left[k_1\left(\bar\tau_1\bar\tau_2^2 - \bar\tau_2\bar\tau_1^2\right) - k_2\frac{\bar\tau_1^2-\bar\tau_2^2}{\epsilon} + 2\frac{\bar\tau_2-\bar\tau_1}{\epsilon^2}\right]G\mathbf{x}(t-\rho\epsilon\bar\tau_3).
\end{aligned} \tag{11.6}$$

According to the scaling property presented in [220], a scaling of each delay term in the control law by a factor of ρ together with a scaling of $\frac{1}{\rho}$ in each corresponding gain does not change the stability of the system. Hence the closed-loop system (11.6) is stable if and only if the system (11.1) using control law (11.3) is stable. Thus, we can design ϵ, $0 < \bar{\tau}_1 < \bar{\tau}_2 < \bar{\tau}_3$ and k_i, $i = 1, 2, 3$ according to Theorem 11.1, and choose $\rho = \frac{\tau}{\epsilon \bar{\tau}_1}$ to achieve stabilization under delay. $\Box$

This theorem allows us to design the control law to stabilize a decentralized network with measurement delay. Essentially, by using the low-gain technique [220] and choosing the scaling factor ρ to match $\rho\epsilon\bar{\tau}_1$ with the measurement delay τ, we can absorb the measurement delay into the delay in the control law, and hence transform the closed-loop dynamics of the system with measurement delay into exactly the same form discussed in Section 11.2. A design that achieves stability can thus be implemented. The performance of the controller can be optimized by choosing $\bar{\tau}_i$ and k_i, $i = 1, 2, 3$ such that ρ is minimized.

Remark: The above result applies to systems with known measurement delay. However, the design can straightforwardly be adapted to plants with unknown but upper-bounded delays, which are also of common interest [219]. In particular, noting that the multiple-derivative controller achieves stability for all sufficiently large k_3, we see that the plant can be stabilized using the controller (5) with $\rho \geq \frac{\tau_{max}}{\epsilon \tau_1}$, where τ_{max} is the upper bound on the delay. We notice that this generalization requires that the open-loop $j\omega$-axis eigenvalues of the plant are in fact at the origin.

11.4.2 Controller Design for Networks with Input Saturation

In general, low gain techniques can help to resolve instability caused by input saturation [220]. In our setting, as we discussed in Section 11.4.1, we can simultaneously decrease the input gain and introduce more delay to the decentralized control law. With such scaling, we can guarantee that actuators do not saturate while leaving the closed-loop (linear) system's poles in the OLHP, and hence ensure stability. For the following decentralized system with input saturation:

$$\begin{aligned}\ddot{\mathbf{x}}(t) &= \sigma(\mathbf{u}(t)) \\ \mathbf{y}(t) &= G\mathbf{x}(t),\end{aligned} \tag{11.7}$$

we use the following control law:

$$\begin{aligned}\mathbf{u}(t) = \frac{k_3}{\rho^2\Delta}&\left[k_1\left(\bar{\tau}_2\bar{\tau}_3^2 - \bar{\tau}_2^2\bar{\tau}_3\right) - k_2\frac{\bar{\tau}_2^2 - \bar{\tau}_3^2}{\epsilon} + 2\frac{\bar{\tau}_3 - \bar{\tau}_2}{\epsilon^2}\right]\mathbf{y}(t - \rho\epsilon\bar{\tau}_1) \\ +\frac{k_3}{\rho^2\Delta}&\left[k_1\left(\bar{\tau}_3\bar{\tau}_1^2 - \bar{\tau}_3^2\bar{\tau}_1\right) - k_2\frac{\bar{\tau}_3^2 - \bar{\tau}_1^2}{\epsilon} + 2\frac{\bar{\tau}_1 - \bar{\tau}_3}{\epsilon^2}\right]\mathbf{y}(t - \rho\epsilon\bar{\tau}_2) \\ +\frac{k_3}{\rho^2\Delta}&\left[k_1\left(\bar{\tau}_1\bar{\tau}_2^2 - \bar{\tau}_2\bar{\tau}_1^2\right) - k_2\frac{\bar{\tau}_1^2 - \bar{\tau}_2^2}{\epsilon} + 2\frac{\bar{\tau}_2 - \bar{\tau}_1}{\epsilon^2}\right]\mathbf{y}(t - \rho\epsilon\bar{\tau}_3),\end{aligned} \tag{11.8}$$

where we choose ϵ, $\bar{\tau}_i$ and k_i , $i = 1, 2, 3$ in the same way as for the basic multiple-delay controller (i.e., assuming no saturation), and then choose ρ so that the actuator does not saturate. We formalize the design of the control law for a double integrator network with input saturation in Theorem 11.5:

Theorem 11.5. *Consider the double-integrator network with input saturation described in (11.7), where G is nonsingular. Assume that the corresponding (unsaturated) double-integrator network can be stabilized using the basic multiple-delay controller (11.3). The network can be semiglobally*[‡] *stabilized using the multiple-delay control law (Equation 11.8). Specifically, for any bounded set of initial conditions $\mathcal{W}$ and for fixed k_1, k_2 and k_3 satisfying the conditions in Theorem 11.1 and sufficiently small ϵ, we can choose ρ sufficiently large such that $\mathcal{W}$ is in the domain of attraction.*

Proof: According to the scaling property presented in [220], as long as the actuators never saturate, the response $x(t)$ for the scaled system is exactly a time-scaled version of the response to the unscaled system (Equation 11.3). We shall use this fact to prove stability. First, from stability

[‡] See [221] for a full introduction to semiglobal stabilization.

of the original unscaled system, we obtain that, for the given compact set of initial conditions, there is a bound on the absolute value of all state variables $x(t)$ over all $t \geq 0$. Thus, there is also a bound, say Γ, on the inputs over all $t \geq 0$. By choosing $\rho < \sqrt{\frac{1}{\Gamma}}$, invoking the time-scaling of the state, and noting that thus the input is scaled by a factor of Γ, we obtain that the inputs do not saturate. We thus automatically recover semi-global stability. $\Box$

12. THE DESIGN OF MULTI-LEAD-COMPENSATORS FOR STABILIZATION AND POLE PLACEMENT IN DOUBLE-INTEGRATOR NETWORKS

We study decentralized controller design for stabilization and pole-placement, in a network of autonomous agents with double-integrator internal dynamics and arbitrary observation topology. We show that a simple multi-lead-compensator architecture, in particular one in which each agent uses a derivative-approximation compensator with three memory elements, can achieve both stabilization and effective pole placement. The multi-lead-compensator design is practical for modern dynamical network applications, in that it subdivides actuation effort and complexity among the agents and in many cases is robust to agent failure.

12.1 Introduction

Through our efforts in studying control tasks in several modern dynamical networks [1,2,5,10,19, 25], we are convinced that network structure (i.e., the sensing/interaction interconnection structure among the agents) is critical in driving network dynamics, and hence must be exploited in controller design. Due to the crucial role played by the network structure, novel decentralized controller architectures that have the following two features are badly needed in dynamical network control applications: 1) control complexity and actuation are roughly equally contributed by all the agents, and 2) the controller can address control/algorithmic tasks in networks with very general sensing and/or interaction topologies. In this work, we introduce a novel decentralized dynamic controller that can guide network dynamics with arbitrary sensing structures—the very simple *multi-lead-compensator* controller. Specifically, we show that the design of the multi-lead-compensator—precisely, an LTI decentralized state-space controller with a small number of memory elements used in each channel, that approximates a multiple-derivative feedback—allows *stabilization* and *pole-placement* in an autonomous-agent network model with a general sensing structure.

To motivate the network stabilization and pole placement problem addressed here, let us briefly review two bodies of literature: the recent efforts on autonomous-agent network control, as well as historical efforts in decentralized control. The many recent works on autonomous-agent network control are fundamentally derived from prominent work by Chua and his colleges [226, 227] on network synchronization (see also the literature in the Physics community on synchronization, e.g. [147]). Chua in [226] gave the necessary and sufficient condition for a network with identical agents to achieve synchronization; and in [227] gave a graph interpretation of the condition when the network is diffusive, i.e., the sensing structure is described by a Laplacian matrix. Fax and Murray in [228] and Pogromsky in [229] brought forth control interpretations to the synchronization tasks in a diffusive network, and thus gave conditions for synchronization through control. We notice that network synchronization through control is closely connected to the network stabilization task, with only the distinction that network synchronization is concerned with the the stability of an invariant set while network stabilization is concerned with the stability of an equilibrium point. Thus very similar methods apply to both types of network problems, and in fact with this understanding, various other control tasks such as formation, agreement, alignment, tuning, consensus, and distributed partitioning [25, 207, 208, 230–232] have been addressed in essentially similar ways.

Attempts have also been made to design stabilizing controllers in the classical decentralized literature. In a seminal work [28], Wang and Davison give an implicit sufficient and necessary condition for the existence of a stabilizing time-invariant dynamic controller for a general decentralized system. Decentralized controller *wan-roy-saberi-stoorvogel-doubledelay-2008* for network stabilization, and similarly other network control tasks, is difficult due to the limitations imposed by the sensing structures. The book [66] and related works view the network interconnections as disturbances, and provide controller designs based on that assumption. In an alternate direction, building on [28], Corfmat and Morse studied stabilization of *complete* systems in [67]. In their work, applying non-dynamic controller to all but one channel of a complete system makes the system controllable and observable from the remaining channel, and then a dynamical controller is applied to this single channel to achieve stabilization and also pole-placement for the whole system.

Now let us emphasize the contribution of the multi-lead-compensator controller design with respect to both the autonomous-agent network control and the existing decentralized controller design literature. We notice that the approaches in the existing literature (e.g., [66, 67, 228]) do not serve our goal of finding controller architectures that are suitable for many modern network control applications, for two reasons. 1) These existing designs are impractical to implement: either the decentralized stabilization is achieved by hiding the contribution of network connections [66] (while in fact the network connections are critical in modern applications), or by making a single channel dominant in terms of actuation and complexity [67]. 2) Many studies, for instance those giving conditions for stabilization in the recent autonomous-agent-network literature, only work for a limited subclass of sensing structures such as ones specified by a Laplacian matrix [228, 229]. Hence, we are motivated to study the simple multi-lead-compensator architecture as an alternative. We will show that this controller can be designed for stabilization in networks with general sensing structures, using roughly equal actuations at each agent. Moreover, we stress here that the multi-lead-compensator architecture allows practical high performance design (pole placement), which is missing in the literature, as a further major contribution of this work. We focus here on the important class of double-integrator networks (networks with agents whose internal dynamics are double integrators), which are common models for autonomous-agent network applications (see e.g., [25, 232]). Our design for the double-integrator network is also illustrative of the design for much more general plants, which is applicable not only to autonomous-agent networks but ones with hardwired interconnections such as population dynamics [*].

Finally, let us discuss several recent works that have in fact addressed network topology and controller *design*, for high performance. Of interest, Boyd and his coworkers have used linear matrix inequality (LMI) techniques to optimize the Fiedler eigenvalue of a Laplacian matrix through design of an associated graph [80] (and hence to shape e.g. an associated single integrator network dynamics). In complement, building on a classical result of Fisher and Fuller [79], we have taken a structural approach to performance optimization through graph-edge and static decentralized controller design [1, 6, 7, 29]. This meshed control-theory and algebraic-graph-theory strategy has yielded designs for several families of network-interaction models and performance criteria, and has also permitted as to address the partial design problem. This chapter shows that even very simple dynamical controllers when properly used can give significant freedom in shaping a network's response (for instance, allowing pole placement).

[*] The more general case requires use of the special coordinate basis for linear systems, see our previous work on multiple-delay control for further details [215].

12.2 The Main Result

In this section, we first discuss the philosophy underlying our multi-lead-compensator design, then introduce the double-integrator network model formally, and finally demonstrate the controller design for stabilization and pole-placement.

Let us first illustrate the philosophy of the design. Our multi-lead-compensator design is based on 1) construction of a high-gain feedback of multiple derivatives up to degree 2 to place the close-loop poles in desired locations in the OLHP; and 2) approximation of the multiple-derivative controller with lead compensators. The philosophy is that for double integrator networks, high gain feedback of output derivatives up to the degree of 2 can permit each agent to recover its local state information [10], and hence permit pole placement and stabilization. We emphasize that the novelty of the design resides in the concept that one higher derivative (here, the second derivative, or in other words the derivative equal to the *relative degree* of the local plant) is being used in feedback. This feedback concept has not been used in the literature. Moreover, since derivative controllers are not implementable due to their unbounded high frequency gains, we implement the derivative controllers using lead compensators. The lead compensators produce poles close to those of the derivative controller and also extra poles far inside the OLHP, and hence achieve stabilization and high performance. In this controller architecture, the control actions are distributed among all agents rather than being centered at a single agent. This architecture also has the advantage that it is more robust to network failures/attacks, and hence is practical for modern network applications.

Next, let us introduce the **double-integrator network**, i.e. a decentralized system comprising n autonomous **agents** with double-integrator internal dynamics whose (scalar) observations are linear combinations of multiple agents' states. Precisely, we assume that each agent i has internal dynamics $\ddot{x}_i = u_i$, where we refer to x_i as the **position state** of agent i and $\dot{x}_i$ as the **velocity state**, and u_i is the input to agent i. For notational convenience, we define $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ and refer to it as the **full position state** of the network, and also define $\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$. We assume each agent i makes a scalar observation $y_i = \mathbf{g}_i^T\mathbf{x}$, i.e. that its observation is a linear combination of the position states of various agents. We find it convenient to stack the observations into a vector, i.e. $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$. We also stack vectors $\mathbf{g}_i^T$ to form a matrix $G = \begin{bmatrix} \mathbf{g}_1^T \\ \vdots \\ \mathbf{g}_n^T \end{bmatrix}$, which we refer to as the **topology matrix** since it captures the sensing/communication among the agents. In short, a double-integrator network comprises n agents that together have the dynamics

$$\begin{aligned} \ddot{\mathbf{x}} &= \mathbf{u} \qquad (12.1) \\ \mathbf{y} &= G\mathbf{x}, \end{aligned}$$

where each agent i makes an observation y_i and can set the input u_i.

Our goal is to design a linear decentralized controller (mappings from each y_i to u_i) to stabilize the double-integrator-network's dynamics. As a further step, we seek a pole-placement controller, i.e. one that achieves the classical controller design goal of placing the eigenvalues of the closed-loop dynamics at desirable locations. This decentralized controller design problem for the double-integrator network is widely applicable, see [25].

For the double-integrator network, it is necessary and sufficient for stabilization that G has full rank [25], regardless of whether centralized or decentralized control is considered and regardless of whether a linear or a non-linear time-varying (NLTV) controller is used (see also [28, 217]). Here, we will demonstrate not only stabilization but effective pole placement for arbitrary full rank G using the most limited of these schemes, namely an LTI state-space decentralized controller. In fact, we will show that a very simple controller—one that has third-order dynamics at each channel—suffices. The controller that we use may be viewed as applying first- and second-order lead compensation, or in other words approximating feedback of the first- and second- derivatives of the local observation, at each channel. This architecture, though very simple, is novel and specifically of use in the decentralized control context. Structurally, the novelty of the controller lies in that (approximate) feedback of derivatives up to and *including* the relative degree of the local dynamics is used. This is in contrast to the centralized setting, where controllers (whether designed using the observer-plus-state-feedback paradigm or in other ways) at their essence feed back derivatives up to one less than the plant's relative degree to achieve stabilization and pole placement [196].

We note that the dynamical controller design presented here enhances our existing work on stabilizing double-integrator networks [25], which is at its essence derived from Fisher and Fuller's classical result [79]. In fact, the proof of our main result in this chapter also relies on this result. Thus, for the reader's convenience, let us describe the classical result of Fisher and Fuller here, before introducing and proving our main result.

Theorem 12.1. (Fisher and Fuller) *Consider an $n \times n$ matrix A. If the matrix A has a nested sequence of n principal minors that all have full rank, then there exists a diagonal matrix K such that the eigenvalues of KA are in the open left half plane.*

The following theorem, our main result, formalizes that stabilization and pole-placement can be achieved generally in the double-integrator network using third-order compensators at each channel. The proof of the theorem makes explicit the compensator design. Specifically, we describe how to design a controller so that sets of n closed-loop eigenvalues can be placed arbitrarily near to two desired locations (closed under conjugation) in the complex plane, while the remaining $3n$ eigenvalues are placed arbitrarily far left in the complex plane. Here is a formal statement:

Theorem 12.2. *Consider a double-integrator network with arbitrary invertible graph-matrix G. Proper LTI compensators of order 3 can be applied at each channel, so as to place n eigenvalues each close to two desirable locations in the complex plane while driving the remaining $3n$ eigenvalues arbitrarily far left in the complex plane. Specifically, consider using a compensator at each agent i with transfer function $h_i(s) = k_o + \frac{k_1 s}{1+\epsilon\lambda_{fi}s} + \frac{k_2 s^2}{1+\epsilon s\lambda_{di}+\epsilon^2 s^2 \lambda_{zi}}$, and say that we wish to place n closed-loop eigenvalues at each of the roots of $s^2+\alpha s+\beta$. By choosing k_2 sufficiently large, $k_1=\alpha k_2$, and $k_o=\beta k_2$, and choosing λ_{fi}, λ_{di}, and λ_{zi} appropriately, n closed-loop eigenvalues can be placed arbitrarily close to each root of $s^2+\alpha s+\beta$ as ϵ is made small, while the remaining $3n$ eigenvalues can be moved arbitrarily far left in the complex plane (in particular, having order $\frac{1}{\epsilon}$).*

Proof:

The proof is in two steps. In the first step, we show that decentralized feedback of the observation and its first two derivatives can be used to to place the ($2n$) closed-loop eigenvalues arbitrarily near to two locations in the complex plane, and in fact there is a parameterized family of controllers of this form that suffice. In the second step, we use this result to construct proper third-order LTI compensators at each channel that achieve the pole-placement specification given in the theorem statement.

Step 1: Let us study the closed-loop eigenvalues of the system when the (decentralized) control law $\mathbf{u}(t) = k_0\mathbf{y}(t) + k_1\dot{\mathbf{y}}(t) + k_2\ddot{\mathbf{y}}(t)$, where $k_0 = \beta k_2$ and $k_1 = \alpha k_2$, is used. The state-space representation of the closed loop system in this case is $\dot{\mathbf{X}} = A_c\mathbf{X}$, where $\mathbf{X} = \begin{bmatrix} \dot{\mathbf{x}} \\ \mathbf{x} \end{bmatrix}$ and $A_c = \begin{bmatrix} (I - k_3G)^{-1}k_2k_3G & (I - k_3G)^{-1}k_1k_3G \\ I & 0 \end{bmatrix}$. Using the notation $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ for a right eigenvector of A_c, we have $A_c \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \lambda \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$, which implies that $X_1 = \lambda X_2$, and $\lambda(I - k_2G)^{-1}\alpha k_2GX_2 + (I - k_2G)^{-1}\beta k_2GX_2 = \lambda X_1$. The latter yields: $\lambda\alpha k_2GX_2 + \beta k_2GX_2 = \lambda^2(I - k_2G)X_2$, or $(\lambda\alpha k_2 + \beta k_2 + \lambda^2k_2)GX_2 = \lambda^2X_2$. This means that X_2 must be an eigenvector of G with, say, eigenvalue λ_i. In this notation, we have $(\lambda\alpha k_2 + \beta k_2 + \lambda^2k_2)\lambda_i = \lambda^2$, or $\lambda^2 - \frac{\alpha k_2\lambda_i}{1-k_2\lambda_i}\lambda - \frac{\beta k_2\lambda_i}{1-k_2\lambda_i} = 0$. Thus, the closed-loop eigenvalues are the roots of the characteristic equations $\lambda^2 - \frac{\alpha k_2\lambda_i}{1-k_2\lambda_i}\lambda - \frac{\beta k_2\lambda_i}{1-k_2\lambda_i}$, for $i = 1, \ldots, n$. Hence, by making k_2 sufficiently large, the coefficients of the characteristic equation can be made arbitrarily close to the coefficients of the quadratic equation $\lambda^2 + \alpha\lambda + \beta = 0$. From the continuous dependence of roots on parameters, the closed-loop poles thus come arbitrarily close to the roots of this characteristic equation, as desired. To summarize, when the presented multi-derivative compensator is used, the closed-loop eigenvalues can be made arbitrarily close to the two desired locations in the complex plane, for all k_2 sufficiently large.

Step 2: We now consider using a compensator at each channel i with transfer function $h_i(s) = k_o + \frac{k_1s}{1+\epsilon\lambda_{fi}s} + \frac{k_2s^2}{1+\epsilon s\lambda_{di}+\epsilon^2s^2\lambda_{zi}}$, where the gains k_0, k_1, and k_2 are those determined in Step 1, λ_{fi}, λ_{di}, and λ_{zi} are constants to be designed, and ϵ is a positive constant that will be designed sufficiently small after the other parameters have been designed. We note that this controller requires at most three memory elements at each channel to implement.

Substituting for the controllers' transfer functions, one immediately find the closed-loop characteristic polynomial. In particular, the closed-loop system's poles are values s such that $Q(s) = (I+\epsilon s\Lambda_f)(I-k_2G+\epsilon s\Lambda_d+\epsilon^2s^2\Lambda_z)s^2-(I+\epsilon s\Lambda_d+\epsilon^2s^2\Lambda_z)k_1Gs-(I+\epsilon s\Lambda_f)(I+\epsilon s\Lambda_d+\epsilon^2s^2\Lambda_z)k_0G$ lose rank, where Λ_f, Λ_d, and Λ_z are diagonal matrices with ith diagonal entry given by λ_{fi}, λ_{di}, and λ_{zi}, respectively. We notice that the closed-loop system has $5n$ poles (counting multiplicities).

To continue, we note that $Q(s)$ can be written as $Q(s) = s^2I - k_0G - sk_1G - s^2k_2G + \epsilon M_1(s) + \epsilon^2 M_2(s)$, where $M_1(t)$ and $M_2(s)$ do not depend on ϵ. Let us first consider the $2n$ values s for which $s^2I - k_0G - sk_1G - s^2k_2G$ loses rank. We note that these are precisely the closed-loop poles when the derivative-based controller is used, and so these values of s are in two groups of n, arbitrarily near to the two desired pole locations. It follows easily from perturbation arguments that, thus, n poles of the closed-loop system upon lead-compensator control (values s such that $Q(s)$ loses rank) are arbitrarily close to each desired pole location.

What remains to be shown is that the remaining poles are order $\frac{1}{\epsilon}$ and indeed can be placed in the left-half-plane. To see this, let us rewrite the Laplace-domain expression in terms of $\overline{s} = \epsilon s$. Doing so, we recover that $R(\overline{s}) = \epsilon^2 Q(\overline{s}) = \overline{s}^2(I+\Lambda_f\overline{s})(I-k_2G+\overline{s}\Lambda_d+\overline{s}^2\Lambda_z)+\epsilon N_1(\overline{s})+\epsilon^2N_2(\overline{s})$. To characterize the values $\overline{s}$ such that $R(\overline{s})$ and hence $Q(\overline{s})$ lose rank, let us first consider $T(\overline{s}) = \overline{s}^2(I+\Lambda_f\overline{s})(I-k_2G+\overline{s}\Lambda_d+\overline{s}^2\Lambda_z)$. We recognize that $T(\overline{s})$ loses rank at $\overline{s}=0$ with multiplicity $2n$, as well as at the $3n$ values $\overline{s}$ such that $(I+\Lambda_f\overline{s})(I-k_2G+\overline{s}\Lambda_d+\overline{s}^2\Lambda_z)$ loses rank. We note that these $3n$ values are non-zero as long as $I-k_2G$ is made full rank (which we shall shortly guarantee), and we choose Λ_f, Λ_d, and Λ_z full rank and $k_2 \neq 0$, as we shall do. In this case, we see immediately from perturbation arguments that the polynomial $R(\overline{s})$ loses rank at $2n$ values $\overline{s}$ that approach the origin as ϵ is made small, as well as at $3n$ other values $\overline{s}$ that approach the $3n$ non-zero points in the complex plane where $(I+\Lambda_f\overline{s})(I-k_2G+\overline{s}\Lambda_d+\overline{s}^2\Lambda_z)$ loses rank, as ϵ is made small. Rewriting all these values in terms of s rather than $\overline{s}$, we see that the closed-loop system has $2n$ poles that are close to the origin in that they do not grow as fast as $\theta(\frac{1}{\epsilon})$ (and which we have already characterized to be close to two desired locations in the complex plane), as well as $3n$ poles of order $\frac{1}{\epsilon}$ if the poles of $(I + \Lambda_f)(I - k_2 G + \bar{s}\Lambda_d + \bar{s}^2\Lambda_z)$ are nonzero (as we will show shortly).

Finally, let us construct the controller so that the values $\overline{s}$ for which $(I + \Lambda_f\overline{s})(I - k_2G + \overline{s}\Lambda_d + \overline{s}^2\Lambda_z)$ loses rank are all in the OLHP, and hence complete the proof. Clearly, n of these values are the $\overline{s}$ such that $(I + \Lambda_f\overline{s})$ loses rank. We can make these values negative and real by choosing each λ_{fi}, and hence Λ_F, positive and real.

Next, let us consider the $2n$ values $\overline{s}$ such that $(I - k_2G + \overline{s}\Lambda_d + \overline{s}^2\Lambda_z)$ loses rank. Since we have assumed Λ_z is full rank, we can equivalently find $\overline{s}$ such that $\overline{s}^2 I + \Lambda_z^{-1}\Lambda_d\overline{s} + \Lambda_z^{-1}(I - k_2G)$ loses rank. To continue, we note that all principal minors of $I - k_2G$ are full rank for all k_2 except those in a particular finite set, i.e. for all k_2 except those that are inverses of eigenvalues of the principal minors of G. Hence, for any design with k_2 large enough, $I - k_2G$ has a nested sequence of principal minors of full rank. Using any such design, we thus can choose Λ_z to place the eigenvalues of $\Lambda_z^{-1}(I - k_2G)$ at positive real values, according to the classical result of Fisher and Fuller (quoted as Theorem 12.1 above). Finally, let us choose $\Lambda_d = \Lambda_z^{-1}$. In this case, we see from simple eigenanalysis that the values $\overline{s}$ for which rank is lost are the solutions of the n scalar equations $\overline{s}^2 + \overline{s} + \lambda_i$, where each λ_i is an eigenvalue of $\Lambda_z^{-1}(I - k_2G)$. Thus, we obtain that all solutions $\overline{s}$ are in the OLHP. $\square$

We have thus demonstrated a multi-lead-compensator decentralized controller design for an arbitrary double-integrator network, which achieves stabilization as well a certain *group* pole placement. Let us stress that this group pole-placement capability gives us wide freedom to shape the dynamical response (in terms of settling and robustness properties), including by guaranteeing phase margin in the design through an inverse optimality argument (see [10]). It is worth noting that further design of the $\theta(1)$ poles is possible through consideration of heterogeneous derivative controller, however this further effort does not provide much further improvement in design performance and so we omit the details.

The multi-lead-compensator introduced here is promising for modern network applications because it is tailored to distribute controller effort and exploit the network topology (in particular by obtaining the local state at each agent through a combination of uniform feedback control and derivative estimation). Let us conclude the discussion of the multi-lead-compensator by discussing several conceptualizations and extensions of the design:

1) Although we have presented the design assuming each agent makes a single observation for convenience, the result automatically generalizes to the case where each agent may make multiple observations. In particular, whenever stabilization is possible (see [25] for conditions derived from [28]), the above pole-placement design can be applied as follows. First, the multiple observations made by the agent should be combined linearly, with each observation weighted by a randomly chosen constant. It follows automatically that this reduced (single-input, single-observation) double-integrator network can be stabilized (with probability 1), and hence the design technique introduced above can be applied.

2) The design that we have presented can be interpreted as comprising an estimator and a state-feedback controller. Specifically, if the pure derivative controller is used (with k_2 large), the agents can be viewed as immediately obtaining their local state (in particular, by rearranging their initial conditions in such a way that the second-derivative estimate and hence local state estimate are precise); thus, state feedback control can be used. In practice, such an

immediate estimation/rearrangement of initial conditions is not implementable. Instead, the lead compensator design achieves estimation (in part through rearrangement of the state) at a faster time scale than the state feedback response, but not immediately. We note that such use of fast observers is classical, and so our design naturally fits the estimation-plus-state-feedback paradigm. However, the design is fundamentally different from the traditional one, in that the state estimation is only possible when the feedback is in force—that is, the estimation and control tasks are NOT decoupled.

3) In dynamical network stabilization tasks, robustness to agent failure is an important concern. For instance, in an autonomous vehicle network, if the failure of a single agent can spoil the stability of the entire network, the viability of a stabilization design really falls in question. Let us briefly expand on this particular robustness problem. In this case, by agent failure, we mean there exist certain agents that can not be observed by all other agents. Robustness in the presence of agent failure problem is concerned with whether the rest of the agents can still achieve stability using the original controller design. Mathematically, this is the problem of whether, if all rows and columns in the sensing structure associated with the failed agents are removed, stability in the reduced dimensional system remains. We notice that current literature on decentralized controller design does not address this important issue. For instance, the dominant channel design in [67] is highly sensitive to the failure of the dominant channel, due to the significant role played by this single channel in stabilization. In the contrast, since in our lead-compensator design, all agents contribute roughly equally to the stabilization task, this design appears to be more robust to agent failure. For broad classes of sensing structures, e.g., those for which $I - k_2G$ is strictly *D-stable* through a diagonal (sign-pattern) scaling, stability is maintained in the presence of any number of agent failures.

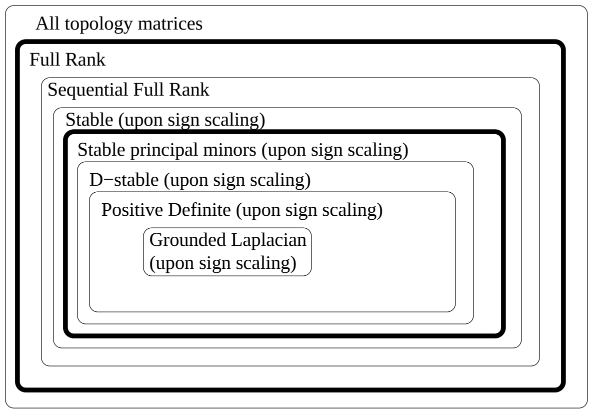

All topology matrices
Full Rank
Sequential Full Rank
Stable (upon sign scaling)
Stable principal minors (upon sign scaling)
D−stable (upon sign scaling)
Positive Definite (upon sign scaling)
Grounded Laplacian (upon sign scaling)

Fig. 12.1: We diagram several matrix classes that are of interest in representing a double integrator network's sensing topology. A multi-lead-compensator design is possible whenever the topology matrix is full rank, and a design that is robust to agent failures is possible if the topology matrix has stable principal minors to within a sign scaling (a scaling of each row by ± 1).

Clearly, subclasses of G that satisfy the above include strictly D-stable, positive definite, and grounded Laplacian topology matrices (see Figure 12.1 for a full illustration).

4) We stress that the value of the gain k_2 needed for stabilization or approximate pole placement is dependent on the structure of the graph matrix G: we refer the reader to our earlier work [10] for precise bounds on the gain. Because the actuation capability is distributed among the agents, and because the gain parameter can be tuned based on the network structure, the gain often need not be large to achieve stabilization and approximate pole placement.

13. SEMI-GLOBAL STABILIZATION OF DOUBLE-INTEGRATOR NETWORKS WITH ACTUATOR SATURATION

As a step toward designing controllers for general decentralized plants with actuator saturation, we develop a low-gain controller design for a network of saturating double-integrator agents with a general sensing topology. We obtain a practical low-gain design (in particular, one in which each agent uses a third-order LTI dynamic compensator) that permits semi-global stabilization for arbitrary sensing topologies.

13.1 Introduction

Actuator saturation nonlinearities are ubiquitous in control systems, and so controller design for linear time-invariant plants subject to actuator saturation has been extensively studied. A classical approach is to use dynamic linear-time-invariant controllers, for the purpose of semiglobal stabilization of the plant. Historically, this effort to design controllers for semi-global stabilization of LTI plants progressed in two steps. First, low gain designs for state feedback and in turn output feedback control were obtained, and secondly a low-and-high gain methodology was pursued for performance design under saturation (e.g. [198, 233]).

On the other hand, applications in such diverse areas as vehicle coordination, virus-spreading control, and air traffic management have led to much recent interest in decentralized control of dynamical networks [1, 2, 5, 10, 25]. These modern networks are very often made up of simple but highly constrained components that *must* coordinate through their network interactions to achieve control goals [5]. In that these modern dynamical networks' components are often constrained, designing controllers under actuator saturation constraints is important. We believe strongly that systematic low- and low-and-high gain controller design methodologies must be obtained for dynamical networks with actuator saturation, and, more generally, for decentralized LTI plants. This short note illustrates a low-gain design for a canonical and representative network model called the *double-integrator network*, namely a network of autonomous agents with double-integrator internal dynamics and actuator saturation that must coordinate using an arbitrary observation topology.

Let us briefly review the limited literature on controlling dynamical networks under saturation, to give context to this work. Stoorvogel and coworkers have given sufficient conditions for existence of decentralized controllers under saturation, in particular showing that stabilization is possible when the open-loop decentralized plant has 1) closed-left-half plane eigenvalues, 2) decentralized fixed modes only in the open left-half-plane, and 3) jw-axis axis eigenvalues with algebraic multiplicity 1 [222]. However, this study does not address the (obviously crucial) case where jw-axis eigenvalues are repeated, nor does it give a practical controller design. Meanwhile, our group has been able to give a sufficient condition on the observation topology under which a double-integrator network (which has repeated and defective $j\omega$-axis eigenvalues) can be stabilized under saturation, by focusing on a proportional-derivative control scheme [25]. As an alternative, for double-integrator networks with symmetric relative-position observations, Ren shows that a diffusive controller with a hyperbolic tangent nonlinearity can achieve a consensus or synchronization task [232]. Here, we present a systematic low-gain *dynamical controller design* for arbitrary double-integrator networks with actuator saturation. The design comprehensively addresses the problem of decentralized control in the presence of actuator saturation for the important double-integrator network model, and also focuses analysis and design efforts for more general decentralized plants with repeated $j\omega$-axis eigenvalues.

The remainder of the note is organized as follows. In Section 13.2, we review the double-integrator network model and revisit a previous design, for stabilization and pole-placement without saturation. In Section 13.3, we develop a low-gain scaling of the previous design, to obtain a controller that achieves stabilization under saturation.

13.2 Double-Integrator Network: Introduction and Preliminary Design

The **double-integrator network**, i.e. a network model comprising agents with double-integrator internal dynamics and general sensing topology, has been widely motivated and studied [10, 11, 25, 232]. Here, let us review the model, see [10, 11, 25] for further details. Formally, the **double-integrator network with saturation** comprises n autonomous **agents** with double-integrator internal dynamics with input saturation whose (scalar) observations are linear combinations of multiple agents' states. Precisely, we assume that each agent i has internal dynamics $\ddot{x}_i = \sigma(u_i)$, where we refer to x_i as the **position state** of agent i and $\dot{x}_i$ as the **velocity state**, u_i is the input to agent i, and $\sigma()$ is the standard saturation function. For notational convenience, we define $\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_n \end{bmatrix}^T$ and refer to it as the **full position state** of the network, and also define $\mathbf{u} = \begin{bmatrix} u_1 & \dots & u_n \end{bmatrix}^T$. We assume each agent i makes a scalar observation $y_i = \mathbf{g}_i^T \mathbf{x}$, i.e. that its observation is a linear combination of the position states of various agents. We find it convenient to stack the observations into a vector, i.e. $\mathbf{y} = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^T$. We also stack vectors $\mathbf{g}_i^T$ to form a matrix $G = \begin{bmatrix} \mathbf{g}_1 & \dots & \mathbf{g}_n \end{bmatrix}^T$, which we refer to as the **topology matrix** since it captures the sensing/communication among the agents. In short, a double-integrator network with saturation comprises n agents that together have the dynamics

$$\ddot{\mathbf{x}} = \sigma(\mathbf{u}) \tag{13.1}$$

$$\mathbf{y} = G\mathbf{x},$$

where each agent i makes an observation y_i and can set the input u_i, and the notation $\sigma()$ represents that each input is subject to the saturation nonlinearity. We also consider the case where input saturation is not in force, and so refer to this model as the **double-integrator network without saturation**.

Our goal is to design a linear decentralized controller (mappings from each y_i to u_i) to make the closed-loop dynamics of the double-integrator network with saturation *semi-globally* stable*.

Let us present some existing results on the control of double-integrator networks without saturation, as a preliminary step. For the double-integrator network without saturation, it is necessary and sufficient for decentralized stabilization using LTI compensation that G has full rank [10, 25]. Two promising dynamical control architectures have been established, using which controllers for stabilization and pole-placement can be designed for general classes of topology matrices: 1) an architecture in which delayed observations are deliberately used in feedback [10], and 2) a multi-lead-compensator architecture [11]. Controllers of both architectures that have been designed for stabilization/pole-placement can be adapted for semi-global stabilization under actuator saturation through a low-gain strategy. These methods yield practical *designs*, in that complexity and actuation are subdivided among the agents and robustness to agent failure is often established. Here, for brevity, let us focus on the multi-lead-compensator architecture since this approach yields finite

* Semi-global stabilization means that, for any bounded set around the origin in the state space, there exists a controller that achieves a region of attraction containing this set [198]. For systemswith saturation constraints, it is classical to pursue designs that achieve semi-global stabilization.

dimensional controllers as considered in the ongoing work on decentralized control of saturating plants [222], and permits design for arbitrary full-rank G. Before progressing to the design under saturation, let us quote the main result on multi-lead-compensator control of the double-integrator-network without saturation:

Theorem 13.1. *Consider a double-integrator network without saturation with arbitrary invertible graph-matrix G. Proper LTI compensators of order 3 can be applied at each channel, so as to place n eigenvalues each close to two desirable locations in the complex plane while driving the remaining $3n$ eigenvalues arbitrarily far left in the complex plane. Specifically, consider using a compensator at each agent i with transfer function $h_i(s) = k_o + \frac{k_1 s}{1+\epsilon\lambda_{fi}s} + \frac{k_2 s^2}{1+\epsilon s\lambda_{di}+\epsilon^2 s^2\lambda_{zi}}$, and say that we wish to place n closed-loop eigenvalues at each of the roots of $s^2+\alpha s+\beta$. By choosing k_2 sufficiently large, $k_1=\alpha k_2$, and $k_o=\beta k_2$, and choosing λ_{fi}, λ_{di}, and λ_{zi} appropriately, n closed-loop eigenvalues can be placed arbitrarily close to each root of $s^2+\alpha s+\beta$ as ϵ is made small, while the remaining $3n$ eigenvalues can be moved arbitrarily far left in the complex plane (in particular, having order $\frac{1}{\epsilon}$).*

We kindly ask the reader to see [11] for a proof of the above result, as well as exact specifications of the compensators' parameters. Analogous results for the deliberate-delay architecture, and further discussion on the benefits of these control schemes, can be found in [10, 14, 15].

For convenience, let us define $H(s) = [diag(h_i(s))]$. We note that $H(s)$ is the full controller's transfer function.

13.3 Stabilization under Saturation

In this section, we show that a multi-lead-compensator can semi-globally stabilize a double-integrator network under saturation, for an arbitrary graph matrix G. We stress again that the double-integrator network has $2n$ poles at the origin. Hence, the result is an expansion of the sufficient condition for the existence of a stabilizing controller provided in [222].

In the following Theorem 13.2, we show the design of the stabilizing multi-lead-compensator using a low-gain strategy. More specifically, we discuss a scaling of the compensator design presented in Theorem 13.1, that yields semi-global stabilization under actuator saturation. Here is the result:

Theorem 13.2. *Consider a double-integrator network with input saturation, and arbitrary invertible graph-matrix G. Say that compensators $h_i(s) = k_o + \frac{k_1 s}{1+\epsilon\lambda_{fi}s} + \frac{k_2 s^2}{1+\epsilon s\lambda_{di}+\epsilon^2 s^2\lambda_{zi}}$ have been designed for each agent i according to Theorem 1, so that stabilization is achieved in the double-integrator network without saturation. Then the network with input saturation can be semiglobally stabilized using at each agent i the parameterized family of proper LTI compensators $\hat{h}_{i,\hat{\epsilon}}(s) = k_o\hat{\epsilon}^2 + \frac{\hat{\epsilon}k_1 s}{1+\frac{\epsilon}{\hat{\epsilon}}\lambda_{fi}s} + \frac{k_2 s^2}{1+\frac{\epsilon}{\hat{\epsilon}}s\lambda_{di}+\frac{\epsilon^2}{\hat{\epsilon}^2}s^2\lambda_{zi}}$ That is, for any specified ball of plant and compensator initial conditions W, there exists $\hat{\epsilon}^*(W)$ such that, for all $0 < \hat{\epsilon} \leq \hat{\epsilon}^*(W)$, the compensator with the transfer function $\hat{h}_{i,\hat{\epsilon}}(s)$ at each channel achieves local stabilization of the origin and contains W in its domain of attraction.*

Proof: Without loss of generality, we can limit ourselves to examining the response from the plant initial conditions, since the component of the response due to the precompensator initial conditions can be made arbitrarily small through static pre- and post-scaling of the compensator at each agent, see e.g., [18].

In order to prove that the proposed controller semiglobally stabilizes the double-integrator network under input saturation, it suffices to show that, for any bounded set of initial conditions W the ∞-norm of the input $||\mathbf{u}(t)||$ where $t \geq 0$ remains upper bounded by 1, and also the dynamics without saturation are asymptotically stable. Thus, we can verify semi-global stabilization by showing that, for any bounded set of initial conditions W the norm of the input $||\mathbf{u}(t)||$ where $t \geq 0$

scales by $\hat{\epsilon}$ and further the closed-loop dynamics without actuator saturation are asymptotically stable. Let us prove this through a spectral argument.

Let us first consider applying the new scaled controller, i.e. the controller with transfer function $\widehat{H}(s) = [diag(h_i(s))]$. In the Laplace domain, the closed-loop dynamics of the double-integrator network ignoring saturation (when this scaled controller is used) are given by

$$\begin{aligned} & s^2\mathbf{X}(s) - s\mathbf{x}(0) - \dot{\mathbf{x}}(0) = k_o\hat{\epsilon}^2 G\mathbf{X}(s) + \\ & (I + \frac{\epsilon}{\hat{\epsilon}}\Lambda_f s)^{-1}\hat{\epsilon} s k_1 G\mathbf{X}(s) + (1 + \frac{\epsilon}{\hat{\epsilon}}\Lambda_d s + \frac{\epsilon^2}{\hat{\epsilon}^2}\Lambda_z s^2)^{-1} k_2 s^2 G\mathbf{X}(s), \end{aligned} \tag{13.2}$$

where Λ_f, Λ_d, and Λ_z are diagonal matrices whose ith diagonal entries are λ_{fi}, λ_{di}, and λ_{zi}, respectively.

Let us apply the change of variables $\hat{\epsilon}\bar{s} = s$, and scale both sides of Equation 13.3 by $\frac{1}{\hat{\epsilon}^2}$. The closed-loop system dynamics in terms of $\bar{s}$ in the Laplace domain becomes

$$\begin{aligned} & \bar{s}^2\mathbf{X}(\hat{\epsilon}\bar{s}) - \frac{1}{\hat{\epsilon}}\bar{s}\mathbf{x}(0) - \frac{1}{\hat{\epsilon}^2}\dot{\mathbf{x}}(0) = k_o G\mathbf{X}(\hat{\epsilon}\bar{s}) \\ & +(I + \epsilon\Lambda_f\bar{s})^{-1}\bar{s}k_1 G\mathbf{X}(\hat{\epsilon}\bar{s}) + (1 + \epsilon\Lambda_d\bar{s} + \epsilon^2\Lambda_z\bar{s}^2)^{-1}k_2\bar{s}^2 G\mathbf{X}(\hat{\epsilon}\bar{s}). \end{aligned} \tag{13.3}$$

From Equation 13.4, we get

$$\mathbf{X}(\hat{\epsilon}\bar{s}) = \tag{13.4}$$

$$(\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 + \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\frac{1}{\hat{\epsilon}}\bar{s}\mathbf{x}(0)$$

$$+(\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\frac{1}{\hat{\epsilon}^2}\dot{\mathbf{x}}(0)$$

$$= \frac{1}{\hat{\epsilon}^2}((\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\hat{\epsilon}\bar{s}\mathbf{x}(0)$$

$$+(\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\dot{\mathbf{x}}(0))$$

On the other hand, we note that using the original controller design with transfer function $H(\bar{s})$ and initial conditions $(\hat{\epsilon}\mathbf{x}(0), \dot{\mathbf{x}}(0))$ gives us (for the double integrator network ignoring saturation)

$$\overline{\mathbf{X}}(\bar{s}) = \tag{13.5}$$

$$((\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\hat{\epsilon}\bar{s}\mathbf{x}(0) +$$

$$(\bar{s}^2 I - k_o G - (I + \epsilon\Lambda_f \bar{s})^{-1}\bar{s}k_1 G - (1 + \epsilon\Lambda_d \bar{s} + \epsilon^2 \Lambda_z \bar{s}^2)^{-1} k_2 \bar{s}^2 G)^{-1}\dot{\mathbf{x}}(0)),$$

where we have used the notation $\overline{\mathbf{X}}(\bar{s})$ for the Laplace transform of the state to avoid confusion.

Equations (13.5) and (13.6) together inform us that $\hat{\epsilon}\mathbf{X}(\hat{\epsilon}\bar{s})$ is $\overline{\mathbf{X}}(\bar{s})$ scaled by $\frac{1}{\hat{\epsilon}}$. Hence, clearly, the double-integrator network's response adopting controller with transfer function $\hat{H}_{\hat{\epsilon}}(s)$ is the response of the double-integrator network using the controller with transfer function $H(\bar{s})$ with the initial conditions $(\hat{\epsilon}\mathbf{x}(0), \dot{\mathbf{x}}(0))$, scaled in amplitude by $\frac{1}{\hat{\epsilon}}$ and in frequency also by $\frac{1}{\hat{\epsilon}}$. Hence with just a little algebra, we see that the input norm $||\mathbf{u}(t)||$ is upper-bounded by a multiple of $\hat{\epsilon}$. Furthermore, we directly recover from Equations (13.5) and (13.6) that the closed-loop poles scale with $\hat{\epsilon}$ upon use of the scaled controller, and so asymptotic stability is maintained. Thus, semi-global stabilization is achieved.

In the above theorem, we have shown that a low-gain scaling of a (decentralized) multi-lead-compensator stabilizes a double-integrator network under saturation, for an arbitrary full-rank topology matrix G. Thus, we have fully addressed design of low-gain decentralized controllers for the double-integrator network with saturation. Let us conclude with two remarks about the design:

1) Let us distinguish the approach to design taken here with the traditional approach for centralized systems with saturation [198, 233]. In the low-gain design for centralized plants, actuation capabilities are subdivided between the observer and state feedback. In contrast, the double-integrator network requires integrated design of the entire dynamical controller, and hence we need a scaling of the full design to address control under saturation.

2) It is an open question as to whether decentralized plants with repeated $j\omega$-axis eigenvalues (and with all eigenvalues in the CLHP and all decentralized fixed modes in the OLHP) are amenable to semi-global stabilization. This first result shows that there is some promise for achieving semi-global stabilization broadly.

PART IV: TOOLS

Just as a child needs a village to grow up, a new scientific methodology requires a village of supporting tools to mature. Our efforts on dynamical network control and design—both those presented in this thesis, and those soon to come—require a range of new control-theoretic methods, in such domains as analysis of neutral-type delay systems and design of a linear system's invariant zero structure. Part IV catalogs the various supporting control-theoretic tools that we have developed.

Part IV is organized as follows. Chapter 14 studies the delay implementation of derivative controllers up to one less than the relative degree of a SISO plant. Chapter 15 extends the study to MIMO plants, and also studies the the delay implementation of derivative controllers whose orders are equal to the relative degree of a plant. Chapter 16 gives a thorough comparison of retarded-type delay systems with neutral-type delay systems. Chapter 17 and 18 are concerned with manipulating system zeros. Chapter 17 presents a precompensator design that cancels all OLHP zeros of a general MIMO LTI plant, and Chapter 18 presents a pre- + and post- + feedforward compensator that places the transmission zeros of a stabilizable and detectable MIMO LTI plant at arbitrary locations. Finally, Chapter 19 gives an alternative approach to designing stabilizing compensators for saturating LTI plants using a precompensator plus static-output-feedback architecture.

14. ON MULTIPLE-DELAY STATIC OUTPUT FEEDBACK STABILIZATION OF LTI PLANTS

We develop a control methodology for linear time-invariant plants that uses multiple delayed observations in feedback. Using the Special Coordinate Basis (SCB), we show that multiple-delay controllers can always be designed to stabilize minimum-phase plants, and identify a class of non-minimum-phase plants that can be stabilized using these controllers.

14.1 Introduction

Control systems subject to delays have been extensively studied (see e.g. the textbook [234]). Recently, several researchers have sought to *design* controllers that use *multiple delayed observations*, with the motivation that properly-designed delays can in some special cases act to stabilize a system [220, 235, 236]. Fundamentally, these multiple-delay controllers are constructed by first designing controllers that use *derivatives* of the output, and then approximating these derivatives using delay-differences. Specifically, Niculescu and coworkers have addressed multiple-delay pole-placement controller design for plants that are chains of integrators, and have also established that certain unstable plants cannot be stabilized with multiple delays [220,235]. Independently, the article [236] has pursued multiple-delay control for minimum-phase plants with relative degree 1 and 2, in particular proving stability in the special case where the Markov parameters are all positive. In this chapter, we develop the multiple-delay controller methodology for general linear time invariant

(LTI) plants, using the Special Coordinate Basis (SCB) [205]. In particular, we demonstrate design of stabilizing multiple-delay controllers for arbitrary minimum-phase plants (Section 14.2), and also discuss conditions for stabilizability for arbitrary LTI plants (Section 14.3).

While this note is narrowly focused on extending the existing results on multiple-delay control to general LTI plants, our broader motivation for pursuing delay-based feedback originates from our study of decentralized controller design [10, 215]. In this domain, multiple-delay controllers are highly effective, because 1) they naturally permit control in the presence of intrinsic observation/actuation delays, which are commonplace in modern dynamical networks; and 2) they are often amenable to decentralized implementation, unlike traditional observer-based control strategies. We kindly ask the reader to see or efforts on decentralized controller design for details [10, 215].

14.2 Minimum-Phase Plants

We first study multiple-delay control of square-invertible minimum phase plants, and then address the non-invertible case. In particular, let us first consider multiple-delay control of a plant

$$\begin{aligned}\dot{\mathbf{x}} &= A\mathbf{x} + B\mathbf{u}\\ \dot{\mathbf{y}} &= C\mathbf{x},\end{aligned} \tag{14.1}$$

that is minimum phase (i.e., has finite invariant zeros in the OLHP) and square invertible, with $\mathbf{x} \in R^n$, and $\mathbf{y}, \mathbf{u} \in R^{m_d}$.

Theorem 14.1. *The minimum-phase square-invertible plant (14.1) can be stabilized by a multiple-delay output feedback controller of the form $\mathbf{u}(t) = \sum_{i=1}^{M} K_i y(t - \bar{\tau}_i)$, where $0 \leq \bar{\tau}_1 < \bar{\tau}_2 < ... < \bar{\tau}_M$ and where the K_i are of appropriate dimension. Moreover, the required number of delayed observations M is equal to the maximum order among the infinite zeros of the plant.*

Proof: We prove the theorem by first showing that a multiple-derivative controller stabilizes the plant, and then invoking an equivalence between multiple-derivative control and multiple-delay control. From the SCB ([205]), it follows immediately that there exist invertible linear transformations Γ_x, Γ_y, and Γ_u such that $\begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_d \end{bmatrix} = \Gamma_x \mathbf{x}$, $\mathbf{y}_d = \Gamma_y \mathbf{y}$, and $\mathbf{u}_d = \Gamma_u \mathbf{u}$, satisfy

$$\begin{aligned}
&\dot{\mathbf{x}}_a = A_{aa}\mathbf{x}_a + L_{ad}\mathbf{y}_d \\
&\dot{\mathbf{x}}_i = A_{q_i}\mathbf{x}_i + L_{id}\mathbf{y}_d + B_{q_i}[u_i + E_{ia}\mathbf{x}_a + \sum_{j=1}^{m_d} E_{ij}\mathbf{x}_j], \\
&y_i = C_{q_i}\mathbf{x}_i,
\end{aligned} \tag{14.2}$$

for $i = 1, \ldots, m_d$, where $\mathbf{y}_d^T = \begin{bmatrix} y_1 & \ldots & y_{m_d} \end{bmatrix}$ for scalars y_i $(i \in 1, \ldots, m_d)$, $\mathbf{x}_d^T = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_{m_d} \end{bmatrix}$, $\mathbf{u}_d^T = \begin{bmatrix} u_1 & \ldots & u_{m_d} \end{bmatrix}$ for scalars u_i $(i \in 1, \ldots, m_d)$, $\mathbf{x}_i \in R^{q_i}$, $A_{q_i} = \begin{bmatrix} 0 & I_{q_i-1} \\ 0 & 0 \end{bmatrix}$, $B_{q_i} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}$, $C_{q_i} = \begin{bmatrix} 1 & \mathbf{0} \end{bmatrix}$, and further structure in the SCB parameters is described in e.g. [205]. Here, we notice that $\mathbf{x}_a$ is the state of the finite invariant zero dynamics, while $\mathbf{x}_i$, $i = 1, \ldots, m_d$ comprise the states associated with the infinite-zero structure. Now let us construct a vector $\tilde{\mathbf{y}}$ comprising each y_i and its first $q_i - 1$ derivatives. From the SCB representation, it automatically follows that $\tilde{\mathbf{y}}$ is related to the state associated with the infinite-zero structure, $\mathbf{x}_d$, by an invertible transformation; thus, static feedback of $\mathbf{x}_d$ can be achieved through static feedback of $\tilde{\mathbf{y}}$, and hence through appropriate static feedback of $\mathbf{y}$ and its derivatives up to order $\max_i q_i - 1$. The existence and design of a stabilizing multiple-derivative controller then follows immediately from the asymptotic time-scale and eigenstructure assignment (ATEA) methodology [195, 196, 237]. Specifically, one sees that a high-gain static feedback of the state associated with the infinite-zero dynamics can be used to 1) place a closed-loop pole arbitrarily near to each of the plant's finite invariant zeros, and 2)

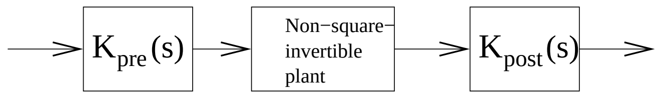


Fig. 14.1: A minimum-phase non-square-invertible plant can be made minimum-phase square-invertible using (in general dynamic) pre- and post-compensation. We can thus develop multiple-delay controllers even in the non-square-invertible case.

drive the remaining eigenvalues arbitrarily far left in the complex plane along desired time scales. Thus, we recover that a controller of the form $\mathbf{u}(t) = \sum_{i=1}^{M} k_i \mathbf{y}^{(i-1)}(t)$, where the k_i are matrices of appropriate dimension and $M = \max_i q_i$, can stabilize the plant.

Second, from Lemma 14.3 (see Appendix), we immediately recover that a controller of the form $\mathbf{u}(t) = \sum_{i=1}^{M} K_i y(t - \epsilon\tau_i)$, where $0 \leq \tau_1 < \tau_2 < ... < \tau_M$ and ϵ is a small positive constant, can stabilize the plant. We thus recover that the result of the theorem, choosing $\bar{\tau}_i = \epsilon\tau_i$. $\Box$

Let us now consider non-square-invertible minimum-phase plants. In this case, a multiple-delay controller can be designed through squaring-down followed by application of the above result for square-invertible plants. Specifically, an (in general dynamic) open-loop precompensator and postcompensator can be applied to make the plant square-invertible and minimum phase [238], see Figure 14.1. In turn, the multiple-delay controller design can be applied. We stress here that in many cases static pre- and post-compensation can be used, in which case the form of the controller is exactly as in Theorem 14.1. We also note that the pre- and post-compensation do not change the infinite zero structure of the plant (see [238]), so the required number of delays is identical to the number required in the square-invertible case.

14.3 Non-Minimum Phase Plants

In essence, multiple-delay controllers use delayed observations to estimate output derivatives. The SCB formulation above clarifies that, while these output derivatives partially identify the system state, they do *not* identify the *zero dynamics* of the plant. We thus expect that non-minimum-phase plants will not generally be stabilizable by multiple-derivative or multiple-delay controllers. Let us first give two examples, one of a non-minimum phase plant that can be stabilized using multiple-delay control and one of a non-minimum-phase plant that cannot be stabilized by any multiple-derivative linear controller (and hence also cannot be stabilized by a controller that approximates derivatives using delays). After presenting the examples, we will clarify that the problem of stabilizing a general plant with multiple delays (or derivatives) can be equivalenced with a static controller design problem, and hence the wide literature on static stabilization [239] can be applied.

Example 1: The non-minimum phase SISO plant with transfer function $H(s) = \frac{s-1}{s^2(s+10)^2}$ can be stabilized using a multiple-delay feedback controller. To verify, notice that a negative feedback controller that uses two derivatives of the output, namely $u(t) = \frac{d^2y}{dt^2} + 2\frac{dy}{dt} + y$, stabilizes the plant. We thus can construct a three-delay stabilizing controller. □

Example 2: A SISO plant with transfer function $H(s) = \frac{(s-1)^3}{s^4}$ cannot be stabilized by any multiple-derivative feedback controller, i.e. any controller of the form $u = \sum_{i=0}^{N} \alpha_i y^{(i)}$, for any N. To see this, notice that the controller's transfer function is a degree-N polynomial, say $p(s)$. Let us first consider the case that $p(s)$ has no zeros at the origin. In this case, the characteristic polynomial of the closed-loop is easily seen to be $s^4 + (s-1)^3 p(s)$. Noticing that $(s-1)^3 p(s)$ is a polynomial with three positive real roots, its coefficients have at least three sign changes according to Descartes' classical rule of signs. Thus, the coefficients of the closed-loop characteristic polynomial

$s^4+(s-1)^3p(s)$ change signs at least once, and so all roots cannot be in the OLHP. In the case where $p(s)$ has roots at the origin, the closed-loop characteristic polynomial takes the form $s^j+(s-1)^3\widehat{p}(s)$ where $j=0,1,2,3$ and $\widehat{p}(s)=\frac{p(s)}{s^{4-j}}$. Thus, by exactly the same argument, we see that not all roots of the characteristic polynomial are in the OLHP, and stability cannot be achieved. □

This example shows that multiple-derivative controllers, and so multiple-delay controllers based on approximating derivatives with delay differences, cannot always be used to stabilize non-minimum-phase plants. More fundamentally, we conjecture that multiple-delay controllers essentially approximate multiple-derivative ones, and so cannot be used to stabilize some non-minimum-phase plants.

Finally, we seek to distinguish general LTI plants that can and cannot be stabilized by multiple-delay controllers. In fact, we can straightforwardly pose this classification problem as a static stabilizability question. Let us present this equivalence for SISO plants (for simplicity).

Theorem 14.2. *Consider a SISO LTI plant $\dot{\mathbf{x}}=A\mathbf{x}+\mathbf{b}u$, $y=\mathbf{c}x$, with n poles and m zeros. Let us construct a SIMO plant with the same state equation, but with the output $\tilde{\mathbf{y}}=\begin{bmatrix} y & y^{(1)} & \dots & y^{(n-m-1)}\end{bmatrix}^T$. If this virtual SIMO plant can be stabilized using static linear feedback, then the SISO plant can be stabilized using using an $(n-m)$-delay controller.*

The proof of this theorem is immediate: static stabilizability of the SIMO plant implies stabilizability of the SISO plant using feedback that is a linear combination of the first $n-m-1$ derivatives of the output, and in turn stabilizability using multiple-delay control. We note that only $n-m-1$ derivatives (equivalently, $n-m$ delays) are considered, since higher derivatives of the output involve the input and hence cannot always be computed/approximated in practice using multiple-delay approximations.

The theorem generalizes automatically to the MIMO case. Specifically, upon transformation to the SCB coordinates (14.2), it is automatically clear that the output can be appended with the first $q_i - 1$ derivatives of each y_i (without involving the input directly), and so we can pose the multiple-derivative controller design as a static controller design problem for this appended system. We thus recover that multiple-delay controllers can be designed whenever an appended static controller design problem can be solved.

Remark: We note that that the system with extended output as given above has the same spectrum and blocking-zero structure as the original plant. It follows immediately that a necessary condition for the multiple-derivative control and hence our multiple-delay control is that the parity interlacing property holds, e.g. [239, 240].

Conclusion

We have demonstrated that multiple-delay controllers can be used to stabilize arbitrary minimum-phase plants, and have given conditions for stabilization of general LTI plants using multiple-delay controllers. These results significantly broaden the class of plants for which multiple-delay control is possible.

Appendix

We show that a multiple-delay controller can be designed to stabilize an LTI plant whenever a multiple-derivative controller of the form given in Theorem 14.1 can be used to stabilize the plant. Our proof is essentially based on the Razhumikjim-Lyapunov theory (which connects the time-evolution of a Lyapunov function to that of an interval-maximum *functional* [234]), though for clarity we work directly with an interval-maximum functional rather invoking the theory. Specifically, we prove stability using a quadratic functional $C(t) = \max_{\tau \in [0, M\epsilon\tau_M]} x(t-\tau)Px(t-\tau)$ where τ_M is the maximum delay used in the multiple-delay controller. Our argument generalizes that of

Michiels and Roose [241].

Fundamentally, the equivalence between multiple-derivative control and multiple-delay control is evident: because we have the freedom to choose the delays arbitrarily small, the closed-loop trajectory upon multiple-delay control can be made to approximate that achieved by multiple-derivative control (at least as long as the feedback does not include delayed derivatives of the state itself, see e.g. [224] for discussion of these neutral delay-differential equations). In turn the functional can be shown to be non-increasing and attractive. This result is formalized below.

Lemma 14.3. *Consider a SISO LTI plant with relative degree M that can be stabilized by the controller $u = \sum_{i=1}^{M} k_i y^{(i-1)}(t)$. Then the controller $\mathbf{u}(t) = \sum_{i=1}^{M} K_i y(t - \epsilon\tau_i)$, where $K_i = \sum_{j=i}^{M} \frac{(j-1)! k_j d_{Q(j)_{i,j}}}{\epsilon^{j-1} det(Q(j))}$, $Q(j) = \begin{bmatrix} 1 & \tau_1 & ... & \tau_1^{j-1} \\ 1 & \tau_2 & ... & \tau_2^{j-1} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \tau_j & ... & \tau_j^{j-1} \end{bmatrix}$, and $d_{Q(j)_{i,j}}$ is the (i,j)th minor of $Q(j)$, also stabilizes the plant for sufficiently small ϵ*[*].

Proof: We will prove stability using a Lyapunov functional of the form $V(t) = max_{s\in[0,M\epsilon\tau_M]}\mathbf{x}^T(t-s)P\mathbf{x}(t-s)$. Specifically, we shall show that $V(t)$ is not only non-increasing but also attractive to the origin, and hence we shall prove stability (see e.g. [234]). The proof is organized as follows: first, we formalize that the closed-loop dynamics can be viewed as a stable finite-state LTI dynamics plus a small approximation error. Using this formulation, we specify a Lyapunov functional that can be used to prove stability and attractivity.

First, let us express the closed-loop system dynamics when the multiple-delay controller is used in terms of the closed-loop dynamics upon multiple-derivative control, plus an error. For

[*] We note that the gains in the multiple-delay controller are based on approximating the observation $y(t)$ with a polynomial interpolations over the interval $[0, \epsilon\tau_M]$.

convenience, let us first define an extended output vector $\tilde{\mathbf{y}} = \begin{bmatrix} y & \dot{y} & \dots & y^{(M-1)} \end{bmatrix}^T$. In this notation, the input $u(t)$ when the multiple-delay controller is used can be written as the input when multiple-derivative control is used (which constitutes a static linear feedback of $\tilde{\mathbf{y}}$), plus a (small) correction term that captures the difference between multiple-delay-based approximation of output derivatives and the derivatives themselves. That is, $u(t) = \mathbf{k}^T\tilde{\mathbf{y}} + \mathbf{k}^T\tilde{\mathbf{y}}_{diff}$, where $\tilde{\mathbf{y}}_{diff} = \tilde{\mathbf{y}}_{app} - \tilde{\mathbf{y}}$, where $\tilde{\mathbf{y}}_{app} = \begin{bmatrix} y_{app} \\ \vdots \\ y_{app}^{(M-1)} \end{bmatrix}$, where $y_{app}^{(i-1)}$ is the approximation of the $i-1$st derivative of y using i delays as in the Lemma statement, and $\mathbf{k} = \begin{bmatrix} k_1 & ... & k_M \end{bmatrix}^T$. We notice that our approximation for the ith derivative ($y^{(i)}$) is constructed by interpolating the observation $y(t)$ at $i+1$ points on the interval $[t - \epsilon\tau_i, t]$. We note that, from the classical mean value theorem for divided differences, there exist $\theta_i \in [0, \epsilon\tau_i]$, $i \in 1, \dots, M$ such that $y^{(i-1)}(t) - y_{app}^{(i-1)}(t) = \epsilon\tau_i y^{(i)}(t - \theta_i)$; we shall use this fact subsequently to show that the functional is non-increasing and attractive[†].

Using the re-written observation vector, we can straightforwardly express the closed-loop dynamics as $\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}\mathbf{k}^T(\tilde{\mathbf{y}} + \tilde{\mathbf{y}}_{diff})$. Noting that $\tilde{\mathbf{y}} = \tilde{C}\mathbf{x}$ for appropriate $\tilde{C}$ (since, from the SCB, each of the first M derivatives of $y(t)$ can be written as linear combinations of $x(t)$), we obtain that $\dot{\mathbf{x}} = (A + \mathbf{b}\mathbf{k}^T\tilde{C})\mathbf{x} + \mathbf{b}\mathbf{k}^T\tilde{\mathbf{y}}_{diff}$. We note here that $\overline{A} = A + \mathbf{b}\mathbf{k}^T\tilde{C}$ is Hurwitz stable, and so there exists $P > 0$ and $Q > 0$ such that $\overline{A}^T P + P\overline{A} \leq -Q$.

Next, we prove stability of the closed-loop using the functional $V(t) = max_{s \in [0, M\epsilon\tau_M]}\mathbf{x}^T(t - s)P\mathbf{x}(t-s)$. We do so from first principles, in two steps: *1)* we show that $V(t)$ is a non-increasing

[†] We note that the parameters θ_i vary with time, i.e. the given approximation is correct for the particular time t. Our argument only requires consideration of the Lyapunov function at individual times, so we do not make the time-dependence explicit in our notation.

function of time, and *2)* we show that $V(t)$ approaches 0 (in fact exponentially) by bounding the times at which $V(t)$ is less than arbitrary fractions of its initial value.

To show that $V(t)$ is non-increasing, let us first show that if $V(\widehat{t}) = c$, then $V(t) \leq c$ for $t \geq \widehat{t}$, for any c (when ϵ is chosen sufficiently small). From the fact that $V(\widehat{t}) = c$, we know that $W(t) = \mathbf{x}^T(t)P\mathbf{x}(t)$ is less than or equal to c for $\widehat{t} - M\epsilon\tau_M \leq t \leq \widehat{t}$. From continuity of the solution, we thus know that there must be a particular time t such that $W(t) = c$ for the first time if c is to be exceeded, and further $W()$ must increase from less than c to greater than c at this time t. We prove this is impossible by showing that $\dot{W}$ is less than 0 for $W(t) = c$, and hence prove that $V(t) \leq c$ for $t \geq \widehat{t}$. Specifically, note that

$$\dot{W} = \mathbf{x}^T(\overline{A}^T P + P\overline{A})\mathbf{x} + \mathbf{x}^T P\mathbf{b}\mathbf{k}^T\tilde{\mathbf{y}}_{diff} + \tilde{\mathbf{y}}_{diff}^T\mathbf{k}\mathbf{b}^T P\mathbf{x} \leq -\mathbf{x}^T Q\mathbf{x} + 2|\mathbf{x}||P\mathbf{b}\mathbf{k}^T\tilde{\mathbf{y}}_{diff}|. \tag{14.3}$$

Substituting for $\tilde{\mathbf{y}}_{diff}$, we obtain that

$$\dot{W} \leq -\mathbf{x}^T Q\mathbf{x} + 2|\mathbf{x}||P\mathbf{b}\mathbf{k}^T| \begin{vmatrix} \epsilon\tau_1 y^{(1)}(t-\theta_1) \\ \epsilon\tau_2 y^{(2)}(t-\theta_2) \\ \vdots \\ \epsilon\tau_M y^{(M)}(t-\theta_M) \end{vmatrix}, \tag{14.4}$$

for some $\theta_1, \ldots, \theta_M \in [0, \epsilon\tau_M]$. Note that $\epsilon\tau_i y^{(i)}(t-\theta_i)$ are clearly bounded linearly with the norm of $\mathbf{x}(t-\theta_i)$ and with ϵ for $i = 1, \ldots, M_1$, since each of these derivatives is a linear function of the concurrent state. However, further effort is needed in bounding $\epsilon\tau_M y^{(M)}(t-\theta_M)$.

To continue, notice that this term can be rewritten as $\epsilon\tau_M C_{(M-1)}\dot{\mathbf{x}}(t-\theta_M)) = \epsilon\tau_M C_{(M-1)}(A\mathbf{x}(t-\theta_M) + \mathbf{b}\mathbf{k}^T\tilde{\mathbf{y}}_{app}(t-\theta_M))$, where $C_{(M-1)}$ describes the linear transformation from $\mathbf{x}$ to $y^{(M-1)}$. Notice that $\epsilon\tau_M C_{(M-1)}A\mathbf{x}(t-\theta_M)$ is guaranteed to be bounded by a function that is linear with ϵ and with the norm of $\mathbf{x}(t-\theta_M)$, but $\mathbf{b}\mathbf{k}^T\tilde{\mathbf{y}}_{app}(t-\theta_M))$ requires more work to bound since the approximation $\tilde{\mathbf{y}}_{app}(t-\theta_M)$ depends on ϵ. Let us thus study this term a bit further. In particular, note that $\tilde{\mathbf{y}}_{app}(t-\theta_M))$ can be rewritten as $\tilde{\mathbf{y}}_{app}(t-\theta_M) = \tilde{\mathbf{y}}(t-\theta_M)+\tilde{\mathbf{y}}_{diff}(t-\theta_M)$. The first term, $\tilde{\mathbf{y}}(t-\theta_M)$ is bounded. The second term, $\tilde{\mathbf{y}}_{diff}(t-\theta_M)$ can be approximated in the same way as $\tilde{\mathbf{y}}_{diff}(t)$, specifically as $\tilde{\mathbf{y}}_{diff}(t-\theta_M) = \begin{bmatrix} \epsilon\tau_1 y^{(1)}(t-\theta_M-\phi_1) \\ \epsilon\tau_2 y^{(2)}(t-\theta_M-\phi_2) \\ \vdots \\ \epsilon\tau_M y^{(M)}(t-\theta_M-\phi_M) \end{bmatrix}$, for some $\phi_1,\ldots,\phi_M$. Here, all terms are guaranteed to be bounded with respect to ϵ and the norm of $\mathbf{x}$ at the appropriate time, except the highest-order one. However, substituting this highest-order term into the expression for $\dot{W}$, we finally recover that the only (possibly) unbounded term has the form $\epsilon^2 q\mathbf{y}^{(M)}(t-\theta_M-\phi_M)$, where q is a fixed constant. Repeating this process $M-2$ further times, we finally recover that the only (possibly) unbounded term has the form $\epsilon^M r\mathbf{y}^{(M)}(t-\theta_{big})$, where $\theta_{big} < M\epsilon\tau_M$ and r is a fixed constant.

Finally, noting that $\mathbf{y}^{(M)} = C_{(M-1)}\dot{\mathbf{x}}$, we obtain that this term is $\epsilon^M rC_{(M-1)}\dot{\mathbf{x}}(t-\theta_{big}) = \epsilon^M rC_{(M-1)}(A\mathbf{x}(t-\theta_{big}) + \mathbf{b}u(t-\theta_{big}))$. The first of the two terms in the above expression is bounded with $||\mathbf{x}||$ and ϵ (in fact, ϵ^M). Meanwhile, from the expression for the multiple-delay controller, we see that $u(t-\theta_{big})$ can be bounded by $\frac{C}{\epsilon^{M-1}}$ for some positive constant C (for $\mathbf{x}$ in the given ball). Thus, we recover that $\epsilon^M rC_{(M-1)}A\mathbf{x}(t-\theta_{big}) + \mathbf{b}u(t-\theta_{big})$ is bounded by a linear function of ϵ and the norm of $\mathbf{x}$. Hence, we have proved that the perturbation of $\dot{W}$ from that upon use of a multiple-derivative controller (see Equation 14.3) can be bounded by a sum of terms that are each linear with ϵ, and with a norm of $\mathbf{x}$ at a time in $[t-M\epsilon\tau_M, t]$. In turn, we recover that $\dot{W} \leq -\lambda_{min}(Q)|\mathbf{x}|^2 + \epsilon L|\mathbf{x}|^2$, where the positive constant L (which does not depend on c) is not worthwhile to compute. By choosing $\epsilon < \frac{\lambda_{min}(Q)}{L}$, we can guarantee that the derivative of $W(t)$ is negative for $W(t) = c$, and hence $W(t)$ and in turn $V(t)$ do not exceed c. Since this statement holds for all c, we automatically recover that $V(t)$ is a non-increasing function of time.

We can straightforwardly extend the above argument to show that the functional $V(t)$ not only is non-increasing but in fact approaches 0. In particular, we can prove the following: if $V(\widehat{t}) = c$, then $V(\widehat{t} + 2\frac{\lambda_{max}(P)}{\lambda_{min}(Q)} + M\epsilon\tau_M) \leq \frac{c}{2}$, as long as we choose $\epsilon < \frac{\lambda_{min}(Q)}{4L}$ (where L is the positive constant described above). To prove this, simply note that by choosing ϵ in this way and using the fact that the norms of the delayed versions of $\mathbf{x}$ are bounded by $\frac{2\lambda_{max}(P)}{\lambda_{min}(P)}|\mathbf{x}(t)|$ while $W(t)$ is between $\frac{c}{2}$ and c, we guarantee that $\dot{W} \leq -\frac{1}{2}\lambda_{min}(Q)|\mathbf{x}|^2$. Thus, while $\frac{c}{2} \leq W(t) \leq c$, it is guaranteed that $W(t)$ decreases at a rate of at least $\frac{1}{2}\lambda_{min}(Q)|\mathbf{x}|^2 \geq \frac{1}{2}\frac{\lambda_{min}(Q)}{\lambda_{max}(P)}W(t) \geq \frac{1}{2}\frac{\lambda_{min}(Q)}{\lambda_{max}(P)}\frac{c}{2}$. We thus recover that $W(t)$ can remain between c and $\frac{c}{2}$ for a maximum time of $2\frac{\lambda_{max}(P)}{\lambda_{min}(Q)}$. Once $W(t)$ has dropped below $\frac{c}{2}$, it is clear from the fact that the derivative is negative for $W(t) \geq \frac{c}{2}$ that $V(t)$ cannot again exceed $\frac{c}{2}$. Thus, we recover the result above. Repeating the argument, we obtain that $V(t) \leq \frac{c}{2^n}$ for $t \geq \widehat{t} + 2n\frac{\lambda_{max}(P)}{\lambda_{min}(Q)} + nM\epsilon\tau_M$, and so we have proved asymptotic (and in fact exponential) convergence of the state to the origin.

We have thus proved that, if the state is upper-bounded by a constant c over the interval $[-M\epsilon\tau_M, 0]$, then it is bounded by c for all $t \geq 0$ and in fact converges exponentially to the origin. The only remaining step in proving stability (see e.g. [234]) is to show that boundedness over the shorter interval $[-\epsilon\tau_M, 0]$ yields boundedness and convergence. However, it is trivial to show that the shorter interval suffices by viewing the response over the longer interval as that of a finite-dimensional linear system with bounded input. □

Remark: In Lemma 14.3, we have approximated the ith derivative using degree-$(i+1)$ polynomial interpolation, for ease of presentation. Alternately, a single approximation of all the derivatives from a single M-point interpolation can be shown to achieve stability, using a similar argument. Thus, our result generalizes Niculescu's result for integrator chains [220] to general SISO plants. □

The result generalizes naturally to the MIMO case. We can approximate all the required derivatives of the the output (specifically, of linear combinations of output variables), as delay differences. Again, as we make the delays small, we find that the multiple-delay controller approximates the multiple-derivative controller more and more accurately, and hence an identical Lyapunov argument suffices to prove stability—the only difference is that the observation and its required derivatives are vectors (rather than scalars) that depend linearly on the state and its derivative. Since the highest derivative used by the controller is the maximum M among the orders of the infinite zeros minus 1, we recover that $M - 1 + 1 = M$ delays are needed.

A couple further notes about the multiple-delay approximation are worthwhile. First, the above argument can straightforwardly be extended to show that the Lyapunov exponent for the multiple-delay control can be made arbitrarily close to that for the multiple-derivative control. Second, it can be shown that the finite poles of the multiple-delay-controlled system approach the poles of the multiple-derivative-controlled system, while the additional poles move toward $-\infty$ as ϵ becomes small. A full treatment of this second point is deferred to future work. We also leave it to future work to select the delays $\tau_1, \dots, \tau_M$, so as to trade off accuracy in the derivative approximation (and hence in the settling response) with robustness to additive noise, see e.g. [242] for relevant analysis.

15. ON MULTIPLE-DELAY APPROXIMATIONS OF MULTIPLE-DERIVATIVE CONTROLLERS

We study approximation of multiple-derivative output feedback for linear time-invariant (LTI) plants using multiple-delay approximations. We obtain a condition on the plant and feedback that yields an equivalence between the closed-loop spectra for the approximate feedbacks and the desired multiple-derivative feedbacks. On the other hand, we use a scalar example to illustrate that multiple-delay approximations of sufficiently high derivatives may in some cases yield closed-loop spectra that differ greatly from the dynamics upon derivative feedback (for instance, containing many and very large right half plane poles), while in other cases replicating the derivative feedback perfectly. Finally, through understanding this dichotomy, we present a condition for stabilizing a SISO relative degree-1 plant when a delay implementation of the first-order derivative is used in the output feedback control law.

15.1 Introduction

Control of linear time-invariant plants at its essence requires feedback of the output's derivatives, and so control schemes explicitly or implicitly must obtain approximations of output derivatives (see e.g. [196]). Typically, finite-dimensional filters (for instance, lead compensators) are used to obtain output derivatives. However, in recent years, feedback controllers in which derivatives are approximated directly from current and delayed output samples have gained some prominence [13, 220, 236, 244–246]. Specifically, these *multiple delay controllers* have been of interest as alternatives to the typical finite-state controllers for several reasons, including: 1) the need for new signal-based control schemes in applications where the traditional observer design fails (such as decentralized and adaptive control applications), 2) the simplicity of approximating derivatives with delay-differences in some application areas, and 3) the intrinsic presence of delays in many modern control systems. Thus, modeling and analyzing feedback control systems that use delay approximations for derivatives, and in turn *designing* delay-approximation schemes in controls, is important.

Differential equations with delays, and more specifically the closed-loop dynamics of control systems subject to delay, have been extensively studied [219]. However, the systems studied here are distinct from those studied in the delay literature, in that *multiply-delayed outputs are deliberately being used to approximate output derivatives and hence to implement desirable feedbacks.* This deliberate use of delays engenders new analyses—namely, efforts to equivalence the performance of the delay-based controller with a true derivative feedback control. It also forces study of delay systems in the case where the delays are made small, as is needed for accurate approximation of derivatives using delayed outputs. To the best of our knowledge, a systematic treatment of deliberate-delay-based output-feedback controllers has not been given: several works have addressed design for particular plants or particular controllers. Of interest to us, integrator-chains and relative degree one and two plants with certain high frequency gain constraints have been addressed in [220, 236, 244]. Moreover, motivated by their study of stabilizing uncertain steady states using difference feedback, Kokame and Mori in [246] studied the stability when using difference counterparts in a feedback that is only involved with a first-order derivative. Our earlier work [13] expanded on these efforts by showing that deliberate-delay-based controllers can *stabilize* a large class of LTI plants, but did not address more complicated controls goals such as pole placement (which we will address here); this previous effort also only gave a detailed proof of results for the SISO case, while we will fully study the MIMO case here. We note that the studies [13, 220] of deliberate-delay-based output feedback controls only consider approximation of sufficiently low output derivatives, in particular ones that are less than the relative degree of the plant. Meanwhile, several researchers have recognized in the state-feedback arena that deliberate-delay approximations of some higher derivative feedbacks may fail, while stability is achieved upon approximation of other feedbacks [236, 246]. This motivates the systematic study of higher derivative feedbacks pursued here.

In this chapter, we further the study of deliberate-delay control of LTI plants. We first show, in Section 15.2, that the closed-loop spectrum of MIMO plants upon derivative feedback can be achieved asymptotically using deliberate-delay approximations, as long as the approximated derivatives are of sufficiently low order. In Section 15.3, we demonstrate through a scalar example the phenomenon that approximation of higher-derivative feedbacks can in some cases yield unexpected and undesirable response characteristics (including spectra with poles far in the right half plane), while replicating derivative control exactly in other cases. Based on this understanding, we also give a more general result on delay approximation of first-derivative output feedback control of relative-degree-1 plant. These characteristics of the closed-loop spectrum and response are similar in flavor to characteristics of neutral-type delay differential equations, but also have some significant distinctions.

15.2 A pole equivalence result

Here, we give conditions under which delay approximations of multiple-derivative controllers achieve the same closed-loop performance as derivative feedback, in the sense that the closed-loop eigenvalues upon delay control either approach those upon derivative control, or move arbitrarily far to the left in the complex plane. In particular, we find that delay-based and derivative-based controllers can be made arbitrarily close as long as the derivatives being approximated are of sufficiently low order, namely such that the closed-loop system under derivative feedback would be strictly proper. Derivative feedback of this form is well known to permit stabilization and pole placement for a large class of LTI plants (see the classical literature on asymptotic timescale eigenstructure assignment, or ATEA, design [196]), and so we see that delay-based approximations are apt for stabilization and pole placement.

Specifically, let us consider a MIMO LTI system:

$$\begin{aligned} \dot{x} &= A_0 x + B_0 u \\ y &= C_0 x \end{aligned} \tag{15.1}$$

with $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $y \in \mathbb{R}^p$. Say that a multiple-derivative feedback controller

$$u(t) = \sum_{i=1}^{n} K_i y^{(i-1)}(t), \tag{15.2}$$

where $K_i \in \mathbb{R}^{m \times p}$, (which has transfer function $F(s) = \sum_{i=1}^{n} K_i s^{i-1}$) can be designed for this system, so that the closed-loop transfer functions

$$C(sI - A)^{-1} B K_i s^{i-1} \tag{15.3}$$

are strictly proper for $i = 1, \dots, n$ and the zeros of

$$sI - A - BF(s)C$$

are in the open left half plane.

Remark: We stress that such multiple-derivative controllers can be designed for a wide class of LTI plants, including all minimum phase plants [13].

Let us consider approximating the derivative feedback using a multiple-delay scheme. In particular, we consider approximating $y^{(1)}, \ldots, y^{(n-1)}$ by developing a polynomial interpolation of the observation at times $t - \varepsilon\tau_1, \ldots, t - \varepsilon\tau_n$, and computing the derivatives from the interpolation (see [13, 220] for background). With a little algebra, we can specify the transfer function $F_\varepsilon(s)$ of this approximate feedback (which is parametrized on ε) as follows. First defining

$$L_{i,\varepsilon}(s) = \tfrac{(i-1)!}{(-\varepsilon)^{i-1}}\, e_i' \begin{pmatrix} 1 & \tau_1 & \cdots & \tau_1^{n-1} \\ 1 & \tau_2 & \cdots & \tau_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & \tau_n & \cdots & \tau_n^{n-1} \end{pmatrix}^{-1} \begin{pmatrix} e^{-\varepsilon\tau_1 s} \\ \vdots \\ e^{-\varepsilon\tau_n s} \end{pmatrix},$$

where $0 \le \tau_1 < \tau_2 < \cdots < \tau_M$ and e_i denotes the ith unit vector, we get

$$F_\varepsilon(s) = \sum_{i=1}^{n} K_i \begin{pmatrix} L_{i,\varepsilon}(s) & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & L_{i,\varepsilon}(s) \end{pmatrix} = \sum_{i=1}^{n} K_i \tilde{L}_{i,\varepsilon}(s). \tag{15.4}$$

We shall consider the closed-loop spectrum upon use of this approximate feedback control*, as a function of ε. We obtain the following main result:

Theorem 15.1. *For sufficiently small ε, the closed system upon application of the approximate feedback $F_\varepsilon(s)$ to the system* (15.1) *is also stable. In particular, n closed-loop poles approach the closed-loop poles when the true derivative feedback* (15.2) *is used, while the remaining (infinite number of) poles move arbitrarily far left in the complex plane as ε is decreased.*

* We stress that these control problems that we are addressing here is not simply a delay-independent stability problem, in that not only the delays but the parameters are changing with ε.

Proof: The proof consists of two steps. We will first show that the closed-loop system using the approximate feedback $F_\varepsilon(s)$ has exactly n poles in the right half plane $\operatorname{Re} s \geq -1/\varepsilon$. Then we will show that these n poles converge to the the n poles of the closed-loop using derivative feedback as $\varepsilon \to 0$.

Let us look at the resulting closed loop system:

$$sI - A_0 - B_0 F_\varepsilon(s) C_0 \tag{15.5}$$

and the associated zeros. We use the factorization:

$$C_0(sI - A_0)^{-1} B_0 = \frac{1}{q(s)} P(s)$$

where P is a polynomial matrix and $q(s) = \det(sI - A)$. Note that our earlier assumption guarantees that

$$P(s) K_i$$

is a polynomial of order less than or equal to $n - i$ for $i = 1, \ldots, n$. Next we note:

$$\begin{aligned} q(s)^{p-1} G_\varepsilon(s) &= q(s)^{p-1} \det\left(sI - A_0 - B_0 F_\varepsilon(s) C_0\right) \\ &= q(s)^p \det\big(I - (sI - A_0)^{-1} B_0 F_\varepsilon(s) C_0\big) \\ &= q(s)^p \det\big(I - C_0(sI - A_0)^{-1} B_0 F_\varepsilon(s)\big) \\ &= \det\left(\, q(s) I - P(s) F_\varepsilon(s)\,\right). \end{aligned}$$

The next step is to apply a scaling:

$$\bar{s} = \varepsilon s$$

and obtain:

$$\begin{aligned}
\bar{G}_\varepsilon(\bar{s}) &= \varepsilon^{pn} q(s)^{p-1} G_\varepsilon(s) \\
&= \varepsilon^{pn} q(s)^{p-1} \det\left(sI - A_0 - B_0 F_\varepsilon(s) C_0\right) \\
&= \det\left(\, \varepsilon^n q(s) I - \varepsilon^n P(s) F_\varepsilon(s)\,\right) \\
&= \det\left(\varepsilon^n q(s) I - \varepsilon^n \sum_{i=1}^n P(s) K_i \tilde{L}_{i,\varepsilon}(s)\right) \\
&= \det\left(\bar{q}_\varepsilon(\bar{s}) I - \sum_{i=1}^n \bar{P}_{i,\varepsilon}(s) \bar{L}_{i,\varepsilon}(s)\right)
\end{aligned}$$

where

$$\bar{q}_\varepsilon(\bar{s}) = \varepsilon^n q(\varepsilon^{-1}\bar{s}),$$

$$\bar{P}_{i,\varepsilon}(\bar{s}) = \varepsilon^{n-i} P(\varepsilon^{-1}\bar{s}) K_i,$$

and

$$\bar{L}_{i,\varepsilon}(\bar{s}) = \varepsilon^i \tilde{L}_{i,\varepsilon}(\varepsilon^{-1}\bar{s}).$$

We note that $\bar{q}_\varepsilon(\bar{s})$, $\bar{P}_{i,\varepsilon}(\bar{s})$ and $\bar{L}_{i,\varepsilon}(\bar{s})$ all depend polynomially on ε and converge to $\bar{s}^n$, $P_{i,0}\bar{s}^{n-i}$ and 0 respectively when $\varepsilon \to 0$ where $P_{i,0}$ is some constant matrix (which might be zero). Hence $\bar{G}_0(\bar{s}) = \bar{s}^{pn}$ has exactly pn zeros at the origin.

Next we note that for $\operatorname{Re}\bar{s} \geq -1$ there exists constants N_1, N_2 and N_3 such that $\|\bar{L}_{i,\varepsilon}(\bar{s})\| \leq \varepsilon N_1$, $\|\bar{P}_{i,\varepsilon}(\bar{s})\| \leq N_2 |\bar{s}|^{n-i}$ and $|\bar{s}^n - \bar{q}_\varepsilon(\bar{s})| \leq \varepsilon N_3 |\bar{s}|^{n-1}$. But then

$$\left(\bar{q}_\varepsilon(\bar{s}) I - \sum_{i=1}^n \bar{P}_{i,\varepsilon}(s) \bar{L}_{i,\varepsilon}(s)\right) v = 0$$

for some v with $\|v\| = 1$ implies:

$$\bar{s}^n v = \left(\bar{s}^n - \bar{q}_\varepsilon(\bar{s})\right) v + \sum_{i=1}^n \bar{P}_{i,\varepsilon}(s) \bar{L}_{i,\varepsilon}(s) v.$$

But then

$$|\bar{s}|^n \leq \varepsilon N_3 |\bar{s}|^{n-1} + \sum_{i=1}^{n} \varepsilon N_1 N_2 |\bar{s}|^{n-i}$$

which clearly implies that for ε small enough we must have that $|\bar{s}| \leq 1$. Hence for ε small enough, all zeros of $\bar{G}_\varepsilon(\bar{s})$ in $\operatorname{Re}\bar{s} \geq -1$ are also inside the unit circle. Next, an application of Hurwitz's theorem, see [243], implies that $\bar{G}_\varepsilon(\bar{s})$ has exactly np eigenvalues inside the unit circle for ε small enough and which converge to zero as $\varepsilon \to 0$. Hence we know that $\bar{G}_\varepsilon(\bar{s})$ for ε small enough has exactly np eigenvalues in $\operatorname{Re} s \geq -1$. This implies immediately that $G_\varepsilon(s)$ has, again for small ε, exactly n eigenvalues in $\operatorname{Re} s \geq -1/\varepsilon$.

Next we choose a region $\mathcal{K}$ such that it contains all zeros of

$$sI - A_0 - B_0 \left(\sum_{i=1}^{n} K_i s^{i-1} \right) C_0. \tag{15.6}$$

We will show that $G_\varepsilon(s)$ has exactly n zeros in $\mathcal{K}$ which converge to the zeros of (15.6) as ε converges to zero. As already indicated in [220]

$$F_\varepsilon(s) = \sum_{i=1}^{n} K_i s^{i-1} + O(\varepsilon s).$$

in the region $\mathcal{K}$. But this implies that inside the compact region $\mathcal{K}$, (15.5) converges uniformly to (15.6) as $\varepsilon \to 0$. This implies that $G_\varepsilon(s)$ has, for small enough ε, n zeros inside $\mathcal{K}$ which converge to the zeros of (15.6) as $\varepsilon \to 0$. Since $G_\varepsilon(s)$ has exactly n zeros in the region $\operatorname{Re} s \geq -1/\varepsilon$, we find that $G_\varepsilon(s)$ has no other zeros outside $\mathcal{K}$ in the region $\operatorname{Re} s \geq -1/\varepsilon$. This clearly implies that n zeros converge to the zeros of (15.6) while the remaining zeros approach $-\infty$. This completes the proof of stability and the associated convergence of eigenvalues.

Let us make one note about the above result. For SISO plants and MIMO uniform rank plants, the condition given in the above theorem reduces to the condition that the derivatives being approximated are strictly lower in order than the (common) relative degree of the plant; the theorem indicates that approximation of these derivatives can be used successfully in feedback control. More generally, the theorem indicates that output derivatives which can be written as linear functions of the concurrent state are amenable to multiple-delay approximation in feedback. We note that, often, many fewer than n delays may be needed for approximation; a careful delineation of the number of delays needed requires the *special coordinate basis* for linear systems, see the study of asymptotic timescale and eigenstructure assignment (ATEA) in [196].

15.3 Anatomy of higher derivative approximations: scalar examples

In Section 15.2, we have shown that multiple-delay approximations to multiple-derivative controllers achieve equivalent performance in the limit of small delay, as long as the highest output derivative approximated is less than the relative degree of the plant (i.e., the closed loop transfer functions are strictly proper). In several domains including decentralized and adaptive control, one encounters the problem that multiple-derivative controllers involving derivatives up to and including the relative degree of the plant (i.e., controllers that make the closed-loop non-strictly proper) are needed (see e.g., [10]). We are thus motivated to understand the closed-loop dynamics when multiple-delay approximations for derivatives of order equal to the relative degree are used.

In this section, we expose the complexity of the dynamics when delay approximations for derivatives of order equal to the relative degree are used. It turns out in this case that the delay approximation does not always yield the dynamics achieved using the derivative controller, even in the limit of small delay. We show this interesting phenomenon using a canonical (scalar) example.

We consider the scalar system

$$\dot{x}(t) = u(t), \tag{15.7}$$

and consider delay approximation of the following stabilizing derivative-based controller

$$u = ax(t) + b\dot{x}(t), \tag{15.8}$$

where the gains a and b need to satisfy either $a > 0$ and $b > 1$ or $a < 0$ and $b < 1$ for stability. Specifically, we approximate the derivative term $\dot{x}(t)$ in the controller (15.8) as $\frac{x(t)-x(t-\Delta)}{\Delta}$, where Δ is a small time delay, in which case the delay-based controller is

$$u = \frac{b + a\Delta}{\Delta} x(t) - \frac{bx(t-\Delta)}{\Delta}. \tag{15.9}$$

We notice here that the derivative being approximated is in fact that of the full state (in this case, a scalar), and so the special output feedback structure used in Section 15.2 to prove equivalence is not in force here.

In Section 15.3.1, we show that delay approximations to stabilizing derivative controllers can lead to instabilities that are not present if derivative control is used, and compare this effect with the instabilities observed in delay-differential equations of neutral type. Next, Section 15.3.2 identifies a class of derivative feedbacks (for the scalar plant) that are amenable to approximation by multiple-delay controllers. Using this insight into the dichotomy of approximation performance, we give conditions in Section 15.4 on feedback controllers for relative-degree-1 plant with higher derivative feedbacks that allow approximation.

15.3.1 Instabilities caused by delay approximations

Here we use a simple first-order system (15.7) to show that the delay approximation of certain derivative controllers may introduce ORHP poles. We find the pole locations of the closed-loop delay-based system, and present an interesting phenomenon: the unstable pole becomes larger with more accurate approximations. Let us present several results describing this possible instability caused by approximation.

The delay-based controller (15.9) may introduce ORHP poles, although the corresponding derivative controller stabilizes the system when $a > 0$ and $b > 1$. When (15.9) is used, the closed loop poles satisfy

$$s = a + \frac{b}{\Delta}(1 - e^{-\Delta s}). \tag{15.10}$$

As an example, when $a = 1$, $b = 10$, and $\Delta = 0.001$, the closed-loop system has a real root at $s \approx 10000$. With the decrease of Δ, the real pole moves further to the right.

This peculiar phenomenon motivates us to further characterize the precise locations and number of the ORHP poles. Interestingly, we can constrain the pole locations of the system upon use of the delay-based approximation to a contour in the complex plane. We show this result in Lemma 15.2.

Lemma 15.2. *Consider the first order system* (15.7) *and a stabilizing derivative-based controller* (15.8). *The poles of the closed-loop system using the corresponding delay-based implementation of the controller* (15.9) *are located on the contour*

$$(r - a - \frac{b}{\Delta})^2 + q^2 = \frac{b^2}{\Delta^2}e^{-2\Delta r}.$$

Proof: Rewrite (15.10) with $s = r + qj$, where r is the real part of a pole, and q is the imaginary part. A little bit of algebra leads to

$$r = a + \frac{b}{\Delta} - \frac{b}{\Delta}e^{-\Delta r}\cos(\Delta q) \tag{15.11}$$

$$q = \frac{b}{\Delta}e^{-\Delta r}\sin(\Delta q) \tag{15.12}$$

By combining equations (15.11) and (15.12) and noticing $\sin(\Delta q)^2 + \cos(\Delta q)^2 = 1$, we obtain the condition.

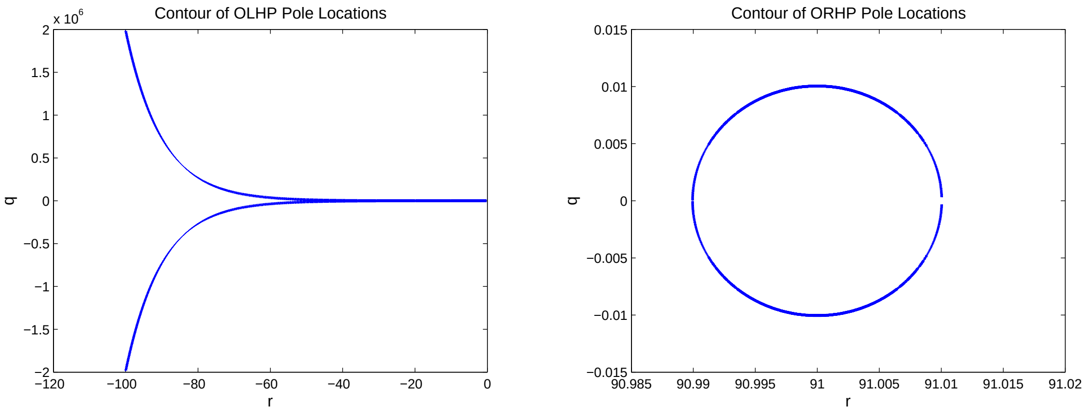


Fig. 15.1: The contour of poles when $a = 1$, $b = 9$ and $\Delta = 0.1$. a) the contour of the OLHP poles; b) the contour of the ORHP poles.

This result shows that the poles reside on a contour centered at $(a + \frac{b}{\Delta}, 0)$ with varying radius $\frac{b}{\Delta}e^{-\Delta r}$. An example of the contour when $a = 1$, $b = 9$ and $\Delta = 0.1$ is shown in Figure 15.1. When the delay Δ is small, Δr roughly equals the constant b, hence the shape of the ORHP contour is roughly a circle. Also, with the decrease of Δ, the ORHP contour shifts to the right with the radius roughly scaled with $\frac{1}{\Delta}$. On the other hand, the shape of the contour in the OLHP is determined by the dominant exponential term. Of interest to us, we note that the OLHP contour scales to the left as Δ decreases. Meanwhile, the OLHP contour changes significantly with r, due to the exponential term. We again stress that, as the delay Δ is decreased, the unstable pole becomes larger and larger.

The delay-based controller (15.9) for any fixed set of parameters is formally one of retarded type, and hence has a finite number of ORHP poles. We notice that if we instead use a controller with a *delayed-derivative* approximation for the scalar plant, i.e.

$$u(t) = ax(t) + b\dot{x}(t - \Delta) \tag{15.13}$$

the resulting closed-loop system is a neutral delay-differential equation. One characteristic of neutral delay-differential equations (see e.g. [219, 247]) is the presence of an "infinite root chain," i.e. of an infinite number of ORHP eigenvalues all with real part located in an interval. Besides the difference, the controllers in (15.13) and (15.9) may result in unstable dynamics that do not resemble that using the derivative controller (15.8). It is quite interesting to observe when using delay approximation, how this number of poles depends on the system parameters, to see whether the dynamics are similar to those of a neutral type system.

Let us describe the dependence of the number of ORHP poles on the delay Δ and on the gain b.

Lemma 15.3. *Consider the first order system* (15.7) *and a stabilizing derivative-based controller* (15.8) *with $a > 0$ and $b > 1$. The corresponding delay-based implementation of the controller* (15.9) *introduces a finite number of poles in the ORHP. The number of ORHP poles can grow with the decrease of the time delay Δ, but remains bounded. However, the number of ORHP poles grows unboundedly with b.*

Proof: First, we notice that when $a > 0$ and $b > 1$, the derivative-based controller (15.8) stabilizes the system. Moreover, from (15.12) we have $|\Delta q| \leq be^{-\Delta r}$.

Next, we recall the well-known property that the closed-loop system using the delay-based controller—which is a retarded delay-differential equation for any fixed Δ—has only a finite number of ORHP poles (see e.g., [219].)

Now let us count the number of ORHP poles. When $q = 0$, the closed-loop system has a single real pole at $(a + \frac{b}{\Delta} - \frac{b}{\Delta}e^{-\Delta r}, 0)$. When $q \neq 0$, combining equations (15.12) and (15.11) and eliminating r from the expression, we obtain

$$\Delta q e^{a\Delta + b} = b \sin(\Delta q) e^{\Delta q \cot(\Delta q)}. \tag{15.14}$$

By introducing $q' = \Delta q$, we can see that the right side of (15.14) is simply an oscillating function of q' with varying amplitude within the bound $|q'| < be^{-\Delta r}$, and the left side is a monotonic function of q' with the scaling factor $e^{a\Delta + b}$. From consideration of the oscillating function on the right side with the observation that $\cot(\Delta q)$ is unbounded, one automatically sees that the number of solutions to (15.14) and hence the number of complex ORHP poles is between $\frac{be^{-\Delta r}}{\pi} - 2$ and $\frac{be^{-\Delta r}}{\pi} + 2$. Hence as Δ decreases, one sees that the number of solutions to (15.14) may increase but always below the bound $\frac{b}{2\pi} + 1$. We also automatically recover from the lower bound that the number grows unboundedly with b.

We notice that our delay-based approximation, though formally yielding a retarded differential equation, has some resemblance to delayed-derivative approximation for large b, in terms of having highly unstable dynamics and a large number of ORHP poles.

The analysis of the example in this section demonstrates that the delay approximation when the stabilizing derivative controller has highest derivative term equal to the relative degree may cause instability. Such a system is different in terms of dynamics from both the delayed-derivative system and the delay approximation when we use one less degree in the derivative controller. The former system is a neutral system that has chains of infinite number of roots in certain right half planes. Meanwhile, the later controller guarantees that the dynamics using the approximation resembles the dynamics using the derivative controller. When a derivative equal to the relative degree is used in the control law, the resulting system remains retarded but may not represent the dynamics using the derivative controller anymore, and in fact show certain neutral type behavior, as we see when b becomes large. This peculiar dynamics is cause by the fact that delay is used to approximate a quantity that is not part of the state of the system.

15.3.2 A Pole Placement Result for Some Approximations

We have shown that, unfortunately, delay approximations to derivative feedbacks of order equal to the relative degree of a plant can fail. Luckily, only some such approximations cause instability. In fact, in many cases, depending on the parameters of the stabilizing derivative controller, certain delay approximations can be shown to achieve equivalence to the derivative feedback in a pole-placement sense, as the delay Δ is made small. Here, we shall demonstrate this pole-equivalence result for the canonical scalar system (15.7) with $a < 0$ and $b < 0$ in (15.9). This simple example provides us with a means for implementing multiple-derivative controllers for decentralized systems. That is, through smart design of the stabilizing derivative controller, stability can be maintained using delay approximation.

Although this delay feedback only differs from the one considered in Section 15.3.1 in its sign, the resulting closed-loop dynamics are stable, and in fact the spectrum become equivalent to that in (15.8) as the delay Δ becomes small. Precisely, the following result holds:

Theorem 15.4. *Consider the poles of the first-order system* (15.7) *using delay-feedback control* (15.9) *where $a < 0$ and $b < 0$. As $\Delta \to 0$, one pole approaches $\frac{a}{1-b}$, i.e. the single pole of the closed-loop system using derivative feedback control* (15.8). *The remaining poles have real parts approaching $-\infty$.*

Proof: We notice that this closed-loop dynamics, which represents a negative derivative feedback to a single-integrator plant, has a single stable pole, at $\frac{a}{1-b}$. As in Section 15.3.1, we can limit the poles to the contour given in Lemma 15.2. For $a < 0$ and $b < 0$, it is immediate that this contour is located entirely within the OLHP, so stability of the closed-loop follows.

What remains to be shown is that one pole approaches $\frac{a}{1-b}$, while the remaining poles move arbitrarily far left. To prove this, let us first show that one pole can be placed arbitrarily close to $\frac{a}{1-b}$ by choosing Δ small, while the remaining poles must be outside a circle centered the origin with radius increasing unboundedly as Δ becomes small.

We notice that the characteristic equation of the delay-feedback system is given by $s = a + \frac{bs}{1+q(s)}$, where $q(s) = \frac{s\Delta}{1-e^{-s\Delta}} - 1$, and we have written the differential equation in this form to highlight that the delay approximation performs a low-pass filtering. Notice that, roughly, $q(s) \approx 0$ for $|s| < \frac{1}{\Delta}$ and $q(s) \approx s\Delta$ for $|s| > \frac{1}{\Delta}$.

Rearranging, we obtain that the characteristic polynomial is $(s-a)q(s) + (1-b)s - a = 0$. Using this form, we shall prove that all poles within a circle in the complex plane whose radius increases unboundedly as Δ decreases can only lie within a small ball around $\frac{a}{1-b}$. In particular, consider a small ball of radius $\varepsilon > 0$ around the point $\frac{a}{1-b}$. For all s outside this ball, notice that $|(1-b)s - a| \geq |(1-b)|\varepsilon$. Now consider a (large) circle of radius f. For any f, it is clear that the maximum value of $|q(s)|$ within this circle can be made arbitrarily small by choosing Δ sufficiently small; more specifically, it is easily seen that $|q(s)|$ within this circle can be upper bounded by $Kf\Delta$, for some positive constant K. In turn, we find that $|(s-a)q(s)|$ within the circle is upper bounded by $Kf(f+a)\Delta$. Thus, by choosing $\Delta < \frac{(1+b)\varepsilon}{Kf(f+a)}$, we can guarantee that $|(1+b)s+a| > |(s-a)q(s)|$ for all s in the circle of radius f, for any f. Choosing Δ in this way, we guarantee that all roots of the characteristic equation within the circle of radius f must be within the ball of radius ε around $\frac{a}{1-b}$. By considering the characteristic equation, we can trivially check that there is indeed precisely one (real) pole within the ball, and that this pole has multiplicity 1.

We can use the contour on which the poles must be located to complete the proof. Notice that, for particular $a < 0$ and $b < 0$, the contour is located in the OLHP and further that as Δ is decreased the real part of the point on the contour for each possible imaginary value decreases (becomes more negative). This observation, together with the fact that Δ can be selected to exclude poles from inside a circle of arbitrary radius f (except the one near $\frac{a}{1-b}$), shows that the remaining poles can be moved arbitrarily far left in the complex plane by choosing Δ sufficiently small.

The equivalence of the delay-feedback approximation with the derivative-based controller for $a < 0$ and $b < 0$ is heartening, because it suggests that some derivative controls of order equal to the relative degree of a plant can be implemented using multiple-delay approximations.

15.4 Designing derivative-approximation controllers for relative degree 1 *plants*

The anatomy of higher-derivative approximations introduced in the above sections shows that instabilities may result when approximating some derivative feedbacks, while approximations of other derivative feedbacks match the derivative feedback's performance. That is, approximation is possible when some feedback gains are applied to the higher-derivative terms, but not when other gains are used. This understanding motivates study of *which* feedback gains allow for use of higher-derivative approximations, for more general LTI plants. Here, let us present a first result in this direction. Specifically, let us show how delay-approximation controllers can be used to stabilize SISO LTI plants with relative degree 1. Formally, consider a SISO plant

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= cx \end{aligned} \tag{15.15}$$

that has relative degree 1, i.e. for which cb is nonzero. Assume that there exist a derivative feedback $u = k_1 y + k_2 \dot{y}$ which stabilizes this system. Let us consider control of this plant by approximating this derivative feedback by

$$k_1 y(t) + k_2 \frac{y(t) - y(t - \varepsilon\tau)}{\varepsilon\tau}, \tag{15.16}$$

for any $\tau > 0$. We note that the derivative of the output to be approximated in feedback is equal to the relative degree of the plant, which equivalently means that 1) the closed-loop transfer function is not strictly proper and that 2) the derivative $\dot{y}$ is not simply a linear function of the concurrent state. Thus, we expect this feedback might not be easily approximated when certain gains k_1 and k_2 are used. The following theorem gives conditions under which the deliberate-delay approximation matches the derivative-feedback control.

Theorem 15.5. *Consider the system* (15.15) *with relative degree* 1 *and the delay-based feedback* (15.16). *Provided k_2 and cb have opposite sign, the closed loop system is asymptotically stable for all small enough $\varepsilon > 0$.*

Proof: The closed loop poles of the approximating feedback are the zeros of:

$$\det\left(sI - A - b\left[k_1 + k_2\tfrac{1}{\varepsilon\tau}\left(1 - e^{-\varepsilon\tau s}\right)\right]c\right)$$

which can be rewritten as:

$$g_\varepsilon(s) = \det\left(q(s) - p(s)\left[k_1 + k_2\tfrac{1}{\varepsilon\tau}\left(1 - e^{-\varepsilon\tau s}\right)\right]\right)$$

where

$$c(sI - A)^{-1}b = \frac{p(s)}{q(s)}$$

where $q(s)$ is a monic polynomial of order n and $p(s)$ is a polynomial of order $n-1$ whose leading coefficient equals cb. The next step is to apply a scaling:

$$\bar{s} = \varepsilon s$$

and obtain:

$$\begin{aligned} g_\varepsilon(\bar{s}) &= \varepsilon^n g_\varepsilon(s) \\ &= \det\left(\bar{q}_\varepsilon(\bar{s}) - \bar{p}_\varepsilon(\bar{s})\left[k_1\varepsilon + k_2\tfrac{1}{\tau}\left(1 - e^{\tau\bar{s}}\right)\right]\right) \end{aligned}$$

where

$$\bar{q}_\varepsilon(\bar{s}) = \varepsilon^n q(\varepsilon^{-1}\bar{s}),$$

$$\bar{p}_\varepsilon(\bar{s}) = \varepsilon^{n-1} p(\varepsilon^{-1}\bar{s}),$$

We note that $\bar{q}_\varepsilon(\bar{s})$ and $\bar{p}_\varepsilon(\bar{s})$ depend polynomially on ε and converge to $\bar{s}^n$ and $p_0\bar{s}^{n-1}$ respectively when $\varepsilon \to 0$ where p_0 is some constant matrix (which might be zero).

Next we note that for $\operatorname{Re}\bar{s} \geq -1$ there exists constants N_1, N_2 and N_3 such that

$$\|k_1\varepsilon + k_2\tfrac{1}{\tau}\left(1 - e^{\tau\bar{s}}\right)\| \leq N_1$$

$$\|\bar{p}_\varepsilon(\bar{s})\| \leq N_2|\bar{s}|^{n-1}$$

and

$$|\bar{s}^n - \bar{q}_\varepsilon(\bar{s})| \leq \varepsilon N_3 |\bar{s}|^{n-1}.$$

But then $\bar{g}_\varepsilon(\bar{s}) = 0$ implies:

$$|\bar{s}|^n \leq \varepsilon N_3|\bar{s}|^{n-1} + N_1 N_2 |\bar{s}|^{n-1}$$

which clearly implies that for ε small enough we must have that $|\bar{s}| < N_4$ for some constant N_4.

Next, we note that

$$\bar{s}^n - \tfrac{k_2(cb)}{\tau}\bar{s}^{n-1}(1 - e^{-\tau\bar{s}}) \tag{15.17}$$

has n zeros at the origin since $k_2(cb) \neq 1$. Moreover, this function has no zeros with $\operatorname{Re}\bar{s} > 0$. After all in that case

$$\bar{s} = \tfrac{k_2(cb)}{\tau}(1 - e^{-\tau\bar{s}}) \tag{15.18}$$

with $\operatorname{Re}(1 - e^{-\tau\bar{s}}) \geq 0$ and $k_2(cb) < 0$ yields that the right hand side lies in the open left half plane while $\bar{s}$ lies in the open right half plane which provides us with a contradiction. Thus, we have that

$\bar{q}_\varepsilon(\bar{s})$ has no zeros in the ORHP. Next, we consider the possibility of imaginary axis zeros. If there were any, one would need

$$\tau\bar{s} = 2q\pi j$$

for some integer q. Since in (15.18) the right hand side must be on the imaginary axis. However, this immediately yields $\bar{s} = 0$ if we go back to (15.18).

We find thus that (15.18) has exactly n zeros in $\operatorname{Re}\bar{s} \geq 0$ and $|\bar{s}| \geq N_4$. Using that the function is analytic, we find that there exists $\delta > 0$ such that this function has exactly n zeros in $\operatorname{Re}\bar{s} \geq -\delta$ and $|\bar{s}| \leq N_4$. But this implies that $\bar{g}_\varepsilon(s)$, which converges uniformly in the region $\operatorname{Re}\bar{s} \geq -\delta$ and $|\bar{s}| \leq N_4$ to (15.17), has exactly n zeros in this region. Since we already established that this function did not have zeros with $|\bar{s}| \geq N_4$, we find that (15.17) has exactly n zeros in $\operatorname{Re}\bar{s} \geq -\delta$.

Next we choose a region $\mathcal{K}$ in the open left half plane such that it contains all zeros of

$$\det\left(sI - A - b\left[k_1 + k_2 s\right]c\right) \tag{15.19}$$

which is possible since the derivative based feedback was stabilizing. We will show that $g_\varepsilon(s)$ has exactly n zeros in $\mathcal{K}$ which converge to the zeros of (15.6) as ε converges to zero. Similar to the proof of Theorem 15.1 we find:

$$k_1 + k_2\tfrac{1}{\varepsilon\tau}\left(1 - e^{-\varepsilon\tau s}\right) = k_1 + k_2 s + O(\varepsilon s).$$

But this implies that inside the compact region $\mathcal{K}$, $G_\varepsilon(s)$ converges uniformly to (15.19) as $\varepsilon \to 0$. This implies that $g_\varepsilon(s)$ has, for small enough ε, n zeros inside $\mathcal{K}$ which converge to the zeros of (15.6) as $\varepsilon \to 0$. Since $g_\varepsilon(s)$ has exactly n zeros in the region $\operatorname{Re}s \geq -\delta/\varepsilon$, we find that $g_\varepsilon(s)$ has no other zeros outside $\mathcal{K}$ in the region $\operatorname{Re}s \geq -\delta/\varepsilon$. This clearly implies that n zeros converge to the zeros of $g_\varepsilon(s)$ while the remaining zeros approach $-\infty$. This completes the proof of stability.

16. A CLASS OF NEUTRAL-TYPE DELAY DIFFERENTIAL EQUATIONS THAT ARE EFFECTIVELY RETARDED

We demonstrate that some delay-differential equations of neutral type can be equivalenced with retarded-type delay differential equations. In particular, for two classes of neutral-type delay differential equation models, we use state transformations to show that delayed derivatives can in some cases be expressed in terms of the model's state. Hence, we obtain conditions on the neutral-type delay differential equations for retarded equivalence.

16.1 Introduction

The class of differential equations that involve delayed derivatives is classically referred to in the mathematics and control communities as the neutral delay differential equations [234]. As opposed to retarded delay differential equations, those of neutral type *may* exhibit such peculiarities as spectra with chains of infinite numbers of roots in certain right half planes with imaginary parts tending to infinity, which unfortunately brings stability and robustness to parameter variations of such systems into question [234, 250]. In this chapter, we point out that many delay differential equations that are traditionally classified as neutral type (i.e., having delayed derivatives in the equation) are essentially retarded type. Specifically, we study two neutral delay differential equation models; the first is motivated in the study of output feedback control, while the second (and very classical) model arises in numerous feedback control as well as modeling applications. For both models, through using smart state transformations including the widely-used special coordinate basis (SCB) transformation [?] and more tailored transformations, we give conditions under which the delayed derivatives can be expressed in terms of the models' states, and hence show that such equations actually have retarded type dynamics. This study significantly helps clarify the definitions of neutral and retarded delay differential equations.

Although we focus here on linear time-invariant delay-differential equations, the retarded equivalences that we obtain can naturally be generalized to nonlinear delay systems using their normal-form representation. We do not believe that the non-linear case yields further application of or insight into our equivalence results, and so we limit ourselves to the linear case.

The remainder of the chapter is organized as follows. In Section 16.2, we motivate and describe the first neutral-type delay-differential equation model, namely one that arises when multiple output derivatives of an LTI system are used in feedback upon delay. Then we give a condition under which such a differential equation is equivalent to ones of retarded-type, using the SCB. In Section 16.3, we study a delay-differential equation model that is classical in the study of neutral systems, namely one in which multiply-delayed first derivatives are present. We give the necessary and sufficient condition that such a differential equation can be made equivalent to one of retarded type through a state transformation.

16.2 Equivalent Retarded Representations for a Multiple-Derivative-Feedback Model

Time-derivatives of system outputs (up to a certain order) are well-understood to codify state information [195]. Thus, computation of output derivatives is in its essence equivalent to (partial) state estimation, and is requisite for feedback controller design. For systems that are subject to time delays in observation, as well as ones where model-based observer design is impracticable and instead signal-based methods are needed (e.g., adaptive or decentralized systems), direct computation/approximation of output derivatives for feedback control is a promising strategy [10, 236, 246]. One natural means for using output derivatives in feedback is through delayed measurement. Complementarity, various natural and engineered systems from such diverse domains as computational biology and electric power system management are modeled using differential equations with delayed-derivative terms. Motivated by these complementary controls and modeling applications, we study the dynamics of a class of linear delay systems (linear delay differential equations) with delayed-derivative feedback.

The delayed-derivative model that we consider here comprises an LTI plant $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, $\mathbf{y} = C\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$, where the input $\mathbf{u}$ is a linear combination of delayed output derivatives of multiple orders. In particular, the input is

$$\mathbf{u}(t) = \sum_{i=0}^{M-1} K_i \mathbf{y}^{(i)}(t-h), \quad t \geq 0,$$

where the delay h is strictly positive, the gains $K_i \in \mathbb{R}^{m \times p}$ may be arbitrary, M is a positive integer, and the initial condition of the system is the signal $\mathbf{x}(t)$ over the time-interval $[-h, 0]$. This class of feedback models is representative of systems where observations of outputs and their derivatives (e.g., velocity or position-derivative measurements) are subject to delay (e.g., due to the need for communication through a data channel). Substituting for the input in terms of the output and then the state, we automatically see the closed-loop dynamics are described by the following delay-differential equation:

$$\dot{\mathbf{x}} = A\mathbf{x} + B \sum_{i=0}^{M-1} K_i C \mathbf{x}^{(i)}(t-h). \tag{16.1}$$

This delay differential equation is of neutral type for $M = 2$ and of advanced type for $M > 2$, since it involved first derivatives (respectively, higher derivatives) of the delayed state vector $\mathbf{x}(t-h)$ for $M = 2$ (respectively, $M > 2$). We refer to this model as **multiple-derivative-feedback model**.

The multiple-derivative-feedback model, which is nominally described by delay differential equations of neutral type, can equivalently be represented by delay differential equations of retarded type. That is, the delay differential equation can be rewritten without any delayed *derivative* terms. The concept underlying this reformulation is simple: derivatives of linear-system outputs (or their linear combinations) up to a certain order generally can be written as linear functions of the state variables, and hence in our case the delayed-derivative terms (up to a certain order) can be re-written in terms of the the state. The number of derivatives of particular output linear combinations that can be written in this way follows immediately from a structural decomposition of linear systems known as the *special coordinate basis (SCB)* [195,196]. This equivalence of output (linear combination) derivatives with states is well-established for finite-dimensional LTI plants. What our efforts here clarify is that such an equivalence is in force for delayed-derivative models, and in fact permits us to represent seemingly neutral/advanced-type systems as retarded ones.

To make the presentation clear to both control theorists and modelers, we focus our analysis on the control representations but then also explicitly consider the model form (closed-loop form) as needed. We develop the results in three steps. We first give a sufficient condition for the maximum number of derivatives that can be used in feedback such that, for any set of gains, the system can be equivalenced to a retarded one (Theorems 16.1). Second, we discuss the possibility of using higher derivatives of particular linear combinations of outputs while maintaining the retarded structure. A formal description of this general case would require us to develop the SCB in full intricacy (which detracts somewhat from the perspective put forth here), and so we only give a conceptual discussion.

Let us begin with the multiple-derivative-feedback model. Our condition for the maximum number of delayed-derivatives for which the dynamics is effectively retarded is easily phrased in terms of the *Markov parameters* of the plant (from which the special coordinate basis can be constructed, see [195]). We recall that the ith Markov parameter is given by $\mathcal{M}_i = CA^{i-1}B$, $i = 1, 2, \ldots$. In terms of the Markov parameters, we recover the following upper bound on the order of the delayed derivative, such that any controller will yield a retarded-type system:

Theorem 16.1. *Consider the multiple-derivative-feedback model, and assume that the delay is fixed (rather than designable). If the first q Markov parameters are identically zero, then the delay-derivative model for any $M \leq q+1$. can be rewritten as a retarded-type model.*

Proof: We claim that $\mathbf{y}^{(i)}(t-h) = CA^{(i)}\mathbf{x}(t-h)$, $i = 0, 1, 2, \ldots, q$. Let us verify this recursively. To do so, notice that the expression is clearly true for $i = 0$. Now say that the expression holds for arbitrary $i \in 1, \ldots, q-1$, and consider $\mathbf{y}^{(i+1)}(t-h)$. However, noting that $\mathbf{y}^{(i+1)}(t-h)$ equals $\frac{d}{dt}\mathbf{y}^{(i)}(t-h)$, we obtain that $\mathbf{y}^{(i+1)}(t-h) = \frac{d}{dt}CA^{(i)}\mathbf{x}(t-h) = CA^{(i)}(A\mathbf{x}(t-h) + B\mathbf{u}(t-h)) = CA^{(i+1)}\mathbf{x}(t-h) + CA^{(i)}B\mathbf{u}(t-h)$. Noticing that the first q Markov parameters are nil, we recover the result for the first q output derivatives. From this result, we automatically find that $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} = A\mathbf{x} + B\sum_{i=0}^{M-1} K_i\mathbf{y}^{(i)}(t-h)$ can in fact be written as $\dot{\mathbf{x}} = A\mathbf{x} + B\sum_{i=0}^{M-1} CA^{(i)}\mathbf{x}(t-h)$, for any $M \leq q+1$. Hence, the system is of retarded type in this case.

We thus see that many feedback control systems that at first glance appear to be of neutral or even advanced type are in fact retarded systems. We notice that their spectra do not display any of the characteristics of neutral-type systems, including infinite root chains and hyper-sensitivity to parameter variations. This observation indicates that feedback of delayed derivatives of low enough order will not yield highly unstable/sensitive dynamics, and in fact may be of use in stabilization and other control tasks.

When the highest derivative M in the multiple-derivative-feedback model is greater than or equal to the number of the first non-zero Markov parameter, it is easy to check that the dynamics will display the characteristics of neutral-type systems (e.g., infinite root-chains) for some feedback gains. However, certain linear combinations of the higher output derivatives may still be linear functions of the concurrent (in particular, identically delayed) part of the state, hence permitting a retarded representation of the closed-loop system for other gains. The number of derivatives of a particular combination of the output that can be written as linear functions of the state is made precise by the the *special coordinate basis* for linear systems [195]. In particular, by rewriting a linear system in its SCB through a linear transformation of the input, output, and state, we can view particular output linear combinations as being governed by input linear combinations that pass through chains of integrators of various depths. These integrator-chain depths, which equivalently are the orders of the *infinite-zeros* of the plant, immediately identify the number of derivatives of particular output combinations that are linear combinations of the state. Thus, we can determine whether particular feedback gains in multiple-derivative-feedback model yield dynamics that are effectively of retarded type. It is worth noting that, for the class of *square-invertible uniform-rank plants* (ones in which the the infinite zeros are of the same order $\widehat{M}$, or in other words the first non-zero Markov parameter is in fact square and full rank), only the first $M-1$ derivatives of any output combination can be written as a linear combination of past states.

16.3 Retarded Equivalence in a Multiply-Delayed-Derivative Model

Delay differential equations with multiply-delayed first derivatives of the state vector are also prominently used in modeling systems subject to time delay. These neutral-type models originate variously from control systems applications in which multiply-delayed observation derivatives are being used in feedback, as well as from modeling of systems in nature with response delays. Because these differential equations with multiply-delayed derivatives have traditionally been introduced in their differential equation form (rather than a control system form), we also progress from this *modeling* rather than controller design formulation. From this formulation, we study whether a state transformation can be used to to equivalence the neutral differential equation with a delay-differential equation of retarded type (in an algebraic sense as well as in terms of the sprectrum and sensitivity). In this case, we are able to obtain necessary and sufficient conditions for equivalence to a retarded system through any state transformation. We first present this general case along with some motivational examples. We then remark on the development of delay-independent conditions, and illustrate our results in the simple but useful case that the model originates from a feedback control paradigm.

Formally, let us consider the following system:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(x(t) - \sum_{j=1}^{M} H_j x(t-\rho_j)\right) = Ax(t) + \sum_{j=1}^{M} \overline{H}_j x(t-\tau_j) \tag{16.2}$$

This is a classical model for neutral linear time-invariant delay systems, which we refer to the **multiply-delayed-derivative model**. On the other hand we have the classical model for retarded delay systems:

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = Ax(t) + \sum_{j=1}^{M} \overline{H}_j x(t-\tau_j) \tag{16.3}$$

We recall an important property of retarded delay systems:

Lemma 16.2. *Consider a retarded system of the form* (16.3) *and the associated spectrum, i.e. the zeros of*

$$g(s) = \det\left(sI - A - \sum_{j=1}^{M} \overline{H}_j e^{-s\tau_j}\right)$$

Then for any $r \in \mathbb{R}$ there exists only a finite number of zeros of $g(s)$ in the half plane $\operatorname{Re} s \geq r$.

Proof: Note that, since $\tau_1, \ldots, \tau_M > 0$, there exists N such that for all s with $\operatorname{Re} s \geq r$ we have:

$$\left\| A + \sum_{j=1}^{M} \overline{H}_j e^{-s\tau_j} \right\| \leq N$$

But then clearly for $s \in \mathbb{C}$ with $|s| > N$ we have that

$$sI - A - \sum_{j=1}^{M} \overline{H}_j e^{-s\tau_j}$$

is invertible. Hence all zeros of $g(s)$ with $\operatorname{Re} s \geq r$ are in a bounded set $|s| < N$. But a nonzero analytic function has only a finite number of zeros in a bounded set.

Let us first present an example, that makes clear that state transformation can achieve retarded equivalence in the multiply-delayed-derivative model:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(x(t) - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} x(t-1) - \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} x(t-2) \right) = Ax(t) + \sum_{j=1}^{M} \overline{H}_j x(t - \tau_j)$$

where A and $\overline{H}_1, \ldots, \overline{H}_M$ can be arbitrary. We define a state space transformation:

$$\tilde{x}(t) = x(t) - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} x(t-1) - \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} x(t-2)$$

which is nicely invertible:

$$x(t) = \tilde{x}(t) - \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tilde{x}(t-1) - \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tilde{x}(t-2)$$

This transformation results in a model in terms of $\tilde{x}(t)$ which is of retarded type (16.3):

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\tilde{x}(t) = A &\left(\tilde{x}(t) - \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tilde{x}(t-1) - \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tilde{x}(t-2) \right) \\ + \sum_{j=1}^{M} \overline{H}_j &\left(\tilde{x}(t-\tau_j) - \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tilde{x}(t-\tau_j-1) - \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tilde{x}(t-\tau_j-2) \right) \end{aligned} \tag{16.4}$$

Consider the same example as in Example (16.3). Consider this model in the frequency domain:

$$s\left(x(s)-\begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}e^{-s}x(s)-\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}e^{-2s}x(s)\right) \quad = \quad Ax(s) \quad + \quad \sum_{j=1}^{M}\overline{H}_j e^{-\tau_j s}x(s)$$

Premultiply the above equation on both sides from the left by:

$$I-\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}e^{-s}-\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}e^{-2s}$$

We obtain, in the frequency domain:

$$sx(s) \quad = \quad \left(I-\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}e^{-s}-\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}e^{-2s}\right) \quad \times \quad \left(Ax(s)+\sum_{j=1}^{M}\overline{H}_j e^{-\tau_j s}x(s)\right)$$

which in the time domain yields a model in terms of $x(t)$ which is of retarded type (16.3):

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}t}x(t)=Ax(t)+\sum_{j=1}^{M}\overline{H}_j x(t-\tau_j)- \qquad (16.5)\\ &\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}\times\left(Ax(t-1)+\sum_{j=1}^{M}\overline{H}_j x(t-\tau_j-1)\right)-\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\left(Ax(t-2)+\sum_{j=1}^{M}\overline{H}_j x(t-\tau_j-2)\right)\end{aligned}$$

The interesting aspect is that this new model is of retarded type in the original state space coordinates without even using a basis transformation.

Based on these examples, we are motivated to determine conditions such that a neutral system of the form (16.2) can be transformed into a retarded system of the form (16.3). We have the following result:

Theorem 16.3. *Consider a system of the form* (16.2)*. There exists an invertible basis transformation of the form:*

$$\tilde{x}(t)=x(t)-\sum_{j=1}^{M}H_j x(t-\rho_j) \qquad (16.6)$$

such that $\tilde{x}(t)$ satisfies a retarded delay model of the form (16.3) *if and only if*

$$f(s) = \det\left(I - \sum_{j=1}^{M} H_j e^{-s\rho_j}\right)$$

has no zeros in the complex plane or, equivalently, the function f is equal to a constant a. Moreover, the basis transformation (16.6) *has the property that*

$$\left(I - \sum_{j=1}^{M} H_j e^{-\rho_j s}\right)^{-1} = I - \sum_{j=1}^{N} \overline{V}_j e^{-\mu_j s}$$

for appropriately chosen $\overline{V}_1, \ldots, \overline{V}_N$ and we have, besides (16.6)*, that:*

$$x(t) = \tilde{x}(t) - \sum_{j=1}^{N} \overline{V}_j \tilde{x}(t - \mu_j)$$

Proof: Note that an invertible basis transformation will not affect the spectrum of the system. After Laplace transformation our model is of the form

$$H(s)x(s) = 0$$

where

$$H(s) = sI + \sum_{j=1}^{M} sH_j e^{-s\rho_j} - A - \sum_{j=1}^{M} \overline{H}_j e^{-s\tau_j}$$

Assume $H(s_0)$ is singular, i.e. s_0 is an element of the point spectrum. In other words, there exists $x_0 \neq 0$ such that $H(s_0)x_0 = 0$. Then $x(t) = \operatorname{Re} e^{-s_0 t} x_0$ satisfies the system dynamics. Given the structure of our basis transformation, this implies that $\tilde{x}(t) = \operatorname{Re} e^{-s_0 t} \tilde{x}_0$ satisfies the retarded model and this yields that s_0 is also an element of the point spectrum of the retarded model. Hence if our original model has a point spectrum which has an infinite number of points in $\operatorname{Re} s \geq r$ then by Lemma 16.2 it can not be transformed into a retarded model by a basis transformation.

Next, note that $f(s)$ can clearly be written as:

$$f(s) = 1 - \sum_{i=1}^{R} \alpha_i e^{-\beta_i s}$$

where, without loss of generality, we assume that $\beta_1 < \beta_2 < \ldots < \beta_R$. We note that for $s \in \mathbb{C}$ with $\operatorname{Re} s$ sufficiently large we have:

$$\left| \sum_{i=1}^{R} \alpha_i e^{-\beta_j s} \right| \leq \frac{1}{2}$$

and hence $f(s)$ has no zero in that region. Next, we note that:

$$f(s) = -\alpha_R e^{-s\beta_n} \left(1 - \tfrac{1}{\alpha_R} e^{\beta_R s} + \sum_{i=1}^{R-1} \tfrac{\alpha_i}{\alpha_R} e^{(\beta_R - \beta_i)s} \right)$$

Hence for $s \in \mathbb{C}$ with $\operatorname{Re} s$ sufficiently small we have

$$\left| \tfrac{1}{\alpha_R} e^{\beta_R s} - \sum_{i=1}^{R-1} \tfrac{\alpha_i}{\alpha_R} e^{(\beta_R - \beta_i)s} \right| \leq \frac{1}{2}$$

and hence $f(s)$ has no zero in that region either. Therefore all zeros of $f(s)$ are in a band:

$$\mathcal{L} := \{\, s \in \mathbb{C} \mid \beta_1 \leq \operatorname{Re} s \leq \beta_2 \,\}.$$

We note that the spectrum of the system is defined by the zeros of

$$h(s) = \det \left[s \left(I - \sum_{j=1}^{M} H_j e^{-s\rho_j} \right) - A - \sum_{j=1}^{M} \overline{H}_j e^{-\tau_j s} \right]$$

Assume r_1 is such that $h(s)$ has an infinite number of zeros in the region $\operatorname{Re} s \geq r_1$. It is easily established that $h(s)$ has no zeros in the region $\operatorname{Re} s \geq r_2$ for r_2 sufficiently large. Hence if we choose without loss of generality $\beta_2 > r_2$ and $\beta_1 < r_1$ we find hat $h(s)$ has an infinite number of zeros in the strip $\mathcal{L}$. Note that we have:

$$h(s) = s^n \left(f(s) + \tfrac{1}{s} g(s) \right)$$

where $g(s)$ is an analytic function which is bounded on the strip $\bar{\mathcal{L}}$ defined by:

$$\bar{\mathcal{L}} := \{\, s \in \mathbb{C} \mid \beta_1 \leq \operatorname{Re} s \leq \beta_2,\ \ |s| > 1 \,\}.$$

Clearly if $h(s)$ or, equivalently, $f(s) + \frac{1}{s} g(s)$ has an infinite number of zeros in $\mathcal{L}$ then it also has an infinite number of zeros in $\bar{\mathcal{L}}$, i.e. we have s_k for which $h(s_k) = 0$ and since necessarily $s_k \to \infty$

we find that $f(s_k) \to 0$ as $k \to \infty$. We will show that this implies that f has an infinite number of zeros in $\mathcal{L}$.

Conversely, we will show that if $f(s)$ is not a constant then $f(s)$ has an infinite number of zeros in $\mathcal{L}$ and we will also establish that in this case also $h(s)$ has an infinite number of zeros in $\mathcal{L}$ Hence, using these results we find that a basis transformation exists if and only if $f(s)$ is a constant.

Next, we present our core mathematical result proving the claims made above. This result can be based on a fundamental result for almost functions obtained in [248] and Hurwitz's theorem (see [243]) as used in [249, Lemma 1, p. 268] to establish the existence of an infinite number of zeros of $f(s)$ using its structure as an exponential function. Our proof is based on first principles.

Note that the structure of $f(s)$ implies that f and all its derivatives are bounded on $\mathcal{L}$. Moreover $f(s)$ is bounded away from zero outside the set $\mathcal{L}$. Hence if $f(s)$ is bounded away from zero inside the set $\mathcal{L}$ then the function $1/f(s)$ would be a bounded analytic function which, by Liouville's theorem is then constant and therefore $f(s)$ is equal to a constant. Therefore if $f(s)$ is not a constant then either f has a zero in $\mathcal{L}$ or it has a sequence $s_1, s_2, \ldots$ with $s_k \to \infty$ and $f(s_k) \to 0$ as $k \to \infty$. In case $\mathcal{L}$ has a zero $\bar{s}$ in $\mathcal{L}$ then the fact that $f(s)$ is almost periodic implies that also in this case, we can construct a sequence $s_1, s_2, \ldots$ with $s_k \to \infty$ and $f(s_k) \to 0$ as $k \to \infty$. Remains to show that this implies that $f(s)$ has an infinite number of zeros.

We construct a subsequence of s_k such that

$$\begin{aligned} f(s_{k_j}) &\to 0 \\ f(s_{k_j}^{(1)}) &\to 0 \\ \vdots \quad & \vdots \\ f(s_{k_j}^{(\ell-1)}) &\to 0 \\ |f^{(\ell)}(s_{k_j})| &> \beta \end{aligned}$$

as $j \to \infty$. This is possible for some $\ell \leq r$ since otherwise we would find a sequence such that:

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & -\beta_1 & \cdots & -\beta_r \\ \vdots & \vdots & & \vdots \\ 0 & (-\beta_1)^r & \cdots & (-\beta_r)^r \end{pmatrix} \begin{pmatrix} 1 \\ -\alpha_1 e^{-\beta_1 s_{k_j}} \\ \vdots \\ -\alpha_r e^{-\beta_r s_{k_j}} \end{pmatrix} \to 0$$

as $j \to 0$ which is impossible since the given matrix is invertible (using the invertibility of the Vandermonde matrix). We can even guarantee that $f^{(\ell)}(s_{k_j})$ converges to some fixed value Z since this sequence is bounded and hence an appropriate subsequences converges. We then obtain:

$$f(s) = T_{s_{k_j}}(s - s_{k_j}) + \tfrac{f^{(\ell)}(s_{k_j})}{\ell!}(s - s_{k_j})^\ell + v(s)(s - s_{k_j})^{\ell+1}$$

where $T_{s_{k_j}}(s)$ is the Taylor polynomial around s_{k_j} of order $\ell - 1$ whose coefficients, by construction, converges to zero. Hence

$$f_1(\bar{s}) = T_{s_{k_j}}(\bar{s}) + \tfrac{f^{(\ell)}(s_{k_j})}{\ell!}\bar{s}^\ell$$

converges uniformly in the region $|\bar{s}| \leq r$ to

$$Z\bar{s}^\ell$$

Hence by Hurwitz's theorem $f_1(\bar{s})$ has a zero in the region $|\bar{s}| \leq r$ for j large enough. For r small enough we have:

$$v(\bar{s} + s_{k_j})\bar{s}^{\ell+1} < f_1(\bar{s})$$

for all $\bar{s}$ with $\bar{s} = r$ as long as j is large enough since h is bounded and $f_1(\bar{s})$ converges to $Z\bar{s}^{\ell}$. But then Rouche's theorem guarantees that also

$$\bar{f}(\bar{(}s)) = f_1(\bar{s}) + h(\bar{s} + s_{k_j})\bar{s}^{\ell+1}$$

has a zero in $|\bar{s}| \leq r$ for j large enough. But this implies that $f(s)$ has a zero in the ball $|s - s_{k_j}| < r$ for all j large enough which yields an infinite number of zeros in the given region.

We need to prove that also $h(s)$ or

$$f(s) + \tfrac{1}{s}g(s)$$

has an infinite number of zeros where $g(s)$ is bounded. Since s_{k_j} converges to infinity within $\mathcal{L}$ we can assume without loss of generality that we are within $\bar{\mathcal{L}}$ where $g(s)$ is bounded. Next we use the same arguments as above. Clearly:

$$f_2(\bar{s}) = T_{s_{k_j}}(\bar{s}) + \tfrac{1}{s_{k_j}+\bar{s}}g(s_{k_j} + \bar{s}) + \tfrac{f^{(\ell)}(s_{k_j})}{\ell!}\bar{s}^{\ell}$$

converges uniformly in the region $|\bar{s}| \leq r$ to

$$Z\bar{s}^{\ell}$$

Hence by Hurwitz's theorem $f_2(\bar{s})$ has a zero in the region $|\bar{s}| \leq r$ for j large enough. For r small enough we have:

$$v(\bar{s} + s_{k_j})\bar{s}^{\ell+1} < f_2(\bar{s})$$

for all $\bar{s}$ with $\bar{s} = r$ as long as j is large enough since h is bounded and $f_2(\bar{s})$ converges to $Z\bar{s}^\ell$. But then Rouche's theorem guarantees that also

$$\bar{h}(\bar{(s)}) = f_2(\bar{s}) + h(\bar{s} + s_{k_j})\bar{s}^{\ell+1}$$

has a zero in $|\bar{s}| \leq r$ for j large enough. But this implies that $h(s)$ has a zero in the ball $|s - s_{k_j}| < r$ for all j large enough which yields an infinite number of zeros in the given region.

Finally, we need to look at the invertibility of the state space transformation (16.6) in case $f(s)$ has a finite number of zeros in $f(s)$ or equivalently $f(s)$ is a constant. We look at the invertibility of the matrix:

$$I - \sum_{j=1}^{M} H_j e^{-\rho_j s}$$

for the case that $f(s)$, which is equal to the determinant of this matrix, is a constant. The inverse is equal to the adjoint matrix divided by the determinant. We note that the elements of the adjoint matrix are defined by multiplication and addition of elements of the original matrix. Hence it will be a linear combination of delays. Dividing by the determinant does not affect this since in our case this is just a constant.

Remark 1. *Note that just as in Example 16.3, instead of a state space transformation we can also find a retarded model in terms of the original state x by premultiplying the model (after Laplace transformation) by:*

$$I - \sum_{j=1}^{N} \overline{V}_j e^{-\mu_j s}$$

which is of course only well-defined in case $f(s)$ is equal to a constant.

In the above theorem, we have given necessary and sufficient conditions such that basis transformations within a particular class can be used to convert the multiply-delayed derivative model (16.2) into a neutral-type equation. The following theorem shows that, if the condition of the above theorem is not satisfied, then in fact there does not exist even a more general state transformation to bring the system into retarded form. Since any reasonable basis transformation should preserve the spectrum, the fact that the spectrum contains an infinite number of poles in a vertical strip in the complex plane means it does not satisfy the property outlined in Lemma 16.2 that retarded systems will always have only a finite number of poles in such a vertical strip. We use this spectral condition to show that retarded-equivalence is not possible if the conditions of the above theorem are not met:

Theorem 16.4. *Consider a system of the form* (16.2) *and define*

$$f(s) = \det\left(I - \sum_{j=1}^{M} H_j e^{-s\rho_j}\right)$$

The function f either has an infinite number of zeros in a vertical strip in the complex plane or is equal to a constant.

If f has an infinite number of zeros in a vertical strip then also

$$g(s) = \det\left(sI + \sum_{j=1}^{M} sH_j e^{-s\rho_j} - A - \sum_{j=1}^{M} \overline{H}_j e^{-s\tau_j}\right)$$

has an infinite number of zeros in a vertical strip. In other words, the spectrum of the system contains an infinite number of poles in a vertical strip in the complex plane.

Proof: This follows directly from the proof of Theorem 16.3

Interestingly, the ability to transform the neutral-type differential equation into a retarded-type equation may be highly sensitive to changes in the delays:

Consider the same system an in example (16.3) but with some uncertainty in the delay terms:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(x(t) - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} x(t-\rho_1) - \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} x(t-\rho_2)\right) = Ax(t) + \sum_{j=1}^{M} \overline{H}_j x(t - \tau_j)$$

Applying Theorem 16.3 we construct:

$$f(s) = 1 - e^{-2\rho_1 s} + e^{-\rho_2 s}$$

and note that the system is equivalent to a retarded system if and only if $2\rho_1 = \rho_2$, a property that is clearly trivially ruined by small perturbations in the delay.

Given the sensitivity to perturbations in the delays of the state space transformations, we can ask ourselves the question of whether we can find a characterization which is independent of the delays. The following theorem gives such a delay-independent characterization:

Theorem 16.5. *Consider a system of the form* (16.2). *There exists for all* $\rho_1, \rho_2, \ldots, \rho_M$ *an invertible basis transformation of the form:*

$$\tilde{x}(t) = x(t) - \sum_{j=1}^{N} \overline{V}_j x(t - \mu_j) \tag{16.7}$$

such that $\tilde{x}(t)$ *satisfies a retarded delay model of the form* (16.3) *if and only if*

$$\bar{f}(z_1, \ldots, z_M) = \det \left(I - \sum_{j=1}^{M} H_j z_j \right)$$

has no zeros in the complex plane or, equivalently, the function $\bar{f}$ *is equal to a constant* a.

Proof: Note that for any value for $\rho_1, \rho_2, \ldots, \rho_M$ we have that such a basis transformation exists if and only if $f(s)$ is a constant. We know

$$f(s) = 1 - \sum_{i=1}^{R} \alpha_i e^{-\beta_i s}$$

while

$$\begin{pmatrix} \beta_1 \\ \vdots \\ \beta_R \end{pmatrix} = A \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_M \end{pmatrix}$$

while the α_i are independent of ρ. Without loss of generality, we can exclude that $\beta_i = \beta_j$ for all $\rho_1, \rho_2, \ldots, \rho_M$ (then we can simple combine both terms in one). But $f(s)$ is then equal to a constant if either all α_i are equal to zero or if $\beta_i = \beta_j$ for some i and j and the corresponding α_i cancel. In the first case, clearly $f(s)$ is equal to a constant for all $\rho_1, \ldots, \rho_M$ and it is easily seen that $\bar{f}(z_1, \ldots, z_M)$ is equal to a constant. Conversely if $\beta_i = \beta_j$ then this is a nontrivial linear equation and the set of $\rho_1, \ldots, \rho_M$ that satisfy this form a hyperplane. Hence the points for which $f(s)$ is a constant form the union of a finite set of hyperplanes and an arbitrary small perturbation brings you to a function $f(s)$ which has a zero and then

$$\bar{f}(e^{\rho_1 s}, \ldots, e^{\rho_M s}) = f(s) = 0$$

Note that the above condition on $\bar{f}$ is still a necessary and sufficient condition, when only small perturbations of the delays (rather than arbitrary valuations of them) are possible. That is, if given $\bar{\rho}_1, \ldots, \bar{\rho}_M$, we require existence of $\varepsilon > 0$ such that for all $\rho_1, \ldots, \rho_M$ with $|\rho_i - \bar{\rho}_i| < \varepsilon$ there is a basis transformation such that the new state satisfies a model of the form (16.3), then the condition is necessary and sufficient.

Of interest, the above delay-independent condition for retarded equivalence can be written explicitly in terms of the matrices H_i, rather than in terms of the existence of zeros of a function defined thereof. Before presenting this result, we require a little further notation and terminology. Specifically, let us consider products of m matrices such that each matrix is one of $H_1, \ldots, H_M$. We note that that there are 3^M such products in total. Of these, $\frac{m!}{k_1! \ldots k_M!}$ have k_1 terms equal to H_1 in the product, k_2 terms equal to H_2, and so on (and where $k_1 + \ldots + k_M = m$). Let us define the sum of these $\frac{m!}{k_1! \ldots k_M!}$ terms as $Q(k_1, \ldots, k_M; m)$, and call them **combinatorial sums**.

Theorem 16.6. *Consider the multiply-delayed-feedback model (Equation), where M is the number of delay terms and n is the dimension of $\mathbf{x}(t)$. If there exists $i \in 1, \ldots, n$ such that the combinatorial* *sums* $Q(k_1, \ldots, k_M; i)$, $k_1 \geq 0, \ldots, k_M \geq 0$, $k_0 + \ldots + k_{M-1} = i$ *are all zero, then the model is equivalent to a retarded-type model. Furthermore, if there is no such* i*, then the model cannot be viewed as retarded-equivalent for at least some sets of delays* $\rho_1, \ldots, \rho_M$.

This result follows algebraically from the above Theorem 16.2. We omit the details.

Finally, let us briefly discuss an example where the multiply-delayed derivative model is obtained from a controls paradigm, to crystallize the connection between the special coordinate basis transformation (as used in the previous section) and the transformation considered here. Precisely, let us consider an LTI plant $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$, $\mathbf{y} = C\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$, where the input $\mathbf{u}$ is a linear combination of multiply-delayed outputs and output derivatives:

$$\mathbf{u}(t) = \sum_{i=1}^{M} \overline{K}_i \mathbf{y}(t - \tau_i) + K_i \dot{\mathbf{y}}(t - \rho_i), \quad t \geq 0,$$

where WLOG $0 < \rho_1 < \rho_2 < \ldots < \rho_M$, $0 < \tau_1 < \tau_1 < \ldots < \tau_M$, and the gains $\widehat{K}_i$ and K_i may be arbitrary. We recover immediately from the special coordinate basis transformation (or from first principles) that the closed-loop dynamics of this neutral-type system is equivalent to a retarded system whenever $CB = 0$. However, we see that the condition is by no means necessary for retarded-equivalence. For instance, consider the system with state equation $\dot{\mathbf{x}}(t) = \mathbf{u}(t)$ and observation $\mathbf{y}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{x}(t)$, with control law $\mathbf{u}(t) = \dot{\mathbf{y}}(t - \rho)$. This system's first Markov parameter CB is nonzero, and yet the closed-loop dynamics satisfy $\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}^3 \mathbf{u}(t - 3\rho) = 0$, or in other words the dynamics are retarded-equivalent. This example makes evident that the special coordinate basis transformation is concerned with equivalencing delayed output derivatives with the *concurrent* state, and so is a special case of the transformation developed in this section for the multiply-delayed-derivative model. We leave it to future work to check the whether the broader transformation can be given a structural control-theoretic interpretation, and whether such a transformation can be applied to the multiple derivative feedback model.

17. EXPLICIT PRECOMPENSATOR DESIGN FOR INVARIANT-ZERO CANCELLATION

We explicitly construct a precompensator that cancels all open left half plane (OLHP) finite invariant zeros of a general MIMO LTI plant, by exploiting the special coordinate basis for linear systems. This approach to cancellation avoids the inclarity inherent in defining generalized zero directions, and so clarifies cancellation of defective eigenvalues in the zero dynamics.

17.1 Main Result

We revisit the classical problem of pole-zero cancellation for MIMO plants. In [251], Douglas and Athans fully characterize the cancellation of *simple* OLHP poles and zeros in square-invertible LTI plants, through consideration of the associated zero directions. Unfortunately, the definition of zero directions for defective eigenvalues of the invariant zero dynamics is problematic, see [238], and so cancellation of such zeros has not been fully characterized. In this brief correspondence, we exploit the *special coordinate basis* (SCB) for linear systems [196, 205] to explicitly construct a precompensator that can cancel all OLHP invariant zeros of an arbitrary LTI plant, and hence we address cancellation of both simple and defective eigenvalues in full generality. The invariant-zero cancellation design involves a cascade of two pre-compensators (see Figure 1): the first compensator C_1 makes the cascaded plant uniform rank, while maintaining the finite-zero structure; the second compensator C_2 serves for invariant zero cancellation, i.e., forces the OLHP invariant zeros of the plant to become output-decoupling zeros. The following theorem formalizes the design:

Theorem 17.1. *Consider a minimum-phase stabilizable and detectable LTI plant with q inputs. A q-input precompensator (see Figure 1) can be designed, so that 1) the finite invariant zeros of the whole system are at the locations of the plant invariant-zeros and the OLHP invariant zeros further become output-decoupling zeros (i.e., they are cancelled), 2) stabilizability and detectability is maintained, and 3) the invertibility properties of the plant are maintained (i.e., a left-invertible plant or right-invertible plant remains so upon precompensation, while a non-left-and-right invertible plant remains non-left-and-right invertible and in fact maintains the same normal rank).*

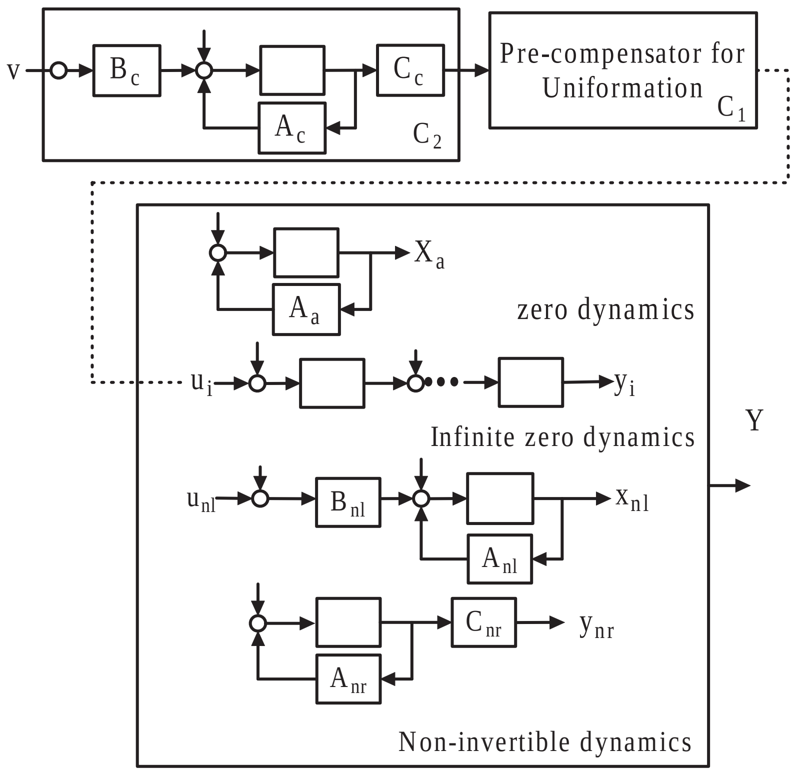


Fig. 17.1: System structure showing SCB blocks and pre-compensator design

Proof:

Let us begin with the cancellation design for square invertible plants. We will address non-invertible plants later in the proof. For invertible plants, the number of inputs, number of outputs, and normal rank are all equal (to, say, m).

The design is composed of two steps. In the first step, we construct a pre-compensator to cascade with the plant which makes the cascaded plant uniform rank, while preserving the finite-zero dynamics. The precise compensator design is given in [254].

In the second step, we construct a pre-compensator for invariant-zero cancellation. The construction is made possible using SCB, which explicitly exposes a plant's invariant zeros and output-decoupling zeros, and hences facilitates the design that transforms the invariant zeros to output-decoupling zeros. A uniform rank system can be written as Σ in SCB (Equation 17.1) through input, state and output transformations, Γ_i, Γ_s, and Γ_o, respectively:

$$\begin{aligned}
\Sigma: \quad \dot{\mathbf{x}}_a &= A_a\mathbf{x}_a + A_1\mathbf{x}_1 \\
\dot{\mathbf{x}}_1 &= \mathbf{x}_2 \\
&\vdots \\
\dot{\mathbf{x}}_q &= E_a\mathbf{x}_a + L(\mathbf{x}_1, ...\mathbf{x}_q) + \mathbf{u} \\
\mathbf{y} &= \mathbf{x}_1,
\end{aligned} \tag{17.1}$$

where $u \in R^{m\times 1}$, $A_a \in R^{n_a\times n_a}$, and $L()$ denotes a linear function of the elements in (). Here, the eigenvalues of A_a are the invariant zeros of the plant, and $\dot{\mathbf{x}}_a = A_a\mathbf{x}_a + A_1\mathbf{x}_1$ is referred as the zero dynamics. The output $\mathbf{y}$ and its derivatives $\mathbf{x}_2, ..., \mathbf{x}_q$ are the states of the infinite-zero chains. An invariant zero is an output-decoupling zero if it is an unobservable eigenvalue of the pair (E_a, A_a). Hence, in order to cancel all invariant zeros of the plant Σ, we can eliminate the appearance of the state associated with zero dynamics in the infinite zero chain. We realize this elimination through the following pre-compensator design.

Let us first construct the m-input pre-compensator, as

$$\dot{\mathbf{x}}_c = diag_m(A_a)\mathbf{x}_c + B_c\mathbf{v} \tag{17.2}$$
$$\mathbf{y}_c = \Gamma_i C_c \mathbf{x}_c,$$

where $\mathbf{u} = \Gamma_i^{-1}\mathbf{y}_c$, $diag_m()$ denotes the block-diagonal with m copies of the argument as the blocks, $B_c \in R^{mn_a \times m}$, and $C_c \in R^{m \times mn_a}$. Clearly, the pre-compensator has stable dynamics.

B_c and C_c are designed in the following steps.

1) Find the minimum polynomial degree* of A_a, and denote it by l.

2) Design B_c as $diag_m(b_c)$, where $b_c \in R^{m\times 1}$ is chosen such that (A_a, b_c) has controllability index l. To see that this is possible, notice that the minimum polynomial degree of a matrix is the sum of the maximum geometric multiplicities of the distinct eigenvalues of the matrix [255], which is precisely equal to the maximum achievable dimension of the controllable space and hence the maximum achievable controllability index.

3) Find the state transformation P such that $P^{-1}A_aP = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$, $P^{-1}b_c = \begin{bmatrix} B_{11} \\ 0 \end{bmatrix}$, where $A_{11} \in R^{l\times l}$ and $B_{11} \in R^{l\times 1}$ are in the controllability canonical form.

4) Design C_c as $diag_m(c_c)$, where $c_c \in R^{1\times m}$ is $\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} P^{-1}$.

* Here, by minimum polynomial degree, we mean the minimum integer l such that A^l can be written as a linear combination of A^i, for $i = 0, 1, 2, \dots, l-1$.

Notice that the above design of B_c and C_c is achieved through simple algebraic procedures. Clearly, this design guarantees that the Markov parameters $C_c diag_m(A_a)^i B_c$ (see [256] for more explanation) are zero for $i = 0, ..., l-2$, while $C_c diag_m(A_a)^{l-1} B_c$ equals identity and is full rank.

Now let us show that this compensator forces the input direction of the invariant-zeros to null, or in other words makes all the invariant-zeros of the closed-loop system output-decoupling zeros or cancels them. To do so, let us analyze the infinite zero structure of the compensated system. We claim that the infinite-zero chains are each expanded by l from the infinite zero chain of the original system, and in turn that the input directions of the invariant-zeros are null. To do this, let us consider the first l derivatives of x_q, as follows:

Step 1: Denote $\dot{\mathbf{x}}_q$ as $\mathbf{x}_{q+1}$. Combining Equations 17.1 and 17.2, we obtain $E_a\mathbf{x}_a + C_c\mathbf{x}_c = L(\mathbf{x}_1, ..., \mathbf{x}_{q+1})$, and also $\dot{\mathbf{x}}_{q+1} = E_a A_a \mathbf{x}_a + L(\mathbf{x}_1, ...\mathbf{x}_{q+1}) + C_c diag_m(A_a)\mathbf{x}_c + C_c B_c \mathbf{v}$. The last term is 0, according to the design of B_c and C_c.

Step 2: Similarly, denote $\dot{\mathbf{x}}_{q+1}$ as $\mathbf{x}_{q+2}$, we obtain $E_a A_a \mathbf{x}_a + C_c diag_m(A_a)\mathbf{x}_c = L(\mathbf{x}_1, ...\mathbf{x}_{q+2})$, and also $\dot{\mathbf{x}}_{q+2} = E_a A_a^2 \mathbf{x}_a + L(\mathbf{x}_1, ...\mathbf{x}_{q+2}) + C_c diag_m(A_a^2)\mathbf{x}_c + C_c diag_m(A_a) B_c \mathbf{v}$. The last term is also 0, according to the design of B_c and C_c.

$\vdots$

Step l: Denote $\dot{\mathbf{x}}_{q+l-1}$ as $\mathbf{x}_{q+l}$, we obtain $E_a A_a^{l-1}\mathbf{x}_a + C_c diag(A_a^{l-1})\mathbf{x}_c = L(\mathbf{x}_1, ...\mathbf{x}_{q+l})$, and also $\dot{\mathbf{x}}_{q+l} = E_a A_a^l \mathbf{x}_a + L(\mathbf{x}_1, ...\mathbf{x}_{q+l}) + C_c diag(A_a^l)\mathbf{x}_c + C_c diag_m(A_a^{l-1}) B_c \mathbf{v}$. According to the design of C_c and B_c, $E_a A_a^l \mathbf{x}_a + C_c diag_m(A_a^l)\mathbf{x}_c$ can be expressed as a linear combination of $E_a A_a^i \mathbf{x}_a + C_c diag_m(A_a^i)\mathbf{x}_c$, where $0 \le i \le l-1$, and also $C_c diag_m(A_a)^{l-1} B_c$ is full rank while $C_c diag_m(A_a)^i B_c = 0$ for $0 \le i \le l-1$. These properties lead to $\dot{\mathbf{x}}_{q+l} = L(\mathbf{x}_1, ...\mathbf{x}_{q+l}) + C_c diag_m(A_a^{l-1}) B_c \mathbf{v}$.

From the expression of $\dot{\mathbf{x}}_{q+l}$, clearly, the invariant zeros are now output-decoupling zeros, or in other words the input directions of the invariant zeros are null [205]. Also, the compensator introduces $m(n_a - l)$ further invariant zeros at the locations of the plant's invariant zeros, which are also output-decoupling zeros. The entire system remains uniform rank and invertible, with normal rank m. It follows immediately that stabilizability and detectability are maintained.

In the non-invertible case, we can use the full Special Coordinate Basis (SCB) transformation (see [205]) to generate the invariant-zero-cancelling design. For instance, for a non-left-invertible but right-invertible system, the SCB for the system upon rank-uniformization has the following form:

$$\begin{aligned}
\dot{\mathbf{x}}_a &= A_a \mathbf{x}_a + A_1 \mathbf{x}_1 \\
\dot{\mathbf{x}}_{nl} &= A_{nl} \mathbf{x}_{nl} + L_{nl} \mathbf{x}_1 + B_{nl}(\mathbf{u}_{nl} + E_{nl} \mathbf{x}_a) \\
\dot{\mathbf{x}}_1 &= \mathbf{x}_2 \\
&\vdots \\
\dot{\mathbf{x}}_q &= E_a \mathbf{x}_a + L(\mathbf{x}_1, ... \mathbf{x}_q) + L_d \mathbf{x}_{nl} + \mathbf{u} \\
\mathbf{y} &= \mathbf{x}_1
\end{aligned} \tag{17.3}$$

In these coordinates, it is clear that we can simply precompensate the dynamics associated with the infinite-zero structure (denoted $\mathbf{u}$) as above, and leave the remaining inputs (denoted $\mathbf{u}_{nl}$) unchanged. In this way, we maintain the invertibility properties of the plant while achieving invariant-zero cancellation. A similar argument permits precompensation for non-right-invertible dynamics, and so the full design for invariant-zero-cancellation is complete.

Notice that in the above proof, we have given an explicit algebraic construction of the precompensator that cancels the invariant zeros of a multivariable plant.

Remarks: There are several further issues worth discussing.

First, we reiterate that [251] also discusses pole-zero cancellation for MIMO systems. However, the definition of MIMO invariant zeros in this work is problematic: when the invariant zeros are not simple, the zero directions are hard to identify (see [252] for a detailed discussion). Our work here shows that the correct and clear way to study invariant zeros is through the SCB transformation [196, 205], as we have done. It is this transformation that permits us to explicitly develop controllers that cancel non-simple invariant zeros. Equivalently, we note that our work can be viewed as providing a full deconstruction of the conditions for invariant zero cancellation presented in [251]: although the work [251] analyzes the concurrence of poles and zeros for an existing system rather than designing extra dynamics to cancel the invariant zeros of a system as we do here, our structural insight into invariant zero cancellation carries over to this case.

Second, the pre-compensator introduces $(n-l)\times m$ extra decoupling zeros, but at the same locations as those of the plant (and hence in the OLHP). We notice that we can trivially modify the pre-compensator to exclude those extra invariant zeros while keeping the input-output relationship unchanged, by implementing a minimal realization of the controller. The system upon use of this reduced pre-compensator maintains the invertibility properties of the original plant, while not introducing any extra invariant zeros.

Third, the design can easily be generalized to permit cancellation of the OLHP invariant zeros of a non-minimum phase plant.

Fourth, the compensator design that we show in the proof is not unique, and may not be the one that introduces the dynamics of minimal order. This is because we only need to design k, C_c and B_c that satisfy the following two requirements: 1) $E_aA_a^k\mathbf{x}_a + C_c diag_m(A_a^k)\mathbf{x}_c$ can be expressed as a linear combination of $E_aA_a^i\mathbf{x}_a + C_c diag_m(A_a^i)\mathbf{x}_c$, where $0 \leq i \leq k-1$; and 2) the Markov parameters are 0 for $0 \leq i \leq k-2$ and non singular for $i = k-1$. The minimal polynomial degree may not be the smallest number that works. For instance, there may exist a number k between the minimal polynomial degree and the observability index of the pair (A_a, E_a) such that $E_a A_a^k$ is a linear combination of $E_a A_a^i$, where $0 \leq i \leq k-1$.

Fifth, the precompensator designed in the proof is a decentralized one, in the sense that the dynamics added at each input channel are decoupled.

Sixth, for some special cases, finding an adequate pre-compensator is quite simple, and we do not need to go through the procedures in the proof to design C_c and B_c. Here we note one special case: if the system is SISO with no decoupling zeros, then $l = m$, and we can choose $C_c = E_a$ and then choose B_c to achieve the desired Markov parameters.

Example 1: In this example, we design a pre-compensator to cancel the invariant zeros of the following system

$$\dot{x} = \begin{bmatrix} 1 & 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} u \tag{17.4}$$

$$y = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} x. \tag{17.5}$$

Clearly, the system is already in its SCB form, and is square invertible and uniform rank. For clarity, we rewrite the system in its SCB form as follows:

$$\begin{aligned}
\dot{\mathbf{x}}_a &= I_3\mathbf{x}_a + \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x}_1 \qquad (17.6)\\
\dot{\mathbf{x}}_1 &= \mathbf{x}_2\\
\dot{\mathbf{x}}_2 &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\mathbf{x}_a + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\mathbf{x}_1 + \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\mathbf{x}_2 + \mathbf{u}\\
\mathbf{y} &= \mathbf{x}_1.
\end{aligned}$$

Since the minimal polynomial degree of A_a is 1, we have $l = 1$. Many designs work for this example. Let us follow the design given in the proof of the theorem: by noticing that A_a is already in the controllability canonical form, we can easily choose $B_c^T = C_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$. Thus, the precompensator is

$$\begin{aligned}
\dot{\mathbf{x}}_c &= I_6\mathbf{x}_c + \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T \mathbf{v} \qquad (17.7)\\
\mathbf{u}_c &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}\mathbf{x}_c.
\end{aligned}$$

Example 2: Let us consider cancelling the invariant zeros of a system in which the zero dynamics have a defective eigenvalue. In particular, we modify the example above to have $A_a = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Notice that the zero dynamics have an eigenvalue at 1 with Jordan block of size two in this case.

For this example, the minimal polynomial degree of A_a is 2. Following the design given in the proof of the theorem, we obtain a precompensator with state matrix $diag_m(A_a)$, $B_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^T$, and $C_c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$.

18. A PRE- + POST- + FEEDFORWARD COMPENSATOR DESIGN FOR ZERO PLACEMENT

We demonstrate the design of a pre- + post- + feedforward compensator that places the transmission zeros of a stabilizable and detectable multi-input multi-output linear-time-invariant plant at arbitrary locations.

18.1 Introduction and Problem Formulation

Relocation of a system's finite invariant zeros using feed-forward controller architectures is of importance in several control applications (e.g., [257–261]), including adaptive control and stable-plant-inversion-based design. In particular, *lifting techniques*—which parallelize and combine plant inputs and outputs to achieve zero relocation and annihilation—have been developed for several plant models. While researchers have developed a range of lifting techniques, a systematic methodology for placing invariant zeros at desired locations is not known, and in fact the literature makes clear the complexity of the zero relocation problem [258,260]. In this brief communique, we develop a systematic methodology for relocating the finite invariant zeros of a continuous-time MIMO LTI plant using the time-invariant feed-forward control architecture shown in Figures 18.1 and 18.2. Our zero-relocation methodology exploits the *special coordinate basis* (SCB) for linear systems [196,205].

Precisely, let us consider an arbitrary stabilizable and detectable linear time-invariant plant:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{18.1}$$

$$\mathbf{y} = C\mathbf{x},$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times m}$, and $\mathbf{x} \in R^n$, $\mathbf{u} \in R^m$ and $\mathbf{y} \in R^p$ are the state, input, and output, respectively. We denote the plant in Equation 18.1 by Σ_P.

We will demonstrate that precompensation plus postcompensation plus feedforward compensation can be used to place the transmission zeros of the plant at will. That is, we will design compensators Σ_{pre}, Σ_{post}, and Σ_{ff} such that $\Sigma = \Sigma_{pre}\Sigma_P\Sigma_{post} + \Sigma_{ff}$ (the cascade of Σ_P with Σ_{pre} and Σ_{post} all in parallel with Σ_{ff}) is stabilizable and detectable, and has transmission zeros at a set of desired locations. The pre- + post- + feedforward compensator is illustrated in Figure 18.1.

18.2 Main Result

In this section, we elucidate our zero-relocation design that combines the common approach of squaring-down and rank uniformization with a specific feedforward controller design.

Before presenting the main theorem on the complete compensator design, let us quote two classical lemmas that together demonstrate that a LTI plant Σ_p can be made square-invertible and uniform rank by adding dynamic compensation. Lemma 18.1 considers design for square-invertibility, see Theorems 3.1 and 3.2 in [238] for the full derivation of the design.

Lemma 18.1. (Squaring Down) *Consider the stabilizable and detectable LTI plant $\sum_P$ (Equation 18.1). Proper pre- and post-compensators can be designed, so that the resulting plant is 1) square invertible and 2) stabilizable and detectable. The compensated system has the same number of inputs as outputs, and this number is equal to the normal rank of the original plant. The com-*

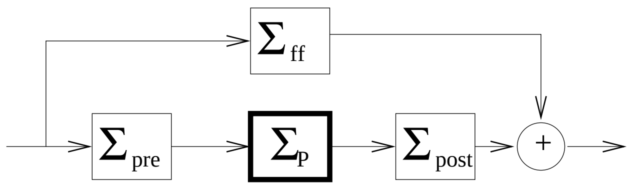


Fig. 18.1: The pre- + post- + feedforward architecture for zero relocation is illustrated.

pensators induce an additional set of invariant zeros in the squared-down system, whose locations can be designed.

The algorithm for designing the squaring-down precompensator can be found in [238]. Upon squaring down, the finite invariant zero structure is expanded by z open left half plane (OLHP) transmission zeros, where z is the order of the non-invertible dynamics; meanwhile, the infinite-zero structure is unchanged.

Lemma 18.2 considers adding further pre-compensation to make the plant uniform rank. Specifically, for a non-uniform rank system (one which has different lengths of infinite zero chains), the shorter chains can be extended by adding integrators at the input side, and hence the system can be made uniform rank without altering the finite invariant zero dynamics. Please see Proposition 4 in [206] for the derivation.

Lemma 18.2. (Rank Uniformization) *Consider a stabilizable and detectable square-invertible plant. A proper pre-compensator can be designed so that the resulting system is stabilizable and detectable, uniform rank, and its invariant zeros are the same as those of the original plant.*

The algorithm for designing the rank-uniformizing pre-compensator is given in [206]. Now we are ready to state the main result of chapter on zero relocation.

Theorem 18.3. *Consider the stabilizable and detectable LTI plant $\sum_P$ (Equation 18.1). A pre- + post- + feedforward compensator of the form shown in Figure 18.1 can be designed, so that the*

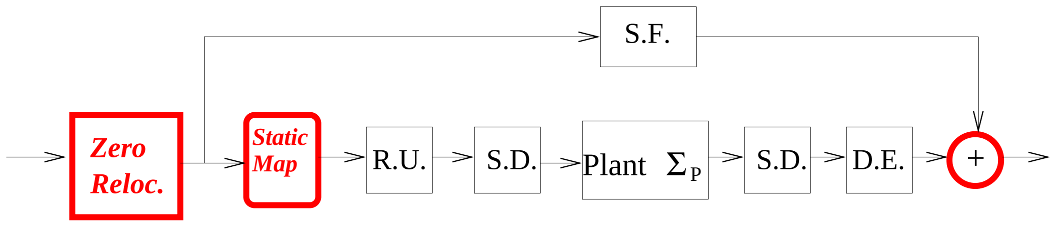


Fig. 18.2: A detailed diagram of the pre- + post- + feedforward controller for zero placement is shown. Here, $S.D.$ indicates compensation for squaring down, $R.U.$ represents a rank-uniformizing compensator, $D.E.$ represents a derivative-estimation filter (specifically one with transfer function $\frac{s^{q-1}}{(1+\epsilon s)^{q-1}}I$ where ϵ is small and q is the relative degree of the rank-uniformized plant), $S.F.$ is a smoothing filter with transfer function $\frac{1}{(1+\epsilon s)^{q-1}}I$, and the zero-relocation filter together with the static map permit arbitrary placement of the plant transmission zeros.

compensated plant remains stabilizable/detectable, and is 1) square-invertible, 2) uniform rank, and 3) minimum phase. In fact, the m transmission zeros of the compensated plant can be placed at any set of locations $x_1, ..., x_m$ that are closed under conjugation. Furthermore, Σ_{pre} (which is of dimension $r \times m$, where r is the normal rank of the system), Σ_{post} (which is of dimension $p \times r$), and Σ_{ff} (which is of dimension $r \times r$) are all proper.

Proof:

We shall give an explicit construction of the controller, by designing the compensator blocks around the plant, as shown in Figure 18.2.

The two lemmas demonstrate the design to make the LTI plant Σ_p square-invertible and uniform rank while maintaining stabilizability/detectability by adding dynamic compensation (blocks $S.D.1$ and $S.D.2$ for achieving square invertibility and $R.U.$ for rank uniformization). Henceforth in this proof, we only consider zero relocation of a square-invertible and uniform-rank system.

Let us denote the input of the uniform-rank square-invertible system as $\mathbf{u} \in R^r$, the output as

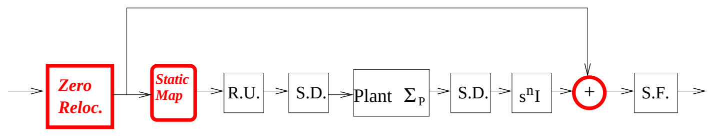


Fig. 18.3: This equivalent block diagram clarifies that the estimation can be viewed as a pure derivative computation, with a smoothing filter after the feedforward addition.

$\mathbf{y} \in R^r$, and the relative degree as q. The system can be written in the Special Coordinate Basis (SCB) as (see [196, 205] for details):

$$\dot{\mathbf{x}}_a = A_a\mathbf{x}_a + A_1\mathbf{x}_1 \tag{18.2}$$

$$\dot{\mathbf{x}}_1 = \mathbf{x}_2$$

$$\vdots$$

$$\dot{\mathbf{x}}_q = E_a\mathbf{x}_a + L(\mathbf{x}_1, ...\mathbf{x}_q) + CA^{q-1}B\mathbf{u}$$

$$\mathbf{y} = \mathbf{x}_1$$

where $A_a \in R^{n_a \times n_a}$, $E_a \in R^{r \times n_a}$ and $L()$ denotes a linear function of the elements in (). Here, the triple (E_a, A_a, A_1) specifies the zero-dynamics of the system, and the system zeros are the eigenvalues of A_a.

Next, let us use post-compensation to obtain the output-derivative x_q from the output y, using blocks $D.E.$ and $S.F.$ in Figure 18.2. To obtain x_q from y, we should use an estimator block with transfer function $s^{q-1}I$, however we require a proper LTI compensator. As an alternative, let us use a high-gain estimator, for instance one with transfer function $\frac{s^{q-1}}{(1+\epsilon s)^{q-1}}I$ with small ϵ, together with a smoothing filter with transfer $\frac{1}{(1+\epsilon s)^{q-1}}I$ on the feedforward path. We notice that this scheme is equivalent to using a pure derivative estimator $s^{q-1}I$ before the addition of the feedforward signal, together with a filter with transfer function $\frac{1}{(1+\epsilon s)}^{q-1}I$ after the addition (see Figure 18.3). Thus, the use of the high-gain estimator only serves to introduce a set of poles that are far in the OLHP; we can thus continue the analysis from here on assuming use of the pure derivative compensator. We notice that this compensator serves to cancel the infinite zeros of the plant, and does not introduce any new finite invariant zeros. Upon compensation, we can view the dynamics as being the same as the above one, but with output $\hat{y} = x_q$. Upon reformulation, a portion of the infinite zero chains (of length $q-1$) is attached to the previous zero dynamics at its input side and hence, clearly, the new system is uniform rank-1 and remains square-invertible, stabilizable and detectable. The SCB of the new system is:

$$\begin{aligned}
\dot{\hat{\mathbf{x}}}_{\mathbf{a}} &= \hat{\mathbf{A}}_{\mathbf{a}}\hat{\mathbf{x}}_{\mathbf{a}} + \hat{\mathbf{A}}_{\mathbf{1}}\hat{\mathbf{x}}_{\mathbf{1}} \\
\dot{\hat{\mathbf{x}}}_{\mathbf{1}} &= \hat{\mathbf{E}}_{\mathbf{a}}\hat{\mathbf{x}}_{\mathbf{a}} + \hat{\mathbf{E}}_{\mathbf{1}}\hat{\mathbf{x}}_{\mathbf{1}} + \hat{\mathbf{C}}\mathbf{B}\mathbf{u} \\
\hat{\mathbf{y}} &= \hat{\mathbf{x}}_1,
\end{aligned} \tag{18.3}$$

where $\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_a & \mathbf{x}_1 & \dots & \mathbf{x}_{q-1} \end{bmatrix}^T$, $\hat{\mathbf{x}}_1 = \mathbf{x}_q$, and $\hat{A}_a$, $\hat{A}_1$, $\hat{E}_a$, $\hat{E}_1$ and $\hat{C}$ can be obtained from Equation 18.2 directly. We see that the zeros of the new system contain all of the previous zeros plus a number $q-1$ of zeros at the origin. Moreover, the zero dynamics $(\hat{A}_a, \hat{A}_1, \hat{E}_a, \hat{E}_1)$ is stabilizable and detectable. This is because the stabilizability and detectability of the reconstructed system in Equation 18.3 implies that the transmission zeros are controllable and observable modes of the zero dynamics [205] and the decoupling zeros are in OLHP.

Let us construct the feed-forward compensator as

$$\begin{aligned}
\dot{\mathbf{x}}_c &= A_c\mathbf{x}_c + B_c\mathbf{v} \\
\mathbf{y}_c &= C_c\mathbf{x}_c \\
\mathbf{u} &= G\mathbf{x}_c + F\mathbf{v} \\
\tilde{\mathbf{y}} &= \bar{\mathbf{y}} + \mathbf{y}_c
\end{aligned} \tag{18.4}$$

where $\tilde{\mathbf{y}}$ is the new system output, $G = (\hat{C}B)^{-1}C_cA_c$, and $F = (\hat{C}B)^{-1}(C_cB_c - I)$. Notice that this feed-forward compensator is shown in the blocks labeled Static Map and Zero Reloc., as well as in the summation at the output in Figure 18.2. Let us show that the system with this pre-compensator can be designed to be minimum-phase (and in fact to have transmission zeros at arbitrary locations). To do so, let us find the zeros of the system with this pre-compensator. By taking $\tilde{y} = 0$, and $\dot{\tilde{y}} = 0$, we see that the zero dynamics is

$$\begin{bmatrix} \dot{\mathbf{x}}_a \\ \dot{\mathbf{x}}_c \end{bmatrix} = \begin{bmatrix} \hat{A}_a & \hat{A}_1 C_c \\ B_c \hat{E}_a & A_c + B_c \hat{E}_1 C_c \end{bmatrix} \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_c \end{bmatrix}. \tag{18.5}$$

Clearly, when the system $(\hat{A}_a, \hat{A}_1, \hat{E}_a, \hat{E}_1)$ is detectable and stabilizable, a dynamic controller (A_c, B_c, C_c) exists to arbitrarily relocate the zeros, and consequently stabilize the zero dynamics. Hence, the proof is complete.

In this theorem, we gave a systematic controller design that moves the invariant zeros of a general stabilizable and detectable LTI plant to arbitrary locations. It is known that invariant zeros are invariant under feedback control (both state feedback and output feedback), and so our zero relocation through use of new controller architecture is significant. Our design is based on the smart construction of a pre- + post- + feedforward compensator, which equivalences the zero-relocation problem with a feedback controller design problem directly of the zero dynamics. Let us make a few further comments about our design:

1) In essence, the estimation of $x_{q-1} = \mathbf{y}^{(q-1)}$ from $\mathbf{y}$ (represented by the block D.E. in Figure 18.2) is straightforward, since the quantity to be estimated is part of the infinite zero structure. As a note, one need not place the smoothing filter $\frac{1}{(1+\epsilon s)^{q-1}}$ in the feedfoward path. In this case, we can use a time-scaling approach [195] to show that the fast dynamics introduced by the estimator only has minor impact to the system dynamics (specifically, moving existing zeros only slightly and introducing highly stable zeros). As an alternative to the high-gain

estimator presented in the proof, a multiple-delay approximation can also be used for estimation [13]. It is worth noting that, for uniform-rank plants with relative degree 1, derivative estimation is not needed at all. Whatever method is used, the estimation of the state from observations may be susceptible to sensor noise, as is true in all controller designs. In that estimation is only required on a feedforward path, while any subsequent controller design is based on a minimum-phase plant, the estimation required in the zero-relocation design can sensibly be implemented.

2) The result presented in this chapter is applicable to many controller design problems. For instance, it permits stabilization directly through use of a high gain output-feedback control, which relies on the fact that the plant is minimum-phase. Such a direct output-feedback methodology is especially valuable in the context of decentralized control, where the standard paradigm of estimation followed by state feedback fails [10, 215]. Zero relocation is also needed for e.g. adaptive control of nonminimum phase plants and for plant-inverse controller design [257–261].

3) The pre-, post-, and feedforward controller gains can be straightforwardly found in practice, using standard software for computing the SCB (as documented in [262]) together with simple algebraic manipulations.

19. AN ALTERNATIVE APPROACH TO DESIGNING STABILIZING COMPENSATORS FOR SATURATING LINEAR TIME-INVARIANT PLANTS

We present a new methodology for designing low-gain linear time-invariant (LTI) controllers for semi-global stabilization of an LTI plant with actuator saturation, that is based on representation of a proper LTI feedback using a precompensator plus static-output-feedback architecture. We also mesh the new design methodology with time-scale notions to develop lower-order controllers for some plants.

19.1 Introduction

Low-gain output feedback stabilization of linear time-invariant plants subject to actuator saturation has been achieved using the classical observer–followed–by–state–feedback controller architecture [198, 263, 264]. In this note, we discuss an alternative controller architecture for designing low-gain output feedback control of linear time-invariant (LTI) plants with saturating actuators. Specifically, we use a classical result of Ding and Pearson to show that a dynamic prefiltering together with static output feedback architecture can naturally yield a stabilizing low-gain controller under actuator saturation (Section 19.2). Subsequently, by using time-scale notions, we illustrate through a single-input single-output (SISO) example that lower-order controllers can be designed, in the case where the $j\omega$-axis eigenvalues of the plant are in fact at the origin (Section 19.3).

The reader may wonder what advantage the alternate architectures provide. Our particular motivation for developing the alternatives stems from our ongoing efforts on decentralized controller design, and in particular our effort to develop a low-gain methodology for decentralized plants [10, 16, 17, 25, 215, 222]. In pursuing this goal, we have needed to use several novel controller structures, in particular ones that utilize pre-compensators and output derivatives together (see our works in [10, 16, 17, 215]). This chapter delineates the particular use of the new controller atchitectures in stabilization under saturation.

What this study of decentralized control makes clear is that freedoms in the structure of the controller facilitate design, because they can permit design that fit the structural limitations of the problem (in this case, decentralization). The alternatives to low-gain control that we propose here serve this purpose, because they naturally permit selection of a desirable controller architecture for the task at hand. While our primary motivation is in the decentralized controls arena, we believe that these alternate architectures may also be useful in such domains as adaptive control and plant inversion through lifting [17].

19.2 Low-Gain Output Feedback Control through Precompensation

In this section, we demonstrate design of low-gain proper controllers for semi-global stabilization of LTI plants subject to actuator saturation, using a novel precompensator-based architecture. We also briefly discuss the connection of our design to the traditional observer-based design, and expose that the design is deeply related to a family of precompensator-based designs that also permit e.g. zero cancellation and relocation.

Formally, we demonstrate design of a proper output feedback compensator that achieves semi-global stabilization of the following plant $\mathcal{G}$:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\sigma(\mathbf{u}) \tag{19.1}$$
$$\mathbf{y} = C\mathbf{x},$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, $C \in R^{p\times n}$, $\sigma()$ is the standard saturation function, A has eigenvalues in the closed left-half-plane (CLHP), and the triple (C, A, B) is observable and controllable[*]. Our design is fundamentally based on 1) positing a control architecture comprising a pre-compensator with a zero-free and uniform-rank structure together with a feedback of the output and its derivatives (see Figure 19.1), 2) designing the controller using this architecture, and 3) arguing that the designed controller admits a strictly proper feedback implementation. This controller design directly builds on two early results: 1) Ding and Pearson's result [265] for pole-placement that is based on a dynamic pre-compensation + static feedback representation of a proper controller (Figure 19.1a and 19.1b); and 2) Lin and Saberi's effort [198] on stabilization under saturation using state feedback. For clarity, we cite the two results in the lemmas before we present our main result.

Lemma 19.1 concerns pre-compensator and feedback design for pole placement in a general LTI system, i.e. one of the form

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{19.2}$$
$$\mathbf{y} = C\mathbf{x},$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, and $C \in R^{p\times n}$.

[*] In fact, the methods developed here trivially generalize to the case where the dynamics are stabilizable and detectable. We consider the observable and controllable case for the sake of clarity.

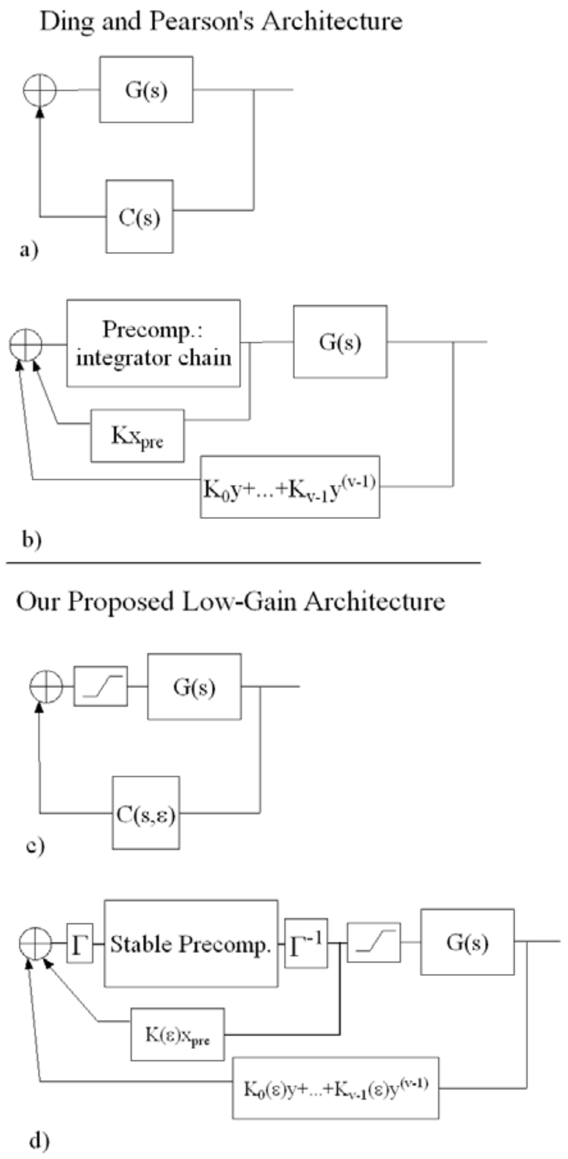

Ding and Pearson's Architecture
G(s)
C(s)
a)
Precomp.: integrator chain
G(s)
Kx_{pre}
$K_0y+...+K_{v-1}y^{(v-1)}$
b)
Our Proposed Low-Gain Architecture
G(s)
C(s,ε)
c)
Γ
Stable Precomp.
Γ^{-1}
G(s)
$K(\varepsilon)x_{pre}$
$K_0(\varepsilon)y+...+K_{v-1}(\varepsilon)y^{(v-1)}$
d)

Fig. 19.1: Compensator artechitectures: a) and b) show the compensator artechitectures presented in Ding and Pearson [265], in particular, a precompensator-together-with-static feedback viewpoint. (b) is used to design a proper compensator of (a). c) and d) show the compensator artechitectures that stabilize a plant under input saturation.

Lemma 19.1. *Consider a plant of the form (19.2) that is controllable and observable, with observability index v. Pre-compensation through addition of $v-1$ integrators to each plant input permits computation of the plant's state $\mathbf{x}$ as a linear function of the plant's output $\mathbf{y}$, its derivatives up to $\mathbf{y}^{(v)}$ and the pre-compensator's state. A consequence of this computation capability is that it permits design of a strictly proper feedback controller $C(s)$ that places the poles of the compensated plant at arbitrary locations (that are closed under conjugation).*

When the matrix C in the system (19.2) is not invertible, the classical method to obtain the state information from output is through observer design. This lemma of Ding and Pearson gives an alternative design for state estimation and feedback controller design, that is based on viewing certain proper compensators $C(s)$ as a dynamic pre-compensation together with static feedback (Figure 19.1a and 19.1b). Specifically, the methodology of design is as follows: first, from the pre-compensator-based representation (Figure 19.1b), a computation of the plant state from the plant output and its derivatives together with the precompensator state can directly be obtained. Second, the classical state feedback methodology thus permits us to compute the static feedback in the pre-compensator-based representation, so as to place the closed-loop eigenvalues at desired locations. Third, the equivalence between the precompensator-based representation and a proper feedback controller is used to obtain a realization of the feedback control (Figure 19.1a). We kindly ask the reader to see [265, 266], both for the details of the state computation and the equivalence between the precompensator-based architecture and the proper feedback controller. In our development, we broadly replicate the design methodology of Ding and Pearson, but use a stable rather than neutral precompensator in order to obtain a controller that works under input saturation.

Lemma 19.2 is concerned with using linear state feedback control to semi-globally stabilize the plant:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\sigma(\mathbf{u}), \tag{19.3}$$

where $A \in R^{n\times n}$, $B \in R^{n\times m}$, and $\sigma()$ is the standard saturation function. Please see Lin and Saberi's work [198] for the proof of the lemma.

Lemma 19.2. *Consider a plant of the form (19.3) that satisfies two conditions: 1) all the eigenvalues of A are located in the CLHP; 2) (A, B) is stabilizable. Then the plant can be semiglobally stabilized using linear static state feedback. That is, a parametrized family of compensators $u = K(\epsilon)x$ can be designed such that, for any specified ball of plant initial conditions $\mathcal{W}$, there exists $\epsilon^*(\mathcal{W})$ such that, for all $0 < \epsilon \leq \epsilon^*(\mathcal{W})$, the compensator $K(\epsilon)$ achieves local exponential stabilization of the origin and contains $\mathcal{W}$ in its domain of attraction.*

Now we are ready to present the main result. Specifically, the following theorem formalizes that a family of proper controllers can be designed for semi-global stabilization of $\mathcal{G}$, based on the precompensator-together-with-derivative-feedback architecture shown in Figure 19.1. The proof of the theorem makes clear the design methodology.

Theorem 19.3. *The plant $\mathcal{G}$ (Equation 19.1) can be asymptotically semi-globally stabilized using proper feedback compensation of order mv, where v is the observability index of the plant. Specifically, a parametrized family of compensators $C(s, \epsilon)$ can be designed (Figure 19.1c) to achieve the following: for any specified ball of plant and compensator initial conditions $\mathcal{W}$, there exists $\epsilon^*(\mathcal{W})$ such that, for all $0 < \epsilon \leq \epsilon^*(\mathcal{W})$, $C(s, \epsilon)$ makes the origin locally exponentially stable and contains $\mathcal{W}$ in its domain of attraction. The design can be achieved by developing a controller of the architecture shown in Figure 19.1d—i.e., comprising an m-input uniform-rank* *square-invertible zero-free precompensator P with input $\mathbf{u}_p$ together with a feedback of the form $\mathbf{u}_p = K_0(\epsilon)\mathbf{y} + K_1(\epsilon)\mathbf{y}^{(1)} + \ldots + K_{v-1}(\epsilon)\mathbf{y}^{(v-1)}$ (where $K_0(\epsilon), \ldots, K_{v-1}(\epsilon)$ are matrices of dimension $m \times p$)—and then constructing a proper implementation.*

Proof:

We shall prove that, for the given ball of initial conditions, a family of proper compensaters $C(s,\epsilon)$ can be designed so that the actuator does not saturate, and further the closed-loop system without saturation is exponentially stable. Together, these two aspects show that the origin is locally exponentially stable with $\mathcal{W}$ in the domain of attraction. We first note that, as long as the compensator permits a proper state-space implementation and the system operates in the linear regime, the additive contribution of the compensator's initial condition on the input can be made arbitrarily small through pre- and post-scaling of the compensator by a large gain Γ and its inverse (see Figure 19.1). Thus, WLOG, we seek to verify that $||\mathbf{u}||_\infty < 1$ for the ball of initial states and assuming null compensator initial conditions. To do so, we will design a compensator of the architecture shown in Figure 19.1 that achieves the design goals, and then note a proper implementation.

To do this, let $\tilde{P}$ be any asymptotically stable LTI system of the following form:

$$\begin{bmatrix} \mathbf{y}_{\tilde{P}}^{(1)} \\ \mathbf{y}_{\tilde{P}}^{(2)} \\ \vdots \\ \mathbf{y}_{\tilde{P}}^{(v)} \end{bmatrix} = \begin{bmatrix} & I_m & & \\ & & \ddots & \\ & & & I_m \\ \tilde{Q}_0 & \ldots & \ldots & \tilde{Q}_{v-1} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\tilde{P}} \\ \mathbf{y}_{\tilde{P}}^{(1)} \\ \vdots \\ \mathbf{y}_{\tilde{P}}^{(v-1)} \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_m \end{bmatrix} \mathbf{u}_{\tilde{P}}, \tag{19.4}$$

where $\mathbf{u}_{\tilde{P}} \in R^m$ and $\mathbf{y}_{\tilde{P}} \in R^m$ are the input and output to $\tilde{P}$. Notice that $\tilde{P}$ is square-invertible, zero-free, and uniform rank. Let us denote the ∞-norm gain of this plant as q.

Let us first consider pre-compensating the plant G using $\tilde{P}$, and then using feedback of the first v derivatives of the output along with the states of the precompensator (see Figure 19.1). That is, upon precompensation with $\tilde{P}$, we consider using a feedback controller of the form $\mathbf{u}_{\tilde{P}} = \sum_{i=0}^{v-1} K_i \mathbf{y}^{(i)} + \sum_{i=0}^{v-1} \tilde{K}_i \mathbf{y}_{\tilde{P}}^{(i)}$, where we have presciently used the notation K_i for the output-derivative feedbacks since these will turn out to be the gains in the conpensator diagrammed in Figure 19.1d, and where we suppress the dependence on ϵ in our notation for the sake of clarity. For convenience, let us define $K = \begin{bmatrix} K_0 & \dots & K_{v-1} \end{bmatrix}$, $\tilde{K} = \begin{bmatrix} \tilde{K}_0 & \dots & \tilde{K}_{v-1} \end{bmatrix}$, $\mathbf{y}(ext) = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(v-1)} \end{bmatrix}$, and $\mathbf{y}_{\tilde{P}}(ext) = \begin{bmatrix} \mathbf{y} \\ \mathbf{y}_{\tilde{P}}^{(1)} \\ \vdots \\ \mathbf{y}_{\tilde{P}}^{(v-1)} \end{bmatrix}$. In this notation, the controller becomes $\mathbf{u}_{\tilde{P}} = \begin{bmatrix} K & \tilde{K} \end{bmatrix} \begin{bmatrix} \mathbf{y}(ext) \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix}$.

We claim that such a controller can be designed, so that 1) the closed-loop system without saturation is exponentially stable, 2) $||u_{\tilde{P}}||_\infty \leq \epsilon$ for the given ball of plant initial conditions and any $0 < \epsilon \leq \frac{0.9}{q}$, and 3) the controller gains K and $\tilde{K}$ are $\mathcal{O}(\epsilon)$. To see why, first note that, based on the fact that the relative degree of the precompensator equals the observability index, the state of the pre-compensated system $\widehat{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix}$ is a linear function of $\begin{bmatrix} \mathbf{y}(ext) \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix}$. In particular, it is automatic that $\mathbf{x} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix} = Z \begin{bmatrix} \mathbf{y}(ext) \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix}$, where Z has the form $\begin{bmatrix} Z_1 & Z_2 \\ 0 & I \end{bmatrix}$, see Ding and Pearson's development [265] for the method of construction. Next, from Lemma 19.2, we see that a low-gain full state-feedback controller $\widehat{K}(\epsilon)$ of order ϵ can be developed for the precompensated plant, that achieves local exponential stabilization of the origin and also makes the ∞-norm of the input less than ϵ for any $\epsilon > 0$, for the given ball of plant initial conditions. Thus, by applying the feedback $\widehat{K}(\epsilon)Z \begin{bmatrix} \mathbf{y}(ext) \\ \\ \mathbf{y}_{\tilde{P}}(ext) \end{bmatrix}$, we can meet the three desired objectives.

It remains to be shown that the plant input $\mathbf{u}$ does not saturate upon application of this compensation. To do so, simply note that $||\mathbf{u}||_\infty \leq q||u_{\tilde{P}}||_\infty \leq 0.9$.

We can absorb the feedback of $\mathbf{y}_{\tilde{P}}(ext)$ into the pre-compensator, so that we obtain a control scheme comprising a precompensator P with dynamics

$$\begin{bmatrix} \mathbf{y}_P^{(1)} \\ \mathbf{y}_P^{(2)} \\ \vdots \\ \mathbf{y}_P^{(v)} \end{bmatrix} = \begin{bmatrix} & I_m & & \\ & & \ddots & \\ & & & I_m \\ \tilde{Q}_0 + \tilde{K}_0 & \dots & \dots & \tilde{Q}_{v-1} + \tilde{K}_{v-1} \end{bmatrix} \begin{bmatrix} \mathbf{y}_P \\ \mathbf{y}_P^{(1)} \\ \vdots \\ \mathbf{y}_P^{(v-1)} \end{bmatrix} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I_m \end{bmatrix} \mathbf{u}_P \tag{19.5}$$

together with feedback $\mathbf{u}_P = \sum_{i=0}^{v-1} K_i \mathbf{y}^{(i)}$.

Finally, exactly analogously to the design method in [265]. we see automatically that the transfer function from $\mathbf{y}$ to $\mathbf{u}$ is in fact strictly proper, and so the design admits a proper state-space implementation. Through appropriate scaling of the compensator, we thus see that saturation is avoided for the ball of plant and compensator initial conditions, while the dynamics without saturation are exponentially stable. Thus, semi-global stabilization has been verified.

We have given an alternative low-gain controller design for semi-global stabilization under saturation. It is worth stressing that the crux of the design is the ability to construct the plant's full state as a static mapping of output derivatives together with precompensator variables, upon adequate dynamic precompensation. This observation yields a design strategy where a dynamic precompensator's impulse response is designed followed by low-gain static state feedback, with the goal of ensuring that the output of their cascade is small (for the given ball of initial conditions).

This is a different viewpoint from the traditional one in limited-actuation output-feedback design [198,222,263], where actuation capabilities are divided between observation and state-feedback tasks.

Remark: By choosing the $\tilde{Q}_i$ appropriately, we can set the gain q of the precompensator $\tilde{P}$ to an arbitrary value. Appropriate selection of the precompensator can potentially facilitate selection of more numerically-stable feedback gains, by permitting a larger input prior to the precompensator. We leave a careful analysis to future work.

19.3 A Compensator that Exploits Time-Scale Structure

Our philosophy for low-gain control using a precompensation-plus-feedback architecture also permits construction of stabilizers that exploit time-scale structure in the plant. Specifically, we here demonstrate design of precompensators for semi-global stabilization of the plant $\mathcal{G}$, that are generally lower-order than those in Section 19.2 because they exploit time-scale separation in the plant. Conceptually, when stabilization under saturation is the goal, low-gain state feedback only need be provided for the plant dynamics associated with jw-axis eigenvalues (see e.g. [222] for use of this idea in observer-based designs). In the case where these eigenvalues are at the origin, the corresponding dynamics are in fact the slow dynamics of the system. Thus, through time-scale separation, we can design precompensation together with feedback so as to stabilize the *slow* dynamics under actuator saturation, and then obtain a proper feedback implementation. The use of time-scale separation ideas in the precompensator-based design becomes rather intricate, and so we illustrate the design only for SISO plants for the sake of clarity. We shall use standard singular-perturbation notions to prove the result. Here is a formal statement:

Theorem 19.4. *Consider a plant $\mathcal{G}$ (as specified in Equation 19.1) that further is SISO, and has*

q poles at the origin. This plant can be semi-globally stabilized under actuator saturation using a proper dynamic compensator of order q.

Proof:

We shall prove that, for any given ball of plant and controller initial conditions, there exists a proper compensator of order q such that 1) the closed-loop system without saturation is exponentially stable and 2) actuator saturation does not occur. Together, these observations yield that the origin is locally exponentially stable for the given ball of initial conditions, and hence that semi-global stabilization is achieved.

To this end, let us begin by denoting the the transfer function of the plant's linear dynamics by $G(s)$. We note that the transfer function can be written as $G(s) = \frac{b_m s^m + \ldots + b_0}{s^q(s^{n-q} + a_{n-q-1}s^{n-q-1} + \ldots + a_0)}$, where m is the number of plant zeros. We find it easiest to conceptualize the compensator as comprising a zero-free dynamic precompensator of order q, together with a feedback of the output y and its first $q-1$ derivatives, as shown in Figure 19.1. We choose the precompensator to be *any* stable system of this form. We denote the precompensator's transfer function by $C_p(s) = \frac{1}{(s^q + c_{q-1}s^{q-1} + \ldots + c_0)}$, and the feedback controller by $K(s) = k_{q-1}s^{q-1} + \ldots + k_0$. We note the entire compensator $K(s)C_p(s)$ has order q and is proper.

With a little algebra, we find that the characteristic polynomial of the closed-loop system is $p(s) = s^q(s^{n-q} + a_{n-q-1}s^{n-q-1} + \ldots + a_0)(s^q + c_{q-1}s^{q-1} + \ldots + c_0) + (b_m s^m + \ldots + b_0)(k_{q-1}s^{q-1} + \ldots + k_0)$. We note that the polynomial $p_f(s) = (s^{n-q} + a_{n-q-1}s^{n-q-1} + \ldots + a_0)(s^q + c_{q-1}s^{q-1} + \ldots + c_0)$ has roots in the OLHP, by assumption, for the chosen stable pre-compensator $C_p(s)$.

Let us now consider a family of multiple derivative output feedbacks, parameterized by a low-gain parameter $\epsilon > 0$. In particular, let us consider feedback with $k_i = \frac{a_0 c_0}{b_0}\gamma_i \epsilon^{q-i}$, where $s^q + \gamma_{q-1}s^{q-1} + \ldots + \gamma_0$ is a stable polynomial with roots $\lambda_1, \ldots, \lambda_q$, and ϵ is a low-gain parameter.

We will verify that the characteristic polynomial has n roots that are within $\mathcal{O}(\epsilon)$ of the roots of $p_f(s)$, while the remaining q roots are within $\mathcal{O}(\epsilon^2)$ of $\epsilon\lambda_1, \dots, \epsilon\lambda_q$. To prove this, notice first that $p(s) = s^q p_f(s) + (b_m s^m + \dots + b_0)(\gamma_{q-1}\epsilon s^{q-1} + \dots + \gamma_0 \epsilon^q)\frac{a_0 c_0}{b_0}$. Noting that the entire second term in this expression is $\mathcal{O}(\epsilon)$, we see that the roots of $p(s)$ are $\mathcal{O}(\epsilon)$ perturbations of the roots of $s^q p_f(s)$. Thus, we see that n roots are within $\mathcal{O}(\epsilon)$ of the roots of $p_f(s)$, while the remaining are within $\mathcal{O}(\epsilon)$ of the origin.

To continue, let us consider the change of variables $\overline{s} = \frac{\epsilon}{s}$. Substituting into the closed-loop characteristic polynomial, we find that $p(\overline{s}) = (\frac{\epsilon}{\overline{s}})^q p_f(\overline{s}) + \epsilon^q (b_m \frac{\epsilon^m}{\overline{s}^m} + \dots + b_0)(\frac{\gamma_{q-1}}{\overline{s}^{q-1}} + \dots + \gamma_0)\frac{a_0 c_0}{b_0}$. Scaling the expression by $\frac{\overline{s}^{n+q}}{a_0 c_0}$, we obtain that the expression $p(\overline{s}) = 0$ is the following degree-$(n+q)$ polynomial equation in $\overline{s}$: $\epsilon^q(\gamma_0 \overline{s}^{n+q} + \dots + \gamma_{q-1}\overline{s}^{n+1}) + \epsilon^q \overline{s}^n + r(\overline{s}) = 0$, where $r(\overline{s})$ is a polynomial in $\overline{s}$ of degree no more that $\overline{s}^{n+q-1}$ with each term scaled by a coefficient of order ϵ^{q+1} or smaller. Thus, dividing by ϵ^q, we find that the solutions $\overline{s}$ to the equation are within $\mathcal{O}(\epsilon)$ of the solutions to $\gamma_0 \overline{s}^{n+q} + \dots + \gamma_{q-1}\overline{s}^{n+1} + \overline{s}^n = 0$. However, the roots of this equation are precisely $\frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_q}$, as well as 0 repeated n times. Noting that $s = \frac{\epsilon}{\overline{s}}$, we thus recover that q roots of the characteristic polynomial are within $\mathcal{O}(\epsilon^2)$ of $\epsilon\lambda_1, \dots, \epsilon\lambda_q$. Thus, we have characterized all the poles of the closed-loop system. We notice that all the poles are guaranteed to be within the OLHP.

Now consider the response for a ball of initial conditions $\mathcal{W}$. As in the proof of Theorem 19.3, we notice that the initial state of the precompensator is of no concern in terms of causing saturation, since the precompensator can be pre- and post-scaled by an arbitrary positive constant. Thus, WLOG, let us consider selecting among the family of compensators, to avoid saturation for a given ball of plant initial conditions and assuming zero precompensator initial conditions. Through consideration of the closed-loop dynamics associated with the slow eigenvalues $(\epsilon\lambda_1, \dots, \epsilon\lambda_q)$, we recover immediately (see the proof of Lemma 19.1 in [198]) that, for any specified ball of initial conditions, $||y^{(i)}(t)||_\infty$, is at most of order $\frac{1}{\epsilon^{q-1-i}}$ for $i = 1, \ldots, q-1$. Thus, from the expression for the feedback controller, we find that the maximum value of the precompensator input $\overline{u}$ is $\mathcal{O}(\epsilon)$, say $v_1\epsilon + \mathcal{O}(\epsilon^2)$ for the given ball of plant initial conditions. Furthermore, the stable precompensator imparts a finite gain, say v_2, so the maximum value of $u(t)$ is $v_1 v_2 \epsilon + \mathcal{O}(\epsilon^2)$. Thus, for any given ball of initial conditions, we can choose ϵ small enough so that actuator saturation does not occur. Since actuator saturation is avoided and the closed-loop poles are in the OLHP, stability is proved.

Conceptually, the reduction in the controller order permitted by Theorem 19.4 is founded on focusing the control effort on only the slow dynamics of the system. That is, the controller is designed only to place the eigenvalues at the origin at desired locations (that are linear with respect to the low-gain parameter ϵ); simply using small gains enforces that the remaining eigenvalues remain far in the OLHP. Thus, one only needs to add precompensation to permit estimation of the part of the state associated with the slow dynamics. In this way, stability can be guaranteed and saturation avoided, without requiring as much precompensation as would be needed to estimate the whole state.

We notice that the time-scale-based design is aligned with the broad philosophy of our alternative low-gain design, in the sense that it provides freedom in compensator design. In particular, as with the design in Section 19.2, we notice that *any* stable precompensator can be used for the time-scale-exploiting design, and further design of feedback component in the architecture only requires knowlege of the DC gain of the plant.

19.4 Example

In this example, we demonstrate the design of a low-gain controller that semi-globally stabilizes the following plant:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \sigma(\mathbf{u}) \tag{19.6}$$

$$\mathbf{y} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}. \tag{19.7}$$

Specifically, we show that the compensator of the form $\mathbf{u}(t)^{(v)} = \sum_0^{v-1} \mathbf{A}_i \mathbf{u}(t)^{(i)} + \sum_0^{(v-1)} \mathbf{B}_i \mathbf{y}(t)^{(i)}$, where v is the observability index of the plant, can stabilize the plant under saturation. As developed in the chapter, the design is achieved by first designing a pre-compensator together with output feedback control law, and then implementing the controller in the proper feedback representation above. We shall use the notation from the above development in our illustration.

Let us begin with the precompensator-plus-output-feedback design. To begin, we notice that the observability index of this system is 2. As per the proof of Theorem 19.3, let us thus choose $\tilde{P}$ to be

$$\begin{bmatrix} \dot{\mathbf{y}}_{\tilde{P}} \\ \ddot{\mathbf{y}}_{\tilde{P}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{\tilde{P}} \\ \dot{\mathbf{y}}_{\tilde{P}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}_{\tilde{P}} \tag{19.8}$$

The eigenvalues of this system are located at $-1/2 \pm \sqrt{3}/2i$, hence it is clearly asymptotically stable.

Now let us pre-compensate the plant (Equation 19.6) using $\tilde{P}$ and design feedback of the form

$$\mathbf{u}_{\tilde{P}} = \tilde{K}_0(\epsilon)\mathbf{y}_{\tilde{P}} + \tilde{K}_1(\epsilon)\dot{\mathbf{y}}_{\tilde{P}} + K_0(\epsilon)\mathbf{y} + K_1(\epsilon)\dot{\mathbf{y}} \tag{19.9}$$

To design the feedback, we first recover the state of the entire system including the pre-compensator from the outputs $\mathbf{y}_{\tilde{P}}, \dot{\mathbf{y}}_{\tilde{P}}, \mathbf{y}, \dot{\mathbf{y}}$ through linear transformation, and then apply the low-gain state feedback design (see [198]) to shift eigenvalues of the whole system left by $-\epsilon$. Doing so, we have

$$\tilde{K}_0(\epsilon) = \begin{bmatrix} -0.42\epsilon^5 + 0.30\epsilon^4 + 1.8\epsilon^3 - 4.2\epsilon^2 - 0.90\epsilon & 0.082\epsilon^5 - 1.0\epsilon^4 + 5.2\epsilon^3 - 11.4\epsilon^2 + 5.8\epsilon \\ -0.11\epsilon^5 + 0.28\epsilon^4 - 0.55\epsilon^3 - 6.0\epsilon^2 - 2.2\epsilon & 0.024\epsilon^5 - 0.3\epsilon^4 + 1.5\epsilon^3 - 4.3\epsilon^2 - 0.33\epsilon \end{bmatrix},$$

$$\tilde{K}_1(\epsilon) = \begin{bmatrix} -0.0084\epsilon^5 + 0.13\epsilon^4 - 0.66\epsilon^3 + 0.74\epsilon^2 - 5.3\epsilon & 0.029\epsilon^5 - 0.47\epsilon^4 + 2.3\epsilon^3 - 2.5\epsilon^2 + 1.0\epsilon \\ -0.0024\epsilon^5 + 0.039\epsilon^4 - 0.19\epsilon^3 + 0.21\epsilon^2 - 0.95\epsilon & 0.0084\epsilon^5 - 0.13\epsilon^4 + 0.66\epsilon^3 - 0.74\epsilon^2 - 1.70\epsilon \end{bmatrix},$$

$$K_0(\epsilon) = \begin{bmatrix} -0.30\epsilon^5 - 1.2\epsilon^4 - 1.2\epsilon^3 - 0.91\epsilon^2 & -0.71\epsilon^5 + 4\epsilon^4 - 15.7\epsilon^3 + 34\epsilon^2 - 23\epsilon \\ -0.088\epsilon^5 - 0.35\epsilon^4 - 0.35\epsilon^2 - 0.26\epsilon^2 & -1.0\epsilon^5 + 1.18\epsilon^4 - 4.6\epsilon^3 + 10\epsilon^2 - 6.7\epsilon \end{bmatrix},$$

and $K_1(\epsilon) = \begin{bmatrix} 0.39\epsilon^5 - 0.06\epsilon^4 - 3.3\epsilon^3 - 2.5\epsilon^2 - 1.8\epsilon & 0 \\ 0.091\epsilon^5 + 0.020\epsilon^4 - 0.95\epsilon^3 - 0.72\epsilon^2 - 0.53\epsilon & 0 \end{bmatrix}$.

Hence, the control scheme can be viewed as comprising a precompensator P:

$$\begin{bmatrix} \dot{\mathbf{y}}_P \\ \ddot{\mathbf{y}}_P \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} + \tilde{K}_0(\epsilon) & -\mathbf{I} + \tilde{K}_1(\epsilon) \end{bmatrix} \begin{bmatrix} \mathbf{y}_P \\ \dot{\mathbf{y}}_P \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}_P \tag{19.10}$$

and the feedback flow $\mathbf{u}_P = K_0(\epsilon)\mathbf{y} + K_1(\epsilon)\dot{\mathbf{y}}$.

Finally, the above procedure leads to the proper feedback compensator design

$$\mathbf{u}^{(2)}(t) = (-\mathbf{I} + \tilde{K}_0(\epsilon))\mathbf{u}(t) + (-\mathbf{I} + \tilde{K}_1(\epsilon))\mathbf{u}^{(1)}(t) + K_0(\epsilon)\mathbf{y}(t) + K_1(\epsilon)\mathbf{y}^{(1)}(t) \tag{19.11}$$

Now let us show how ϵ can be chosen. WLOG, let us assume that the precompensator initial conditions to nil, with the understanding that scaling of the precompensator (with appropriately revised proper implementation) permits design with non-zero compensator initial conditions. In

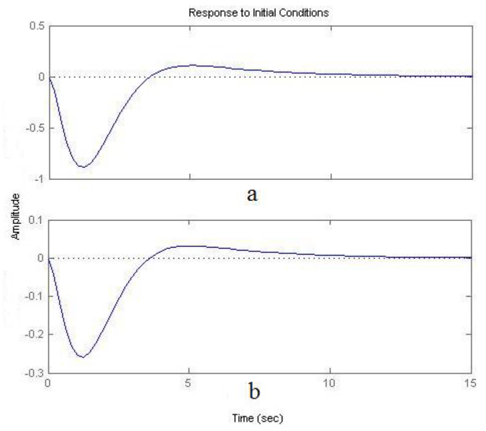


Fig. 19.2: The inputs a) u_1 and b) u_2 are shown, for $\epsilon = 0.5$. Each plant state variable is initialized at 0.99.

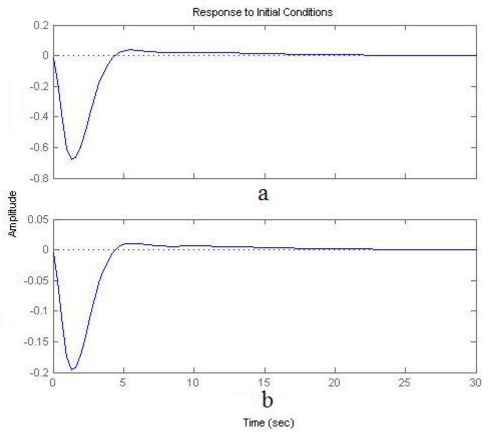


Fig. 19.3: The inputs a) u_1 and b) u_2 are shown, for $\epsilon = 0.25$. Each plant state variable is initialized at 0.99.

particular, consider the case where initial conditions of the plant are in a ball $\mathcal{W}$ with infinity-norm radius 1, i.e., where each initial condition has a magnitude less than or equal to 1. We find that $\epsilon^*(\mathcal{W}) \approx 0.5$ through an exhaustive search. Thus ϵ can be chosen between 0 and 0.5. Trajectories of the two inputs are shown for an initial condition at the edge of the ball, for two different values of ϵ.

PART V: NUMERICS

As all of the earlier parts of the thesis have shown, modern dynamical networks are extraordinarily complex and intricate. While the focus of this thesis has been on addressing design and control in networks analytically (using sensible abstractions of the network dynamics), the sheer scale and complexity of modern networks may force use of numerical methods and simulations in aspects of the design process. Here, we introduce a few numerical and smart-simulation tools for networks, that enrich the analytical design methodology by allowing completion of more intricate network tasks (e.g., network self-partitioning) and permitting evaluation of designs using detailed simulation models.

Part V is organized as follows. Chapter 20 introduces a decentralized network partitioning algorithm that uses a stochastic automaton—the influence model. Chapter 21 gives further studies of the probabilistic collocation method for effective simulation, characterizing its properties and its use with data.

20. A FLEXIBLE STOCHASTIC AUTOMATON-BASED ALGORITHM FOR NETWORK SELF-PARTITIONING

This chapter proposes a flexible and distributed stochastic automaton-based network partitioning algorithm that is capable of finding the optimal k-way partition with respect to a broad range of cost functions, and given various constraints, in directed and weighted graphs. Specifically, we motivate the distributed partitioning (self-partitioning) problem, introduce the stochastic automaton-based partitioning algorithm, and show that the algorithm finds the optimal partition with probability 1 for a large class of partitioning tasks. Also, a discussion of why the algorithm can be expected to find good partitions quickly is included, and its performance is further illustrated through examples. Finally, applications to mobile/sensor classification in ad hoc networks, fault-isolation in electric power systems, and control of autonomous vehicle teams are pursued in detail.

20.1 Introduction

Networks of communicating agents—including sensor networks and autonomous-vehicle teams—require distributed algorithms for a variety of tasks, including data communication/routing, estimation/agreement, and pattern-formation control, among others (see [267] and [142] for interesting overviews). In this chapter, we put forth the perspective that algorithms for network *self-partitioning* or *self-classification*, i.e. algorithms using which a network's nodes can form groups so as to minimize a cost while communicating in a distributed manner, are needed. We further contend that partitioning algorithms for these communicating-agent networks—whether distributed or centralized—must be flexible, in the sense that the algorithms should permit minimization of complex and varied cost measures. With these motivations in mind, we develop a flexible algorithm for network partitioning and self-partitioning using a stochastic automaton known as the influence model [268].

Distributed algorithms for self-partitioning may be valuable for various sensor networking and autonomous vehicle control applications. Consider the following:

- A group of autonomous vehicles in the field may need to self-assemble into multiple teams, in order to simultaneously complete multiple control tasks, e.g. search-and-destroy tasks (see e.g. [25, 228] for formation-control algorithms for autonomous vehicles). The vehicles should be grouped (partitioned) in such a manner that the self-assembly takes little time, and the robots in each group can easily communicate with each other.

- Sensors in an ad hoc network must choose one of several base stations for communication, so as to minimize the power required for multicasting as well as the latency of transmission from the sensors back to the base (see [273] for an overview of multicasting in ad hoc networks). Further, the sensors may need to classify themselves in such a manner that all the sensors associated with a particular base station can communicate among themselves, and further the network can tolerate any single failure in a communication link.

- Weakly-connected subnetworks within a computer network may need to be identified, so as to isolate a spreading computer virus.

In each of the these tasks, the nodes in a network must be partitioned so as to minimize a cost.

Further, for a variety of reasons (including security concerns, need for low-power and hence localized communication, and possibility for topological changes that are not known by a central authority), we may require a distributed algorithm for these partitioning tasks.

While there is a wide literature on graph partitioning (which derives primarily from parallel-processing applications, see [269] for an overview), partitioning tasks for the communicating-agent networks described above are novel in several respects:

1) As motivated above, the algorithms used often must be distributed, e.g. because of the high power cost of communicating with a central agent or the need for security. For the same reasons, sparsity of communication in use of the algorithm is also often a must. Further, algorithms that are scalable, i.e. ones in which the computational cost for each agent grows in a reasonable manner with the network size, are needed; distributed algorithms can permit scalability.

2) The cost to be minimized is often a complex or multivariate one (e.g., for sensor network applications, delay, power dissipation, and reliability may each play a role in the cost), and varies from one applciation to another. Thus, we require algorithms that are flexible with respect to the cost minimized. This contrasts with the bulk of the literature on partitioning [270–272], in which algorithms are designed for a particular cost, typically a min-cut cost or a min-cut cost with partition-size constraints*.

3) Communicating-agent networks are commonly subject to topological changes, for instance due to the addition of an agent or the failure of a particular communication link. Thus, partitions of the network may need to be adapted rapidly and frequently, ideally with minimal

* Bisection, in which the minimum cut that breaks the network into multiple equal-sized partitions is found, is of particular interest in the partitioning community [269].

communication.

These novel features have motivated us to develop a distributed and flexible algorithm for network partitioning/classification.

Specifically, we introduce an algorithm for network self-partitioning (i.e., distributed partitioning) that is based on a stochastic automaton known as the influence model. The influence model can be viewed as a network of discrete-time, finite-state Markov chains, which interact in the sense that the current status (state) of each site (chain) probabilistically influences the future statuses of its neighboring sites [268]. The basic premise for using the influence model (specifically the copying influence model) for partitioning graphs is that groups of sites in the model that are separated by weak influences tend to have different statuses, while sites interconnected by strong influences tend to form a cluster with a common status. Therefore, by associating influences with edge weights in a graph, allowing the influence model to run for some time, and then examining the statuses, we can identify a good partition quickly with respect to many typical cost functions. At the same time, the algorithm randomly searches through many potential partitions, and hence holds promise for minimizing multi-objective and complex cost functions. The technique is distributed in that each site only needs to communicate with graphical neighbors to determine its own partition.

This algorithm for network partitioning builds on our earlier work on a control-theoretic approach to distributed decision-making or agreement [68] (see also [230, 274] for other control-theoretic approaches to agreement and [275] for a study of sensor fusion that addresses/motivates distributed detection/decision-making). In the context of decision-making, we used the influence model to reach consensus among nodes in a manner that reflected their initial divergent opinions about a topic of interest; here, the influence model does not generate one opinion, but instead finds low cost cuts as boundaries between multiple opinions or statuses. Also of interest to us, stochastic automata have been used as tools for routing in sensor networks (e.g. [276]), and have been used as *gossip protocols* for information dissemination in ad hoc networks [277]. There is also a much broader literature on the analysis of stochastic automata, and their application to modeling and computational tasks. This literature is outside the scope of this chapter, see [278, 279] for general introductions.

While our primary motivation is self-partitioning, our studies suggest that the influence model-based algorithm is also valuable for centralized partitioning problems in which multiple complicated costs must be minimized, or in which costs are implicitly found through a simulation. For instance, motivated by fault-tolerance and fault-isolation applications (e.g. [280]), we have applied the algorithm to partition an electric power system so as to minimize both a line-weight and a power-imbalance cost in isolating two generators from each other. We shall briefly explore this centralized application in the chapter.

The remainder of this chapter is organized as follows. Section 20.2 poses the graph partitioning problem in a quite general way, in the process overviewing commonly-studied partitioning problems and standard algorithms for solving them. Section 20.3 briefly reviews the influence model, on which our partitioning algorithm is based. Section 20.4 describes the influence model-based distributed partitioning algorithm, in particular describing the mapping from the graph to the influence model, the distributed recursion used for partitioning, and centralized/distributed means for stopping the algorithm. In Section 20.5, we prove that the algorithm finds the optimal partition with certainty given certain weak graph-structural assumptions, and also discuss the performance of the algorithm. In Section 20.6, we pursue applications and give several illustrative examples, to better motivate our approach to partitioning and to further evaluate its performance.

20.2 Problem Statement

Since one of the features of our influence model-based partitioning algorithm is its flexibility, we begin by describing the partitioning (classification) problem in a quite general manner, but taking care to highlight sub-problems of particular interest. In the process, we give a brief review of the research on partitioning that is relevant to our development. We refer the reader to [269, 284] for thorough reviews of the partitioning literature.

Broadly, a k-way partitioning algorithm is concerned with classifying the vertices of a graph into k disjoint subsets. Specifically, let us consider a graph with (finite) vertex-set V that has cardinality n. We associate a positive *mass* m_v with each vertex (node) $v \in V$. In addition to the vertices, our graph also comprises a set of positively-weighted, directed edges. That is, for each ordered pair of distinct vertices v_i, v_j, we associate a weight $w_{ij} \geq 0$, where $w_{ij} = 0$ indicates that a directed edge is not present while $w_{ij} > 0$ indicates a weighted edge.

The partitioning problem that we consider is to classify the n vertices into k disjoint, non-empty subsets so as to minimize a cost function, while possibly enforcing one or more constraints. The cost function and constraints are phrased in terms of the total masses of the subsets and the edge weights on cuts.

Formally, we define a **k-way partition** of a graph as a subdivision of the nodes of the graph into k disjoint, non-empty **subsets** (components) $S_1, \ldots, S_k$. We are interested in identifying a partition that minimizes a **cost function**

$$f(M(S_1), ..., M(S_k), W(S_1, S_2), ..., W(S_k, S_{k-1})),$$

where $M(S_i) \overset{\triangle}{=} \sum\limits_{i \in S_i} m_i$ is the **mass** of subset i, and $W(S_l, S_m) = \sum\limits_{i \in S_l} \sum\limits_{j \in S_m} w_{ij}$ is the **size of the cut** between subsets i and j. We seek to minimize the cost function over the class of partitions that, in general, satisfy a number of constraints of the following types:

- **Algebraic constraints.** These are of the form

  $g(M(S_1),...,M(S_k),W(S_1,S_2),...,W(S_k,S_{k-1}))=0.$

- **Set inclusion constraints.** These have the form $v_i \in S_j$, i.e. particular vertices are constrained to lie in particular subsets. We often refer to a vertex that is constrained to lie in S_j as a **reference vertex** for subset j.

We use the notation $S_1^*,\dots,S_k^*$ for a partition that minimizes the cost subject to the constraints, and refer to this partition as an **optimal solution** of the partitioning problem[†]. Our aim is to solve the partitioning problem in a distributed manner, i.e. so that only communications along the edges of the graph are needed in finding the optimal partition.

A variety of partitioning problems considered in the literature are examples of the problem described above. It is worth our while to briefly discuss these problems and associated literature, focusing in particular on *Partitioning with Reference Nodes* (Item 4 below) because of its relevance to our applications. Commonly-considered partitioning problems include the following:

1) The *Min-cut Problem* is a k-way partitioning problem in which the subsets are chosen to minimize the total strength of the cuts between the components, with no algebraic or set inclusion constraints enforced. That is, the components are chosen to minimize the unconstrained cost function $f=\sum_{i=1}^{k}\sum_{\substack{j=1\\j\neq i}}^{k}W(S_i,S_j)$. The min-cut problem is well-known to admit a polynomial-time solution, and several search algorithms have been developed (see e.g. [285]). Spectral methods ([287–289]) and stochastic algorithms based on coalescing strongly-connected nodes [286] have also long been used to find min-cuts.

[†] We can in fact allow a far more general cost function, e.g. one that depends on dynamics defined on the graph. We adopt this form here for clarity in our explanation of why our algorithm is expected to work well.

2) The *Bisection Problem* is a 2-way partitioning problem, in which the subsets are chosen to minimize the strength of the cut between them, subject to the constraint that the masses of each subset are equal. That is, the cost function $f = W(S_1, S_2) + W(S_2, S_1)$ is minimized, subject to the algebraic constraint $g(M(S_1), M(S_2)) = M(S_2) - M(S_1) = 0$. Very often, the masses of the vertices are assumed to be all unity, so that the constraint reduces to enforcing that subsets have equal cardinality. The bisection problem finds its major application in parallel computing [269], where equally distributed workloads are desired. Bisection is a difficult (NP-hard) problem and has a wide literature. Specifically, classical bisection algorithms fall into four categories: (1) geometric partitioning algorithms based on coordinate information [290,291], (2) greedy search algorithms like the Kernighan-Lin algorithm [271], (3) spectral methods (methods based on eigenvalue/eigenvector structure of matrices associated with the network graph) [270, 292], and (4) stochastic algorithms including genetic algorithms [293, 294] and simulated annealing [272, 295].

3) In some applications, the exact mass constraint of bisection is not needed, yet it is useful to have subsets of roughly equal size or mass. For such applications, a *mass-weighted min-cut problem* is often solved. In particular, a cost function $f = \frac{\sum_i \sum_{j \neq i} W(S_i, S_j)}{M(S_1)...M(S_k)}$ is minimized, assuming no algebraic constraints. The form of this cost function has been studied in [297] and is deeply connected to the convergence rate of the linear dynamics defined on the graph. We note that term *ratio cut* has sometimes been used in the literature for these weighted problems.

4) Sometimes, an application dictates that one of the above problems (or another k-way partitioning problem with a different cost) must be solved, subject to set-inclusion constraints, i.e. subject to constraints that certain reference nodes are contained in each component. We refer to such problems as *k-way partitioning problems with reference nodes.* Partitioning with reference nodes is of interest to us for several reasons: 1) problems in several distributed applications—for instance, the problem of grouping sensor nodes with base stations for multicasting—have this form, 2) these problems are known to be NP-hard for $k \geq 3$ and hence still require development of good algorithms [283], and 3) our algorithm is naturally designed to address this problem and hence gives fast solutions to the problem.

There is a wide literature on algorithms for solving these partitioning problems (see, e.g., the review articles [269, 284, 290]). A thorough review of this literature is far beyond the scope of this chapter, but let us attempt to briefly summarize this work with the aim of delineating our approach from those in the literature. Most of the current partitioning algorithms are aimed at solving a particular problem (perhaps most commonly the bisection problem) in a centralized manner. For example, spectral methods have been used for min-cut and bisection problems, while SA, GA and K-L are designed specifically for bisection [271, 272, 293]. In constrast to these methods, our applications motivate us to seek an algorithm that can find the optimal partition for a range of cost functions, even if perhaps at slightly higher computational complexity.

The algorithms in the literature that are stochastic (e.g., GA and SA) are of interest to us [269], since our algorithm is also stochastic. Very broadly, our algorithm is similar to these in that it searches randomly through plausible partitions, using the uncertain generation to seek more optimal partitions. However, our algorithm is significantly different from those in the literature, in that our algorithm does not react to the cost of the current partition: instead, its update is based solely on the graph topology. This topological approach has the advantage of permitting flexibility in the optimized cost, and (as we shall show) of allowing identification of the minimum-cost solution with probability 1.

Perhaps most significantly, we contribute to this broad literature by developing an algorithm for distributed or self-partitioning, i.e. an algorithm in which agents associated with graph vertices can decide their optimal partitions based solely on communication with neighboring agents. To the best of our knowledge, there have been no other algorithms developed that achieve partitioning without any global perspective at all in the graph.

20.3 The Copying Influence Model: A Brief Review

Our algorithm for partitioning is based on evolving a stochastic automaton model. Specifically, we map a graph to a dynamic *stochastic network model*—a model in which values or statuses associated with network nodes are updated based on interactions with neighboring nodes. The statuses associated with the nodes form patterns as they evolve with time; these patterns turn out to identify good partitions of the graph. Since the automaton is updated only through interactions of nodes with graphical neighbors, it permits partitioning in a decentralized manner. The automaton that we use for partitioning is an instance of the *influence model* [268], a stochastic network automaton with a special quasi-linear structure. In this section, we very briefly review the influence model. We refer the reader to [268] for a much more detailed development.

An influence model is a network of n nodes or vertices or *site*s, each of which takes one of a finite number of possible *statuses* at each discrete time-step. We use the notation $s_i[k]$ for the status of site i at time k. We refer to a snapshot of all the sites' statuses at time k as the *state* of the model at time k. The model is updated at each time-step according to the following two stages:

1) Each site i picks a site j as its *determining site* with probability d_{ij}.

2) Site i's next-status is then determined probabilistically based on the current status of the determining site j. That is, the next status is generated according to a probability vector, which is parameterized by the current status of the determining site.

We shall only be concerned with a special case of the influence model called the *copying influence model*, in which each site takes on the same number k of statuses (labeled $1, \ldots, k$ w.l.o.g.), and furthermore each site simply *copies* the status of its determining site at each time step. To reiterate, at each time-step in the copying influence model, each site i picks a neighbor j with probability d_{ij} and copies the current status of that neighbor.

The influence model and copying influence model are compelling as modeling and algorithmic tools because they have a special quasi-linear structure. In general, for stochastic network models such as the influence model, we note that the statuses of all sites together are updated in a Markovian fashion, and hence the joint status of all sites are governed by a very large "master" Markov chain with k^n states. However, for the influence model, status probabilities of individual sites and small groups of sites can in fact be found using low-order recursions. For instance, the probability of site i taking status m at time $k+1$ in the copying influence model can be tracked using the following low-order recursion:

$$P(s_i[k+1] = m) = \sum_j P(s_j[k] = m) d_{ij} \tag{20.1}$$

Furthermore, the special structure of the influence model permits us to identify qualitative features of the master Markov chain based on the low-order recursions. These special tractabilities of the influence model make it possible to characterize the performance of algorithms built using the model, such as the algorithm developed here.

20.4 Algorithm Description

We can use the copying influence model as a tool for solving the partitioning problem described in Section 2 under rather broad conditions. Furthermore, since the influence model update only requires interaction among graphical neighbors (in a sense that will be made precise shortly), the algorithm is essentially decentralized (though a bit further effort is needed to *stop* the algorithm in a decentralized manner). The combination of flexibility and decentralization makes the influence model-based algorithm applicable to a range of partitioning tasks, including those discussed in the introduction. In this section, we describe the influence model-based partitioning algorithm. In the next section, we prove that the algorithm works (finds the optimal solution with certainty) under broad conditions. Here, we first outline the algorithm, and then fill in the details.

1) **Mapping** We map the graph to a copying influence model, by associating large influences with strong interconnections in the graph, and weak influences with weak interconnections. We note that we can permit asymmetric interconnection strengths.

2) **Initialization and Recursion** We choose the initial state for the copying influence model. Here, the status of each site identifies the subset of the corresponding node in the graph. The statuses of the sites are updated recursively according to the developed copying influence model, and hence a sequence of possible partitions of the graph are generated. We note that this is a distributed computation, in that each site updates its status using only local information (i.e. information from graphical neighbors). Thus, in cases where a group of nodes in a real distributed system must self-partition, the influence model recursion can be implemented using localized communications between agents in the network. In presenting and analyzing the recursion, we find it convenient to first consider the case of partitioning with reference nodes[‡], and then address partitioning problems without reference nodes.

3) **Stopping** The recursion is terminated based on cost evaluations for a centralized algorithm

[‡] For notational convenience, we focus on the case where there is one reference node per component, but our development can straightforwardly be generalized to cases where the number of partitions is different from the number of references.

and by decreasing influence model probabilities in the decentralized case. The statuses of the influence model at the stopping time specify the chosen partition.

Mapping

We map the graph to a copying influence model with k possible statuses, with the motivation that we can identify a sequence of partitions of the graph by updating the influence model. That is, our algorithm classifies (partitions) the vertices in the graph according to the statuses of the corresponding influence model sites at each time-step of the recursion. The first step toward building this partitioning algorithm is to map the graph to a copying influence model, in such a manner that the copying probabilities in the influence model reflect the branch weights. In particular, we associate an influence model site with each vertex in the graph. We then choose the copying probabilities (influences) as

$$d_{ij} = \begin{cases} \frac{\Delta w_{ji}}{m_i}, & i \neq j; \\ \\ 1 - \Delta \sum\limits_{l} \frac{w_{li}}{m_i}, & i = j, \end{cases} \tag{20.2}$$

where Δ is chosen such that $\Delta \leq \frac{1}{\max\limits_i \sum\limits_j \frac{w_{ji}}{m_i}}$. Thus, large weights are associated with large influences, and small weights are associated with small influences; moreover, a large mass (inertia) m_i incurs small influence from other sites on site i (and large influence from itself), and a small mass m_i incurs large influence from other sites on site i.

In addition to above direct interpretation, we can also give a linear systems-based interpretation for the mapping. In particular, we can show that the status-probability recursion (Equation 20.1) of the developed influence model is a discretized version of a certain linear differential equation defined on the graph. This linear system viewpoint is valuable because it indicates the close connection of our algorithm with some typical network dynamics, and because it can potentially permit analytical connection of our algorithm with spectral partitioning algorithms. From the linear system viewpoint, the parameter Δ can be interpreted as the discretization step. More generally, Δ should be chosen large enough to achieve a fast convergence rate. We have specified the upper bound to guarantee that all the influence model parameters are valid.

In many decentralized and centralized applications, we envision this mapping stage as being done *a priori* by a centralized authority, even when the partitioning itself must be done in a decentralized manner. For instance, when new sensors are added to an existing network, the network designer can perhaps pre-program information about the communication topology and strengths of interactions between the sensors. However, it is worth noting that the mapping to the influence model is in fact inherently decentralized (i.e., an agent associated with vertex i in the graph can compute the weights d_{ij} from the vertex's mass and the weights of edges to neighbors) except in one sense: the scaling parameter Δ is a global one. Noticing that the maximum allowed value for Δ depends on the total weights of edges out of nodes and node masses, we note that Δ can often be selected *a priori* based on some generic knowledge of the graph topology (for instance, knowledge of the maximum connectivity of any single node), when decentralized mapping is also required.

Initialization and Recursion

Let us first develop an algorithm for k-way partitioning with reference nodes (specifically, with one reference node per partition). For the problem of k-way partitioning with reference nodes, we fix the k *reference sites* (the sites in the influence model corresponding to the reference nodes) with distinct statuses from 0 to $k-1$, and choose the initial statuses of other sites arbitrarily. Here, in order to fix the reference sites' statuses, we need to make a slight modification to the influence model developed in Equation 20.2 such that reference site i always chooses itself as the determining site:

$$d_{ij} = \begin{cases} 0, & i \neq j; \\ 1, & i = j, \end{cases} \tag{20.3}$$

(In a distributed context, notice that we only require that the reference nodes know their own identities to implement this initialization).

To generate a good partition, we then update the copying influence model. The state at each time-step of the recursion identifies a partition of the graph: that is, we classify the nodes whose associated sites are in status i in subset S_i. We note that the partition identified at each time-step automatically satisfies the set inclusion constraints for k-way partitioning with reference nodes. We shall show that this recursion, which generates a random sequence of partitions, eventually finds (passes through) the optimal solution with probability 1 under broad assumptions, after sufficient time has passed. We note that the recursion is completely distributed, in the sense that each node can decide its own subset at each time-step solely from its graphical neighbors.

In practice, we must develop a methodology for stopping the algorithm. Below, we discuss distributed and centralized approaches for stopping. The stopping methodologies seek to select low-cost partitions, while checking possible algebraic constraints. We shall show that appropriate stopping criteria permit identification of the optimal solution under broad assumptions, with probability 1.

Conceptually, one might expect this partitioning algorithm to rapidly identify low-cost partitions, because strongly-connected sites in the influence model (sites that strongly influence each other) tend to adopt the same status through the influence model recursion[§], while weakly-connected sites do not influence each other and hence maintain different statuses. Recalling that the influence

[§] We refer the reader to our earlier work on *agreement* for further discussion about the dynamics of strongly-influencing sites [68].

strengths reflect edge weights and node masses, we thus see that the partitions identified by the model typically have strongly-connected subsets with weak cuts between them. For many typical cost functions, the optimal partition comprises strongly-connected subsets with weak links, and hence we might expect the algorithm to find good cuts quickly.

For k-way partitioning (without reference nodes), we can find the optimum by solving the partitioning problem with reference nodes for all sets of distinct reference node selections, and optimizing over these. (Notice that we can actually keep one reference fixed, and search through possible placements of the other references.) This search is impractical when a large number of partitions is desired; we shall briefly consider alternatives in discussing future work. Most applications of interest to us have natural reference vertices, so we do not focus on the case without references.

A few further notes about the recursion are worthwhile:

- For simplicity of presentation, we have considered a discrete-time update, and hence a distributed implementation of the recursion in a network nominally requires a common clock for the agents in the network. However, we can equivalently use an update in which each site updates its status at random times (specifically, according to a Poisson arrival process); the recursion in this case is amenable to the same analyses as the recursion described here, and hence can be shown to achieve optimal partitioning.

- Regarding scalability in a distributed setting, we note that each agent in a network only needs to randomly select a neighbor and poll that neighbor at each time-step to implement the recursion, so the processing/communication per time step does not increase with the size of the network. The total processing/communication cost thus scales with the duration of the recursion. In the next section, we give an argument that the scaling of the algorithm's duration with the size of the network is good compared to other partitioning algorithms in many cases.

- In some applications, we may already have one partition of a graph, and may wish to improve on this partition (with respect to a cost of interest) or to adapt the partition to changes in the graph. In such cases, we can speed up the recursion by initializing the influence model according to the original partition.

Stopping

Again, consider the k-way partitioning problem with reference nodes. (The adaptation to the general k-way problem is trivial.) For centralized problems, the global partition is known to a central agency at each recursion stage (time-step) and hence the cost can be evaluated and constraints can be checked. The minimum cost partition found by the algorithm can be stored. In this case, we propose to stop the updating after a waiting time, i.e. when the minimum-cost partition has not changed for a certain number of algorithm stages. This waiting time depends on the network structure and should be pre-calculated before the updating process. Generally speaking, the larger the size of the network, and the smaller the influences, the bigger the waiting time should be. We will show that a sufficiently long waiting time guarantees that the optimal solution is identified.

For distributed problems, it is unrealistic that a single agency can evaluate the global cost of a partition as in the centralized case, since each node only has available local information. A simple strategy in the distributed case is the blind one: the algorithm can be stopped after a finite number of time-steps, where this number is based on the convergence properties of influence models. A more complex strategy is to distributedly compute the cost at each stage using an agreement protocol (see, e.g., [230]).

Another clever strategy for distributed stopping is to use an influence model with state-dependent parameters. In particular, we progressively isolate (reduce the influence) between sites with different statuses after each update (and increase the self-influence correspondingly), until the influence model is disconnected (partitioned). More specifically, for each update, the (time-varying) influence $d_{ij}[k]$ is modified as follows:

- If $s_i[k] \neq s_j[k]$ and $d_{ij}[k] \geq \delta$, then $d_{ij}[k+1] = d_{ij}[k] - \delta$ $(i \neq j)$ and $d_{ii}[k+1] = d_{ii}[k] + \delta$;
- If $s_i[k] \neq s_j[k]$ and $d_{ij}[k] < \delta$, then $d_{ii}[k+1] = d_{ii}[k] + d_{ij}[k]$ and $d_{ij}[k+1] = 0$ $(i \neq j)$;
- If $s_i[k] = s_j[k]$, then $d_{ij}[k+1]$ remains the same.

When this time-varying algorithm is used, we note that the statuses of sites converge asymptotically (see Figure 20.1). This is because the influence model becomes disconnected, so that each partitioned component has only one injecting site and is guaranteed to reach consensus. Thus, a partition is found asymptotically. Furthermore, it is reasonable that this algorithm finds a good partition, since weak edges in the original graph tend to have different statuses at their ends in the influence model, and hence these edges are removed by the algorithm. We refer to this strategy as **partitioning with adaptive stopping**.

20.5 Algorithm Analysis

In this section, we prove that the influence model-based partitioning algorithm finds the optimal solution when either centralized or decentralized stopping is used. Specifically, we show that the influence model algorithm identifies the optimal solution with probability 1, given that the optimal solution satisfies certain broad connectivity conditions (which, as we show, is automatic for several

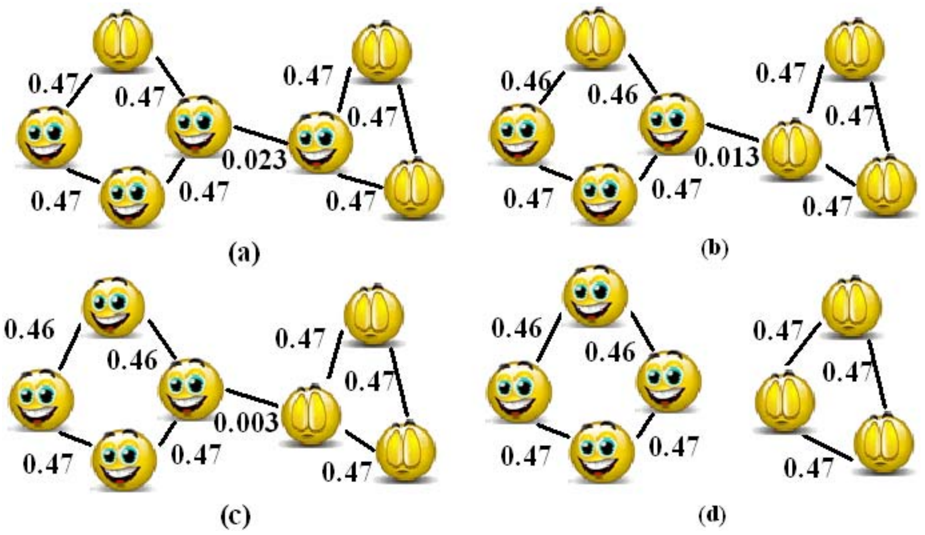


Fig. 20.1: This diagram illustrates how a network partitions itself (based on the update of the time-varying copying influence model) in a totally distributed manner.

common partitioning problems). The (quite-weak) connectivity conditions required of the optimal solution are based on the requirement that the influence model must be able to distribute a single status to all sites corresponding to a particular subset, from a particular *source* site (which in the case of partitioning with reference nodes is the reference).

Before presenting results on the algorithm's ability to find optimal partitions, let us begin by formally defining source vertices, so that we can formalize the connectivity conditions required of the optimal:

Definition 1 *Consider a particular partition of a graph, and a vertex v within a particular subset. For this partition, the vertex v is a* **source vertex**, *if we can find a path from v to each other vertex in the subset that remains within the subset (i.e., never enters a vertex in another subset).*

We are now ready to present the main results on the algorithm's ability to obtain the optimal solution. We assume throughout this development that the partitioning problem of interest to us has at least one feasible solution. We first give conditions under which the algorithm can reach the optimal solution for a partitioning problem with reference nodes.

Theorem 20.1. *Consider the general k-way partitioning problem with reference nodes, as described in Section 20.2. An optimal solution is identified by the influence model algorithm with probability* 1 *(i.e., the algorithm passes through an optimal solution), if there is an optimal solution such that each reference vertex is a source vertex.*

Proof. In order to show that the optimal solution is identified with probability 1, let us consider the master Markov chain for the influence model. We only need to show that the optimal state (the influence model state associated with the optimal solution) can be reached with positive probability from any other state (i.e., there is a sequence of influence model updates that leads from an arbitrary state to the optimal state) [72]. The optimal state has the property that all the sites in each partition have the same status, while sites in different partitions have different statuses.

In showing that the optimal state can be reached, let us limit ourselves to updates in which sites determine their statuses from other sites in the same partition in the optimal solution—only such updates are needed. Now consider a single subset in the optimal solution. Let us call the reference vertex in the subset v_s. Since v_s is a source vertex, there is a path from v_s to every other vertex in the subset that remains in the subset. Let us suppose that the longest path from v_s to another vertex in the subset is m. Then we note that there is a positive probability that all influence model sites corresponding to that subset take a status of the reference site (the site corresponding to the reference vertex) after m time-steps. This can be proved simply by recursion: assume there is a positive probability that all sites within a distance of i from the reference take on the initial status of the reference site at each time step i; since there is a positive probability that each site within a distance of $i+1$ is influenced by a site within a distance of i from the reference, there is also a positive probability that all sites within a distance of $i+1$ from the reference site take on the reference status at time $i+1$. Using this argument, we also find that there is non-zero probability that all sites take on the reference status, at any time $k \geq m$. Thus considering the influence model as a whole, we see that there is a positive probability that all partitions are found after a finite number of time-steps, and so the theorem is proved. $\square$

We note that this proof is closely related with the proof characterizing the asymptotics of a *binary influence model* in [268].

We have thus shown that the algorithm can solve the partitioning problem for a wide variety of cost functions, specifically ones in which the optimal solution has the described connectivity condition. We stress that the connectivity condition—namely, the existence of paths from each reference vertex to the other vertices in its subset—is quite weak: connectedness of the subsets in the (directed) graph is sufficient but not necessary for the connectivity condition to hold. In fact, for a range of distributed applications (for instance, for multicasting in mobile networks or tracking using autonomous-vehicle teams), such connectivity may automatically be required or desired since we need agents/nodes in each identified subset to subsequently communicate among themselves.

Let us next formalize that the algorithm can be used to find optimal solutions for the k-way partitioning problem without reference nodes.

Theorem 20.2. *Consider the general k-way partitioning problem. An optimal solution is identified* *by the influence model algorithm with probability* 1*, if each subset of some optimal solution has a source vertex.*

Proof. Since we solve partitioning problems without reference nodes by searching through distinct reference node placements, this result follows directly from Theorem 20.1.

We have noted that Theorems 20.1 and 20.2 require the optimal solution to have a particular weakly-connected structure to guarantee its identification. Of course, the optimal partition is not known *a priori*, so it is helpful to identify classes of partitioning problems for which this connectivity condition is necessarily true. The following corollary identifies two such classes.

Corollary 20.3. *Consider the min-cut problem and mass-weighted min-cut problem with/without reference nodes. An optimal solution is identified by the influence model with probability* 1*, if the graph has the following structure: all the edges are bi-directional.*

Proof. It is easy to check that optimal partitions for these problems constitute connected subgraphs. Thus, together with the bi-directionality assumption, we see that Theorems 20.1 and 20.2 can be applied.

Theorems 20.1, 20.2 and Corollary 20.3 show that an optimal solution is identified with probability 1 (i.e. the influence model passes through an optimal solution), given that this solutions satisfies the appropriate connectivity conditions. However, we have not yet shown that the algorithm will stop at the optimal solution with certainty. The following two theorems show that our partitioning scheme is successful when the centralized and distributed stopping criteria are used, respectively.

Theorem 20.4. *Consider the general k-way partitioning problem with (without) reference vertices, and assume that each reference vertex is a source vertex (respectively, each subset has a source vertex) in the optimal solution. Then the probability that the influence model algorithm with centralized stopping chooses the optimal solution approaches* 1*, in the limit of long waiting times.*

Proof. This result follows directly from the standard analysis of Markov chains (see e.g. [72]). Specifically, as the waiting time is increased, the probability that a better solution, if one exists, is not found while waiting can be seen to decrease to 0.

Partitioning with distributed stopping (in particular, partitioning with adaptive stopping) is quite a bit more complicated to analyze than the centralized algorithms, because the parameters of the influence model are changing in reaction to the site statuses. Here, we formalize that the partitioning-with-adaptive-stopping algorithm is able to solve the min-cut k-way partitioning problem with reference nodes, in the case where the edges between subsets are weak (of order ϵ in weight) compared to edges in the partition. Although our formal result is in such a limiting case, the proof in fact makes clear that the minimum cut is found whenever the influence model associated with the original graph is more likely to have status differences over the minimum cut than over any other cut. The influence model has this property for a large (albeit somewhat hard to delineate) class of graphs, not only ones with weak minimum cuts; this is sensible, since after all the influence model update is structured to find minimum cuts (not only order-ϵ cuts) more commonly than other cuts. Our examples bear out that the distributed-stopping algorithm is practical for typical distributed applications.

Here is the formal result, with proof:

Theorem 20.5. *Consider the min-cut partitioning problem with reference nodes. Assume that the graph has bi-directional edges, and further that the optimal cut is small (of order ϵ) compared to any other cut. Then the probability that the influence model algorithm with distributed stopping chooses the optimal cut approaches* 1*, in the limit of small δ.*

Proof. First notice that if we can show that all the weights of edges in the optimal cutset (the cutset associated with the optimal solution) go to 0 before any other one does in the average sense, we are done since as δ approaches 0, the probability for a particular run to be deviated from the average run approaches 0. Let $\epsilon_1[0]$, $\epsilon_2[0]$,...$\epsilon_n[0]$ denote the weights of edges in the optimal cutset of an influence network I at time-step 0. Without loss of generality, we arbitrarily pick an edge with weight $\lambda[0]$ other than the edges in the optimal cutset and show that all the n $\epsilon_i[k]$'s approach 0 before $\lambda[k]$ approaches $\lambda' = \lambda[0] - \sum_{i=1}^{n} \epsilon_i[0]$ in the average sense at some time-step k. With the assumption that the edges in the optimal cutset are sufficiently weak, we have $\lambda' > 0$, then we are done.

To do so, we construct a new influence network I', whose only difference with I resides in that the weight $\lambda[k]$ is replaced by λ', and each $\epsilon_i[k]$ is replaced by $\epsilon_i[0]$. The reason to come up with I' is that the original I is a very complex network with varying weights. By proving for I' whose weights never change, that the conclusion holds first, and reducing the problem for I to the one for I', we can simplify the proof.

Considering I' with fixed weights, it is easy to check that in the average sense, $\epsilon_i[k]$ reaches 0, before $\lambda[k]$ reaches $\lambda[0] - \epsilon_i[0]$; consequently, both $\epsilon_i[k]$ and $\epsilon_j[k]$ reach 0 before $\lambda[k]$ reaches $\lambda[0] - \epsilon_i[0] - \epsilon_j[0]$; and finally, all $\epsilon_i[k]$ reach 0 before $\lambda[k]$ reaches λ', where $\lambda[k]$ and $\epsilon_i[k]$ are weakened with time. This is because with the existence of sufficiently small optimal cut, the probability for each site in an optimal partition to take the reference site's status is very high, and thus the joint probability for a pair of sites in an optimal partition to take different statuses are very small. In contrast, the probability for a pair of sites across the optimal cut to have different statues are very high. Therefore, the edges in the optimal cutset are weakened faster than weight $\lambda[k]$ does in the average sense.

Now that we know for I' with fixed weights, all $\epsilon_i[k]$ reach 0 before $\lambda[k]$ reaches λ' in average, we need to show that it implies for I with varying weights, the same conclusion also holds. With the assumption that $\lambda[k]$ is greater than λ', $\lambda[k]$ in I approaches λ' slower than $\lambda[k]$ in I' does, and every $\epsilon_i[k]$ in I approaches 0 no slower than than $\epsilon_i[k]$ in I' does, since $\epsilon_i[k]$ may be weakened in I. The above assumption is true since before $\lambda[k]$ decreases to λ', all the edges in the optimal cutset are already broken. Hence we prove that all the edges in the optimal cutset approach 0 before other edge does in the average sense. The proof is complete.

We have thus shown that our algorithm can find the optimal partition in both a centralized and a distributed manner, under broad conditions. Next, it is natural to characterize or test the performance of the algorithm: of course, any algorithm that searches through all possible partitions can find the optimal one, so an algorithm such as ours is useful only if it can find the optimal solution quickly compared to a combinatorial search. Although we leave a full analytical treatment of the algorithm's performance for future work, we give here a conceptual discussion of why the algorithm is fast, and also evaluate the performance of the algorithm in examples in the next section.

The *No Free Lunch* theorems [299] provide an interesting conceptual framework for the performance evaluation of our algorithm. These negative results state that, over the class of all possible cost functions, there are no algorithms that always perform well; in fact, all algorithms are equally costly (i.e., take equally long) on average. Thus, an algorithm must be tailored for the particular cost function of interest. From this perspective, our algorithm works well because typical optimal costs correspond to weak cuts in the graph and strongly-connected partitions, and hence good algorithms should search through these weak-cut partitions first. Our algorithm is tailored to quickly find these weak-cut solutions (since the strongly-connected sites in the copying influence model tend to adopt the same status while weakly connected ones differ), while also searching through other solutions less frequently.

We have recently obtained an analytical justification for the performance of the algorithm. Specifically, we can show that, on average, the algorithm solves the k-way min-cut problem with reference nodes in polynomial time, given that the minimum cut is sufficiently weak compared to other cuts in the graph. Since the k-way partitioning problem with reference nodes is NP-hard, a polynomial-time algorithm for a class of graphs is a worthwhile result, and gives some indication of the performance of the algorithm. This performance analysis also has the benefit of explicitly connecting the performance with spectral properties of the linear recursion for influence model site statuses, and hence potentially permitting comparison of the algorithm with spectral partitioning methods (e.g., [287–289]). This analysis of performance unfortunately requires rather extensive review of the influence-model's analysis, so we omit the details of the result from this expository chapter.

20.6 Applications and Examples

In this section, we briefly introduce several potential applications of our algorithm, and also present canonical examples that illustrate or enrich aspects of our analytical development. The applications and examples together are meant to further motivate the described algorithm.

Application 1: Classification for Multicasting in Ad Hoc Networks Distributed partitioning holds promise as a tool for classification in distributed sensor networks and mobile ad hoc networks, e.g. for the purpose of multicasting or of transmitting information from the sensors/mobiles back to "leader nodes" or base stations or central authorities.

There is a wide literature on *routing* in ad hoc networks when the absolute positions of the sensors/mobiles are known (see [300] for a survey of methods). Recently, distributed algorithms (specifically local-averaging methods) have been used to infer location information in the case where absolute positions are unknown except at peripheral locations (see e.g. [301]), and hence permit development of routing algorithms for these networks. Beyond routing, classification of sensors/mobiles with base stations is an important task, for the purpose of multicasting (transmitting information to many destinations from multiple sources) or so that subsequently data can be routed to and from appropriate base stations to the sensors/mobiles.

Several recent articles have addressed multi-hop multicasting in ad hoc networks (see e.g. [273]). In multicasting applications as well as other settings where data may be transmitted to/from several sources or base stations, classification of mobiles/sensors with the base stations is important. We contend that the influence model-based partitioning tool can advance the state-of-the-art on classification in ad hoc networks, for several reasons:

- As made clear by the comparison of location-known and location-unknown algorithms for routing, decentralized algorithms for classification may be needed in cases where there is no central authority with full knowledge of the network. Even if classification is done in a centralized manner, only partial information may be known about the network. For instance, distances between sensors/mobiles or at least the connection topology may be known, but the exact locations of each sensor/mobile may not. Conversely, exact locations may be known, but the connection topology may be unknown. The mapping from the graph to the influence model, and the influence model update itself, are based on local information and hence our partitioning method is suited for this setting.

- We may need to optimize the classification with respect to several (possibly complex) cost criteria (including for example minimum (or average) hops to each base station, average delay cost, and various reliability criteria). In fact, the costs may depend on the specifics of the decentralized algorithm used for routing/multicasting. The influence model-based algorithm permits us to consider multiple and complex cost criteria.

- Often, the topologies of sensor networks and mobile ad hoc networks change with time, and hence it is beneficial to use an algorithm that can update the optimum with little effort (in either a distributed or centralized case). The influence model-based algorithm has this advantage.

For illustration of this application, we have used the influence model-based algorithm for sensor classification in a small example (one with 3 base stations and 27 sensor nodes). The example was generated by placing the 30 sensors in a uniform i.i.d. manner within the unit square, allowing communication between sensors within 0.3 units of each other, and choosing three sensors (Sensors 3, 14, and 20) to also serve as base stations. We associate a weighted undirected graph with the sensor network in which the 30 vertices correspond to the 30 sensors, and branches indicate communication between sensors. Each branch weight is chosen to be inversely proportional to the distance between the pair of sensors, with the motivation that longer communication links are more apt to failure and delay and hence are more weakly connected. We consider 3-way partitioning of this graph with reference vertices 3, 14, and 20 using the influence model algorithm.

We consider partitioning with centralized stopping, with respect to two cost functions:

- First, we partition the graph so as to *maximize* the minimum of the positive eigenvalues of the *Laplacian matrices* associated with the three subsets (partitions)[¶]. The minimum non-zero eigenvalue of the Laplacian associated with each subset is well-known to indicate the connectivity of that subset, and can be used to bound several relevant graph-theoretic properties such as the graph diameter (see [73] for a full development). By maximizing the minimum among the non-zero eigenvalues, we thus find a partition with strongly-connected subsets and weak links between them. The optimal partition with respect to this *minimum-subgraph-eigenvalue* cost measure is shown in Figure 20.2.

- Our second cost measure is motivated by consideration of low-cost and low-overhead distributed routing for ad hoc and sensor networks. A simple greedy algorithm for routing when location information is available is to send the message to the node (sensor) closest to the destination during each transmission (see e.g. [300]). Assuming such a greedy routing algorithm is used, our aim is to classify the sensors with base stations so that the maximum number of hops to a sensor from its base station is minimized. (The average number of hops could be used instead.) Thus, we partition the graph using this maximum number of

[¶] We notice a maximization problem can routinely be converted to a minimization by choosing a cost that is the negative of the original cost.

hops when greedy routing is used. The optimal partition when this *greedy-routing cost measure* is shown in Figure 20.2. We note that, as expected, the optimal partition has subsets which are more balanced in size but contain weaker links, as compared to optimum for the minimum-subgraph-eigenvalue measure. This example highlights an interesting advantage of the influence model: the greedy-routing cost function does not admit an analytical form but can be computed for a given partition, but nevertheless the optimal partition can be found.

We have also considered min-cut partitioning with distributed (adaptive) stopping for this example. The result is shown in Figure 20.2. We note that such a distributed algorithm could be implemented in the sensor network itself, and would only require individual sensors to have local parameters (in particular, distances to neighbors). Such a distributed algorithm might be especially useful in cases where the topology is subject to change, so that the sensors must re-classify themselves periodically.

As further illustration, we also show a 4-way partition with reference nodes of a 100-sensor network in Figure 20.2.

Application 2: Flexible Partitioning for Electric Power Systems We believe that our algorithm potentially has significant application in power system analysis, because of its flexibility. One potential application is for *islanding*, i.e. isolation of a portion of the power network to prevent disturbance propagation. We note that a good algorithm for islanding may need to take into account several requirements, including small generation-load imbalance within the isolated component, feasibility of making the desired cut, and disturbance-propagation dynamics. Our algorithm is a natural tool for such partitioning problems, in that we can minimize and keep track of multiple costs while identifying plausible islands (partitions). We have applied the influence model algorithm to

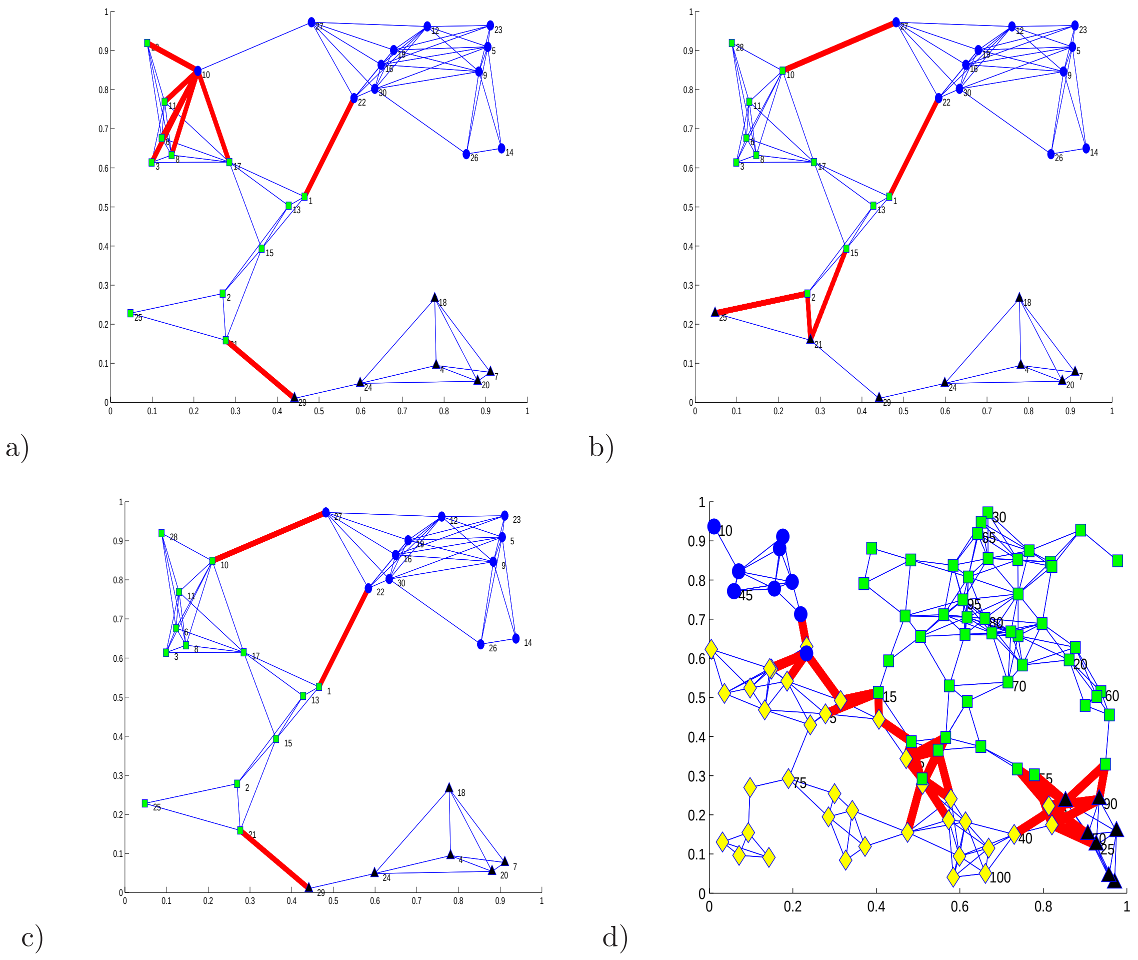


Fig. 20.2: Partitioning a 30-sensor network with reference nodes a) based on a minimum-subgraph-eigenvalue cost, b) based on a greedy-routing cost, and c) with distributed stopping. We also partition a 100-sensor network based on a minimum-subgraph-eigenvalue cost (d).

identify plausible islands in a very small (14-bus) example (see Figure 20.3). Specifically, we have used the influence model algorithm to track and minimize two cost metrics—the cut susceptance and the total absolute generator-load imbalance over the partitions. Our optimization shows the use of a partitioning algorithm that can track multiple costs: two different optimal solutions are identified quickly based on the two cost metrics, and both costs are computed for a family of plausible partitions (Figure 20.3). Interestingly, the optimal partition with respect to the minimum susceptance cost is typically found more quickly than the optimal partition with respect to the generator-load imbalance cost. The better performance in finding the minimum susceptance cost is not surprising, in that the susceptances are directly mapped to influence probabilities.

Because our algorithm can track multiple costs, we believe that it can potentially enhance the recent approach to islanding suggested in [280], which is slow-coherency (equivalently, spectrally) based. More generally, we believe that the flexibility offered by our partitioning algorithm may be valuable for various model-reduction tasks in power system analysis, as well as other network-centric tasks such as the identification of line trips of minimal cardinality that make infeasible the power flow solution.

Application 3: Self-Classification for Autonomous Vehicles Our distributed partitioning algorithm provides a means for networked autonomous vehicles to classify themselves. For instance, say that a network of communicating autonomous vehicles seeks to form two teams, each of which must congregate at a convenient location in space. Our self-partitioning algorithm can be used to identify groups of autonomous vehicles that are close to each other, and so to form the teams, in a completely distributed manner. We note that using partitioning in this context permits task dynamics in autonomous-vehicle applications, for which the role played by each agent actually depends on its initial position (state). In the interest of space, we do not pursue this application in detail here.

Canonical Example 1: Performance Simulations We explore the performance of our algorithm by pursuing min-cut partitioning with reference nodes on a seven-node circuit. Figure 20.4 illustrates the mapping between the circuit's conductance graph and an influence model. It is worth noting that the mapping has a meaningful dynamic interpretation in this case: the expected dynamics of the influence model is a discretization of the dynamics of a circuit with the specified conductances and unit capacitors from each node to electrical ground.

Our algorithm is guaranteed to find the optimal cut. Table 20.1 shows the performance of our algorithm, in terms of the average number of time-steps needed to reach this cut, for several values of the cut strength and discretization time-step. We note that the expected number of time-steps needed to reach the optimal cut is dependent on the strength of the optimal cut: the weaker the cut, the faster the algorithm.

We have compared our algorithm with spectral methods for this circuit example. When the weak cut is between nodes 2 and 3 and nodes 4 and 5, both the spectral method and our algorithm find the min-cut partition for all $0 \leq \epsilon < 1$. We notice that our algorithm requires 5 random number generations and 5 copying actions (communications in a decentralized setting) per time-step, and requires between 5 and 10 time-steps for a good choice of Δ and $0 \leq \epsilon \leq 1$. In comparison, a simple implementation of the spectral method requires on the order of 7^3 additions and multiplications. We notice that the expected computational cost of our algorithm depends on the strength of the cut, in contrast to the spectral algorithm. Interestingly, if the size-ϵ cuts are placed between node 1 and nodes 2 and 3 instead, the spectral method only finds the weak cut for $\epsilon \leq 0.26$, while our algorithm obtains the optimal solution for all ϵ (albeit at higher computational cost).

Canonical Example 2: Convergence of Distributed Stopping Recall that distributed stopping of our algorithm is achieved through reduction of influences. In this case, the algorithm's ability to obtain the minimum cut is predicated on choosing a sufficiently small influence reduction size. In this example (see Figure 20.5), we have characterized the percent of time that the correct partition is found, for several cut strengths and influence reduction sizes. In each case, we have also determined

	ε			
Δ	0.1	0.2	0.5	1
0.05	26.2	27.6	31.8	37.8
0.1	13.2	13.8	16.3	19.6
0.2	6.9	7.0	8.6	10.8
0.25	5.5	5.7	7.1	9.6
0.3	4.7	4.9	NA	NA

Tab. 20.1: The average steps to first reach the partition state with respect to Δ and ε based on 1000 sample runs

the expected number of iterations until the algorithm stops (see Table 20.2 for the results). The simulation results indeed show that the optimal cut is found with certainty, when the influence reduction size is chosen sufficiently small. We note that the time required to find the optimal partition increases as the influence reduction size is decreased.

	ϵ	δ	Δ	Steps	Percent
1	0.1	0.02	0.47	6.2	99.6
2	0.1	0.01	0.47	8.3	100
3	0.5	0.01	0.39	30.759	99.7
4	0.5	0.005	0.39	56.672	100
5	1	0.001	0.33	122.49	86.8

Tab. 20.2: Simulation result for Example 2 based on 1000 sample runs: *Steps* represents the average steps the algorithm takes to distributedly stop, and *Percent* represents the percentage of correct partitions the algorithm finds.

20.7 Future Work

Several directions of future work are worth discussing:

- **Complexity Analysis and Comparison.** A careful analysis of the complexity of our algorithm is required. Roughly, the computation required at each time-step scales with the number of edges in the graph. However, we have not determined the scaling of the number of time-steps required to find the optimal solution with the size of the graph, expect in the case where the optimal cut is sufficient weak compared to the others. We reiterate that, in contrast to e.g. spectral algorithms, the time taken by our algorithm depends on the structure of the graph and the cost being minimized, and hence much remains to be done in connecting the number of time-steps with graph properties. We believe that our earlier study of the settling rates of influence model-based agreement protocols (see [68]) may permit analysis of the settling rate of our partitioning algorithm. A complexity analysis of our model will also permit further comparison of our algorithm's performance with those of other algorithms.

- **Anti-Copying Algorithm for Bisection.** Bisection and recursive bisection are of significant interest in the parallel processing community, and remain hard for some graph structures [296]. In its current form, our algorithm is not well-suited for bisection problems: the constraint of equal-mass components is rarely satisfied by the partition generated at each time-step, so the algorithm is likely to take many time-steps to identify the optimal solution. Further, the bisection solution need not have connected components (even in the symmetric case), and hence the optimal solution may not be found at all. However, we believe that a simple modification of the influence model algorithm can permit bisection. To conceptualize this algorithm, we note that the bisection problem can be rephrased as an unweighted max-cut problem for a certain dual graph [298], and hence that we can seek an algorithm for identifying max-cuts to solve the bisection problem. An interesting approach for finding maximum cuts is to use an two-status influence model in which each site copies the opposite of its determining neighbor's status. This model favors configurations in which nearby neighbors' statuses are different from each other, and so has a tendency to find large cuts. This may turn out to be a computationally effective technique for finding maximum cuts and hence solving bisection problems.

- **Reference-Free Partitioning.** The sequential selection of reference vertex sets required in our algorithm is undesirable when a large number of partitions is needed. We are exploring variants of the influence model-based algorithm that do not require selection of reference generators. One interesting strategy is to initialize the sites in the influence model with different statuses, and then update the model until the desired number of partitions is identified. This solution can possibly be selected as the partition, or it can be used as a pre-partition based on which reference generators are selected influence-model algorithm is applied.

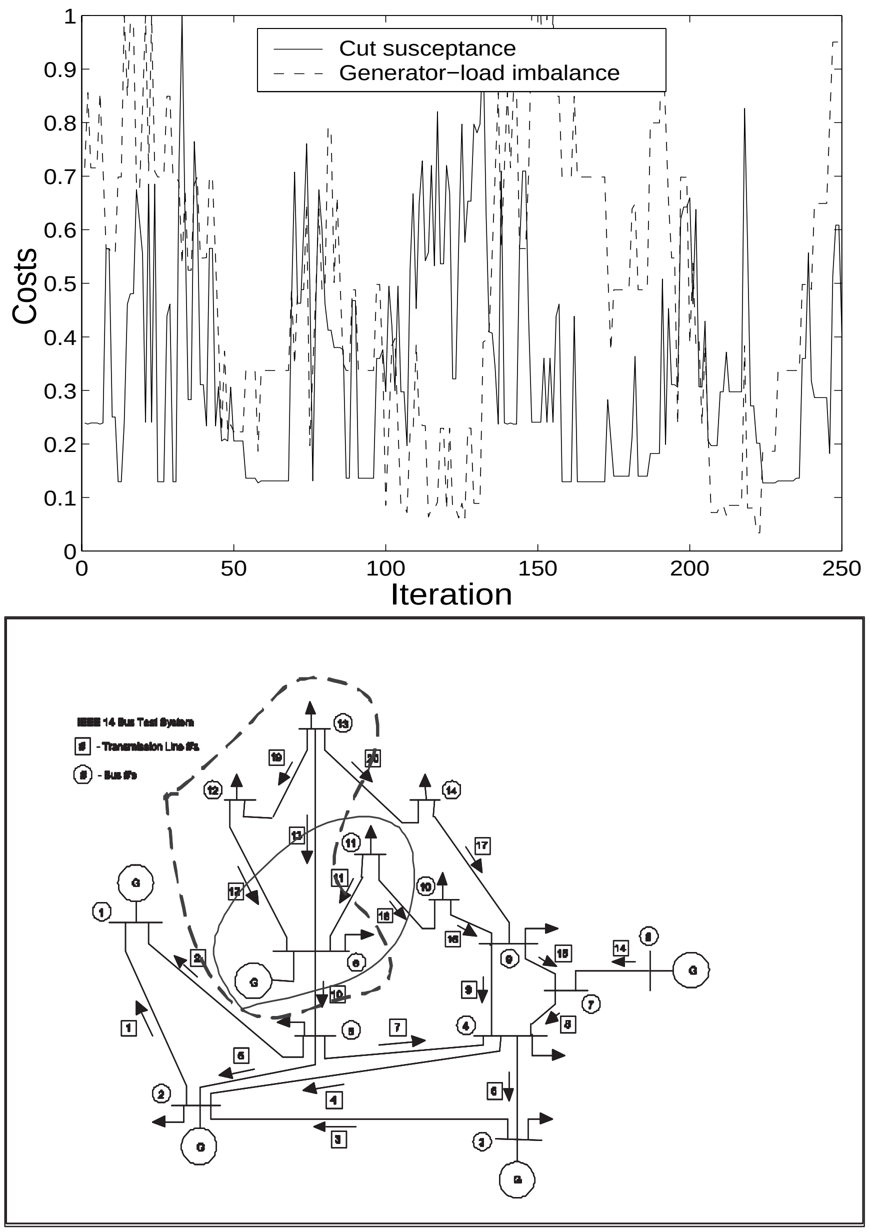
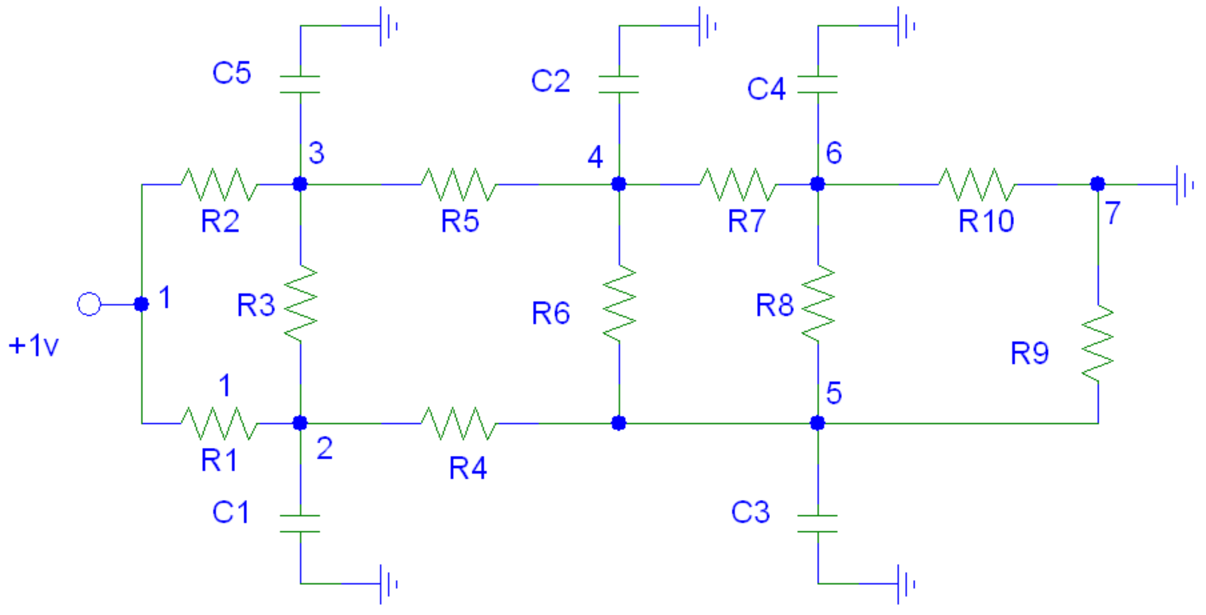


Fig. 20.3: We consider partitioning the Standard 14-Bus IEEE Test System, for the purpose of isolating a disturbance at Bus 6 from the slack bus 1. The upper figure shows the cut susceptance and generator-load imbalance for a sequence of 250 partitions generated by our influence model algorithm. The lower figure highlights that the minimum-cost partitions according to these two criteria are different. In the lower figure, the solid line indicates the minimum susceptance cut while the the dashed line indicates minimum generator-load imbalance.



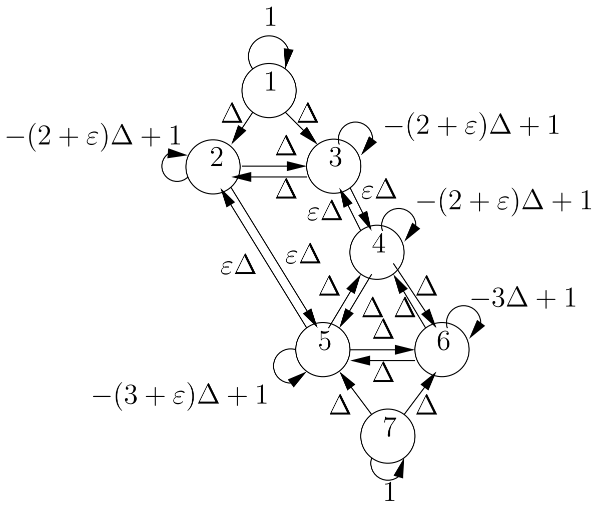


Fig. 20.4: Seven-node circuit and its influence model: Conductance values of R_4 and R_5 are ε, and all the other conductances are 1; all the capacitances are 1; Δ stands for the discretization step.

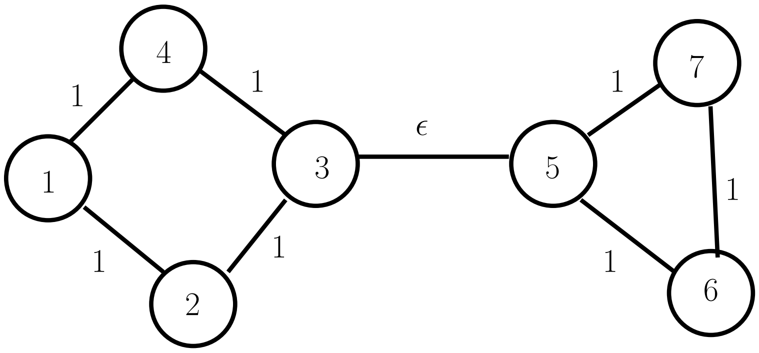


Fig. 20.5: Example for distributed partitioning

21. UNCERTAINTY EVALUATION THROUGH MAPPING IDENTIFICATION IN INTENSIVE DYNAMIC SIMULATIONS

We study how the dependence of a simulation output on an uncertain parameter can be determined, when simulations are computationally expensive and so can only be run for very few parameter values. Specifically, the methodology that we develop—known as the probabilistic collocation method (PCM)—permits selection of these few parameter values, so that the mapping between the parameter and output can be approximated well over the likely parameter values, using a low-order polynomial. We give several new analyses concerning the ability of PCM to predict the mapping structure as well as output statistics. We also develop a holistic methodology for the typical case that the uncertain parameter's probability distribution is unknown, and instead only depictive moments or sample data (which possibly depend on known regressors) are available. Finally, we pursue application of PCM to weather-uncertainty evaluation in air traffic flow management in some detail, and briefly introduce applications in two other domains.

21.1 Introduction

Many large-scale systems (e.g., power systems, air traffic networks, VLSI circuits, and biological regulatory networks) have complicated parameter-dependent dynamics that can only be examined in detail using time-consuming simulations. When these parameters are uncertain, it is often critical to find the dependence of specific simulation outputs on the parameters. However, because these large-system simulations are usually computationally costly, running exhaustive simulations over the range of potential parameter values is not practical, especially for real-time applications. An alternative is to approximate the mapping between the parameters and simulation output with a combination of basis functions, for instance as a polynomial [302, 303, 306]. With this approach, simulating the system at only a limited number of smartly-chosen sample parameter values is often sufficient to identify the mapping. In this chapter, we study representation of the mapping between *uncertain parameters*—whose statistics may in general need to be inferred from data or may only be partially known—and *outputs* from time-intensive simulations, using polynomials.

The problem of characterizing a static nonlinear mapping between parameters and outputs from data has been extensively studied in both the black-box system identification and statistical learning communities (though the problem is typically viewed as one of identifying an unknown system, rather than seeking a simple representation for a computationally-expensive map), see e.g. [308, 309]. However, the problem that we study differs from the basic problems addressed in these domains, in several senses. First, we can generate the output for *only very few sets of parameters*, but we can select these parameter sets as we wish and can expect high-fidelity outputs from the simulations. Given the very limited number of available parameter-output pairs and the high fidelity of these outputs, it is sensible for us to seek a low-order representation of the mapping that matches the output at the simulation points. That is, our focus here is on reducing error with barely sufficient data, rather than on more typical statistical learning problem of avoiding overfitting through e.g. regularization. Specifically, we study how to properly choose simulation points so that this low-order map is accurate over at least the *likely* parameter values.

The task of fitting the mapping between simulation parameters and outputs with polynomials is thus an *interpolation* task. There is an abundant literature on choosing sample points for polynomial interpolation, see e.g. [310]. It is well known that the equally-spaced sample points cannot avoid *Runge's phenomenon*—the interpolation error may increase without bound near the ends of the interpolation interval for certain mappings—and so unequally-spaced point selections such as the *Chebyshev nodes* are often preferable. For our mapping-identification problem, we further need to take advantage of the probabilistic description of the parameters, so as to identify the mapping over the *likely* parameter values. In this chapter, we advance the Probabilistic Collocation Method (PCM)—which was originally introduced in the global climate modeling and electric power systems communities [302–307]—as a technique for choosing simulation points to find a low-order polynomial mapping, when probabilistic descriptions of the parameters are available.

Specifically, PCM uses such a probabilistic description to intelligently choose simulation points, and hence to find a low-order polynomial mapping which predicts the average output correctly even if the actual mapping is a higher-order polynomial [302, 303]. Experimental evidence suggests that in fact PCM can accurately identify polynomial and non-polynomial mappings over the whole range of likely parameter values (not just the output mean), but the performance has not been determined analytically. One of our goals here is to further characterize the performance of PCM, in the process introducing the method as a generic tool for simulation under parametric uncertainty and clarifying its connection to statistical learning and nonlinear-identification methods (Section 21.2).

In practice, a parameter's probability distribution is rarely given explicitly. Commonly, we may only have a set of samples based on the probability distribution (as is typical in statistical learning problems), which e.g. have been collected from historical records. In other cases, we may only know some depictive low moments (e.g., mean and standard deviation). For these partial specifications of the parameter's probability distribution, how to select a good set of simulations to identify the mapping needs to be explored. This selection is influenced by the fact that the PCM algorithm uses *moments* of the uncertain parameter, and hence estimation of these moments is needed when the distribution is not explicitly given. Here, we give a comprehensive development of PCM, by examining the cases where the parameter distribution is not fully known (Section 21.3). We also consider the case that the parameter depends on known regressors (Section 21.4).

PCM is widely applicable to problems where a deterministic mapping between a stochastic parameter and system output must be found using time-consuming simulations. We introduce three applications, in the areas of air traffic flow management under uncertainty, power network fault evaluation, and cell-level simulation in biology (Section 21.5), and pursue the air traffic management application in some more detail. We pursue in some detail the air traffic flow management application, which is concerned with predicting delays due to fog at San Francisco airport, see e.g., [2, 312] for background. In the interest of space, we only summarize the other applications.

For clarity, we focus on the case with scalar uncertain parameter, output, and regressor. However, the development readily generalizes to the multivariate case, see [302, 303] for background on the multivariate case.

21.2 PCM for a Known Parameter Probability Distribution

In this section, we review and further characterize PCM, in the nominal case that the probability distribution for the uncertain parameter is known. Specifically, we first motivate and review the method and its output-mean prediction property, and then we give several new results concerning its performance.

Formally, we are concerned with identifying/representing a static deterministic mapping between a stochastic parameter X with known probability distribution $f_X(x)$ and a simulation out-

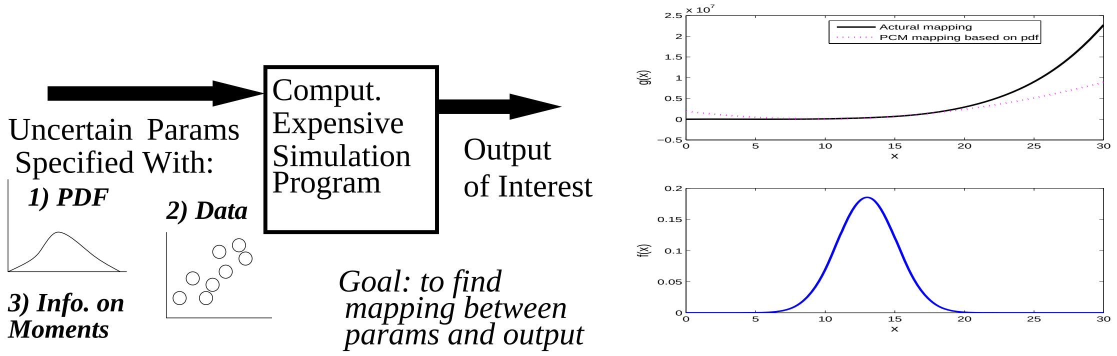


Fig. 21.1: a) PCM finds a polynomial mapping between a stochastic parameter X and a simulation output $g(X)$. b) Comparison of a degree-5 polynomial mapping with the 2nd-order PCM approximation (obtained from the true probability distribution); c) The distribution of the stochastic parameter X.

put $g(X)$, using a polynomial basis (Figure 21.1). Simulation of the mapping (computation of the output for a particular parameter value) is expensive or time-consuming, and so we are only able to simulate the mapping at n **points** (parameter values). The purpose of PCM is to choose the appropriate set of n points for simulation, and so to accurately represent the mapping over the domain of *likely* parameter values. In turn, the probability distribution for the output can be approximated if desired.

For this problem, typically we can only expect to obtain a low-order approximation of the actual mapping, since the output is available for so few simulation points. Here, we pursue such low-order representation using a polynomial basis, with the motivation that polynomials can represent many intricacies of a mapping and yet permit global representation/characterization of the mapping and hence the output probability distribution. When so few simulations are available, we must carefully select the simulation points, so that at least the approximate mapping is close to the real one over the range of likely parameter values. This viewpoint leads us to the following philosophy underlying PCM: simulation points are selected, and in turn the mapping is approximated, in such a manner that error between the actual mean output and the predicted mean output is small even if the actual mapping is of higher order than the predicted one. Formally, PCM is able to predict the expected output exactly using n simulations, whenever the actual mapping is any polynomial of degree at most 2n-1, i.e. even when the actual mapping is of much higher degree (or in other words more quickly varying).

The following is the three-step PCM algorithm:

1) Point selection. A set of n simulation points is chosen based on the probability distribution for the uncertain parameter, $f_X(x)$. Specifically, we compute the first n+1 **orthogonal polynomials** with respect to the input distribution $f_X(x)$ (see, e.g., [302] for an iterative method with low computational cost). That is, we compute the polynomials $h_0(x), \dots, h_n(x)$ of degrees $0, \dots, n$, respectively, such that $\int h_i(x)h_j(x)f_x(x)\,dx = \delta_{ij}$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The n roots of the degree-n orthogonal polynomial $h_n(x)$, denoted $x_0, \dots, x_{n-1}$, are chosen as the simulation points. It is easy to check that these **collocation points** fall within the the domain of random variable X, as long as the domain is a continuous set. We also note that computing $h_0(x), ..., h_n(x)$ requires use of the first $2n-1$ moments of X. We denote the ith moment of X as $\zeta_i = E(x^i) = \int f_X(x)x^i dx$.

2) Mapping identification. Using simulations at these n points, a degree-$(n-1)$ interpolating polynomial approximation $g^*(x)$ for the mapping $g(x)$ is generated. Specifically, we first determine $g(x_0), \dots, g(x_{n-1})$ through simulation of the mapping. Next, we approximate the mapping with the polynomial $g^*(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1}$, choosing the n coefficients $a_0, ..., a_{n-1}$ so that $g(x_0) = g^*(x_0), \dots, g(x_{n-1}) = g^*(x_{n-1})$. We note that, for typical applications of PCM, the evaluation of $g(x)$ at the n simulation points is the most expensive step computationally.

3) Output characterization. The probability distribution $f_g(g)$ of the output $g(X)$, as well as desired statistics of $g(X)$, are approximated based on the approximate mapping $g^*(x)$. Depending on the input distribution and the degree of $g^*(x)$, this computation can be done either analytically or numerically. In our studies, simple numerical computation of the cumulative distribution of $g^*(x)$ and subsequent computation of the probability distribution has been sufficient: simulation of the mapping is the computationally-intensive step, so even this simplistic implementation of the output-distribution calculation does not significantly change the analysis time. More elegant approaches for computing the output distribution—which are especially valuable when multiple parameters are uncertain—are based on expressing the input distribution in terms of certain standard distributions, including Normal and Beta distributions (see, e.g., [313]).

We refer to this algorithm as $(n-1)$th order PCM, since a degree-$(n-1)$ mapping is generated.

Several types of measures can be used to evaluate the performance of PCM, including those concerned with *1)* errors in the mapping, *2)* errors in the output distribution, and *3)* errors in output statistics, see [303]. Here, we shall only introduce a couple measures in the context of the performance analysis.

Let us begin the performance analysis by reviewing the mean output prediction property of PCM. It turns out that this mean-prediction property of PCM is identical to a classical result in the numerical integration or quadrature literature: specifically, it is known that a (weighted) integral of a degree-$(2n-1)$ polynomial function can be computed as a sum of the function value at *only* n properly-chosen arguments[*] (see e.g. [314]). Although the mean-prediction result has been described both in the numerical integration literature and in previous work on PCM (e.g. [303]), we include it here because it is central to the methodology.

Theorem 21.1. *Consider a mapping $g(x)$ that is a polynomial of degree at most $2n-1$, and let $g^*(x)$*

[*] This sparse integration technique is known as *Gaussian quadrature integration.*

be the $(n-1)$*th-order PCM fit for* $g(x)$*, for an uncertain parameter* X*. Then* $E[g(X)] = E[g^*(X)]$*.*

Proof: The degree-$2n-1$ polynomial mapping $g()$ can always be written (see e.g. [311]) in terms of the orthogonal polynomials $h_0(), \ldots, h_n()$, as $g(x) = h_n(x)(b_{n-1}h_{n-1}(x) + \ldots + b_0h_0(x)) + a_{n-1}h_{n-1}(x) + \ldots + a_0h_0(x)$, for appropriately chosen $b_{n-1}, \ldots, b_0, a_{n-1}, \ldots, a_0$. Say that we approximate $g(x)$ by the degree-$(n-1)$ polynomial $g^*(x) = a_{n-1}h_{n-1}(x) + \ldots + a_0h_0(x)$. We see that $g^*(x)$ is the (unique) degree-$(n-1)$ polynomial that matches $g(x)$ at the n collocation points, i.e. $g^*(x)$ is the PCM-generated mapping. Thus, we are interested in comparing $E[g(X)] = \int(h_n(x)(b_{n-1}h_{n-1}(x) + \ldots + b_0h_0(x)) + a_{n-1}h_{n-1}(x) + \ldots + a_0h_0(x))f_X(x)\,dx$ with $E[g^*(X)] = \int(a_{n-1}h_{n-1}(x) + \ldots + a_0h_0(x))f_X(x)\,dx$. Invoking orthogonality, we immediately obtain that $E[g(X)] = E[g^*(X)] = a_0$. $\square$

The mean-prediction property gives some indication of the capability of PCM as a fitting tool, since it reflects that PCM captures a feature that reflects the mapping over entire range of likely parameter values rather than only at a tangent point or over a small range. Here, we aim to enhance this basic performance analysis, by showing that the PCM can to some extent capture the structure of the mapping over the likely parameter range.

To this end, we first study the capability of PCM to predict *cross-statistics* between the parameter and output, in addition to the output statistics. In fact, we find that PCM can predict such cross-statistics even when the actual mapping is of higher order, as formalized in the following theorem:

Theorem 21.2. *Assume that we are using* $(n-1)$*th-order PCM to identify a mapping* $g()$ *that is actually a polynomial of degree* $n+m$*, for some* $m \in 0, \ldots, n-1$*. Then PCM correctly predicts the cross-statistics* $E[X^i g(X)]$ *for all* $i \in 0, \ldots, n-m-1$*. That is,* $E[X^i g(X)] = E[X^i g^*(X)]$*, for all* $i \in 0, \ldots, n-m-1$*.*

Proof: Notice that the actual mapping can be written in the form $g(x) = h_n(x)(b_m h_m(x) + \ldots + b_0 h_0(x)) + a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)$. Thus, we find that $E[X^i g(X)] = \int (h_n(x)(b_m x^i h_m(x) + \ldots + b_0 x^i h_0(x)) + x^i(a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)))f_X(x)\,dx$. For $i \in 0, 1, \ldots, n-m-1$, we notice that $b_m x^i h_m(x) + \ldots + b_0 x^i h_0(x)$ is a polynomial of degree less than or equal to $n-1$, and hence $\int h_n(x)(b_m x^i h_m(x) + \ldots + b_0 x^i h_0(x))f_X(x)\,dx = 0$. We thus recover that $E[X^i g(X)] = \int x^i(a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x))f_X(x)\,dx$. Since the PCM fit is $g^*(x) = a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)$, we find that $E[X^i g(X)] = E[X^i g^*(X)]$. $\square$

As a special case (corresponding to $m = n-2$), the theorem guarantees that the correlation between X and $g(X)$ is predicted correctly by $(n-1)$th-order PCM even when the actual mapping is of much higher order, of at most $2n-2$.

Let us also explore whether the nth-order PCM fit $g^*(x)$ is the minimum mean square error (MMSE) fit, i.e. whether $E[(g(X) - g^*(X))^2]$ is minimized over the class of all polynomials of degree n even when $g(x)$ is a higher-degree polynomial. Unfortunately, it is too much to hope for that such an MMSE fit can be obtained for all $g(x)$, using a small number of points. However, interestingly, we can use nth order PCM to obtain the MMSE estimator for $g(x)$, over a class of lower-degree (lower than n) polynomials. This notion is formalized in the following:

Theorem 21.3. *Consider a mapping $g(x)$ that is actually a polynomial of degree $n+m$, and consider using $(n-1)$th-order PCM to fit this mapping with the degree-$(n-1)$ polynomial $g^*(x) = a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)$. The mean square error of this fit cannot be improved through addition of any polynomial of degree less than or equal to n-1-m. Furthermore, the polynomial $g_r^*(x) = a_{n-1-m}h_{n-1-m}(x) + \ldots + a_0 h_0(x)$ is the MMSE fit for the mapping, among the class of polynomials with maximum degree n-m-1.*

Proof: Let us use the notation $\overline{g}(x)$ for the PCM fit with a degree $n-1-m$ polynomial added, i.e. $\overline{g}(x) = g^*(x) + c_{n-1-m}x^{n-1-m} + \ldots + c_0$. We notice that the mean square error between $g(X)$ and $\overline{g}(X)$ can be written as follows: $E[(g(X) - \overline{g}(x))^2] = E[(g(X) - g^*(X))^2] + E[(g(X) - g^*(X))(g^*(X) - \overline{g}(X))] + E[(g^*(X) - \overline{g}(X))^2]$. Let us consider the second term in this error expression. Substituting $g(x) = h_n(x)(b_m h_m(x) + \ldots + b_0 h_0(x)) + a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)$ and $g^*(x) = a_{n-1}h_{n-1}(x) + \ldots + a_0 h_0(x)$, we find that $E[(g(X) - g^*(X))(g^*(X) - \overline{g}(X))] = E[h_n(X)(b_m h_m(X) + \ldots + b_0 h_0(X))(c_{n-1-m}X^{n-1-m} + \ldots + c_0)]$. Finally, noting that $(b_m h_m(X) + \ldots + b_0 h_0(X))(c_{n-1-m}X^{n-1-m} + \ldots + c_0)$ is a polynomial of degree at most $n-1$, we see that $E[(g(X) - g^*(X))^2] \leq E[(g(X) - \overline{g}(X))^2]$, and the first part of the theorem is proved. Next, we prove that $g_r^*(X)$ is the MMSE estimate for $g(X)$ among estimators that are polynomials with maximum degree n-m-1. To do so, it is sufficient to prove that $E[X^i(g(X) - g_r^*(X)] = 0$, for $i \in 0, \ldots, n-m-1$. However, $E[X^i(g(X) - g_r^*(X)] = E[X^i(g(X) - g^*(X))] + E[X^i(g^*(X) - g_r^*(X))]$. The first of these two terms is 0, from Theorem 21.1. Meanwhile, notice that the second term can be written as $E[X^i(g^*(X) - g_r^*(X))] = E[X^i(a_{n-1}h_{n-1}(X) + \ldots + a_{n-m}h_{n-m}(X)]$, which equals 0 from orthogonality. Hence, the theorem is proved. $\Box$

Conceptually, Theorems 21.2 and 21.3 indicate that PCM optimally predicts certain features of the mapping as well as certain output statistics, with the orders of the optimally-predicted statistics/features growing gracefully with the number of points used.

Of course, we are also interested in the performance of PCM when the actual mapping is not polynomial. Unfortunately, for arbitrary mappings, there is no guarantee that any polynomial fit will be able to achieve small error (in an expected output or mean-square-error sense), even as more and more points are used. In fact, the error can become arbitrarily large with more points; this effect is known as the Runge phenomenon [310]. However, given certain limits on the higher derivatives of the actual mapping, the polynomial fit can be shown to converge quickly (i.e., with a small number of points) to the mapping. Gaussian quadrature (and hence PCM) are in the class of optimal algorithms for avoiding the Runge phenomena, in the sense that convergence is guaranteed given the weakest possible bound on the higher derivatives.

Let us conclude our our characterization of PCM, by noting that it complements methods for nonlinear black box system identification [308] as well as the polynomial chaos and general polynomial chaos methods [313]. First, let us consider the connection with non-linear black box system identification [308]. Such identification fundamentally requires characterizing static nonlinear mappings between inputs and outputs, which can be done through polynomial interpolation (though local methods such as spline-based techniques are more commonly used). PCM aims to achieve such identification with very few excitations (simulation or experiment trials at different parameter/input values), by smartly choosing excitations so as to find mapping accurately *for likely parameter/input values*. We note that, as compared to spline-based methods, PCM permits us to obtain mappings that have globally optimal or guaranteed performance with respect to the specified uncertain parameter's PDF. Second, we note that PCM naturally complements Wiener's classical *polynomial chaos* method and more recent work on *generalized polynomial chaos* (see [313]). These tools permit fast simulation of *known* deterministic dynamical models with random parameters and inputs, through expansion of random processes in terms of polynomials that are orthogonal with respect to the uncertain parameters' distributions. PCM uses a similar polynomial decomposition of the parameter vs. response mapping, but for the complementary task of identifying the mapping.

Example 1 In this simple example, we assume that the mapping between an uncertain parameter X and a simulation output is the fifth-degree polynomial $g(X) = X^5 - 6X^4 + 5X^3 - 4X^2 + 3X - 2$, where X is normally distributed with mean 13 and standard deviation 2.15. We are permitted only three simulations of the mapping, and require an accurate fit over the likely parameters. Using

PCM, we select simulation points and hence obtain a quadratic mapping, that approximates the actual mapping well over the likely parameter values (see Figure 21.1) and exactly predicts the mean output. □

21.3 PCM for an Unknown Parameter Probability Distribution

We discuss using PCM to construct a polynomial mapping between a parameter and simulation output, under the condition that the probability distribution of the parameter is not explicitly or fully known. We consider two common ways in which information about the distribution may be specified—1) through a set of sample parameter data, and 2) only through some descriptions of the low moments. We subsequently study the errors in the PCM points and fit resulting from the inexact knowledge of the parameter distribution.

Superficially, application of PCM seemingly requires estimation of the uncertain parameter's probability distribution. However, considering the PCM algorithm, we note that only some moments of the uncertain parameter are needed for PCM of a certain order, and so only estimation of these moments is critical. On the other hand, we expect that the PCM fit may be highly sensitive to certain errors in the moment estimates; thus, a comprehensive study of moment estimation together with mapping identification is important. This data-based approach is also in analogy with statistical learning and system identification techniques.

21.3.1 PCM based on Sample Parameter Data

Because we can often easily collect data on a parameter X from historical records, we often have available a large set of sample data drawn from the PDF (i.e., enough data that we can approximate well the cumulative distribution of X). In this case, a good polynomial mapping can be obtained directly from the sample moments calculated from the data. Specifically, recall that in order to get a nth-degree polynomial mapping through PCM (i.e., apply nth order PCM), we only need moments of X up to the order of $2n+1$. We call the process of using sample moments rather than the true moments to come up with a PCM mapping as the **sample moment-based PCM** (and refer to the standard PCM algorithm as pdf-based PCM, for clarity). We formalize in Theorem 21.4 that the sample moment-based PCM mapping approaches the one obtained from the inherent pdf, as the size of the data set becomes large.

Theorem 21.4. *Consider a data set $\{x_1, x_2, ..., x_n\}$ consisting of independent and identically distributed samples of a parameter X with distribution $f_X(x)$, that has finite moments. The mth-order sample moment-based PCM mapping approaches the mth-order pdf-based PCM mapping with probability 1 as $n \to \infty$.*

Proof: In order to prove that the polynomial mapping obtained from sample moments converges with probability 1 to the one obtained from the pdf $f_X(x)$, we only need to show the simulation points calculated from sample moments converge in probability 1 to those calculated from $f_X(x)$. As illustrated in Section 21.2, the determination of the mth order polynomial mapping needs $m+1$ simulation points that are the roots of the $(m+1)$th-degree orthogonal polynomial with respect to the weight function $f_X(x)$. Hence, we only need to show that the coefficients of the $(m+1)$th-degree orthogonal polynomial obtained from sample moments converge with probability 1 to those obtained from $f_X(x)$.

Notice that the $(m+1)$th-degree orthogonal polynomial $h_{m+1}(x) = x^{m+1} + a_{m+1,m}x^m + a_{m+1,m-1}x^{m-1} + ... + a_{m+1,0}$ is orthogonal to all the lower-degree polynomials, and hence orthogonal to 1 , x , ... , x^m. Thus, we can calculate the coefficients $a_{m+1,m}$, $...a_{m+1,0}$ from the following equation.

$$\begin{bmatrix} a_{m+1,m} & a_{m+1,m-1} & ... & a_{m+1,0} \end{bmatrix} = - \begin{bmatrix} \zeta_{m+1} & \zeta_{m+2} & ... & \zeta_{2m+1} \end{bmatrix} A^{-1}, \tag{21.1}$$

where $A = \begin{bmatrix} \zeta_m & \zeta_{m+1} & ... & \zeta_{2m} \\ \zeta_{m-1} & \zeta_m & ... & \zeta_{2m-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \zeta_1 & ... & \zeta_m \end{bmatrix}$. It can be easily shown that the matrix A is invertible (using the fact that $E\left[(a_1 X^m + a_2 X^{m-1} + ... + a_{m+1})^2\right]$ is positive for all a_i that are not simultaneously 0. Now consider replacing each moment ζ_i in the coefficient equation by the sample moment $\bar{\zeta_i}$, defined as $\sum_{j=1}^{n} \frac{(x_j)^i}{n}$, where n is the size of sample data set. We denote the changed matrix A as $\bar{A}$. The law of large numbers tells us that $\bar{\zeta_1}$ converges with probability 1 to ζ_1, and similarly (based on applying the law of large numbers to the random variable x^i) $\bar{\zeta_i}$ converges with probability 1 to ζ_i for each i, see [107]. Hence, $\bar{A}$ converges with probability 1 to A. This convergence also guarantees the invertibility of $\bar{A}$ with probability 1 (in the limit of large n). Therefore, the coefficient $\bar{a}_{m+1,i}$ obtained by using sample moments converges with probability 1 to $a_{m+1,i}$. $\square$

Theorem 21.4 shows that PCM can be used to identify polynomial mappings even when the parameter's probability distribution is unknown, by using sample moments directly calculated from a data set. We notice that this method does not require computation of the probability distribution. We also note that stochastic convergence of sample-based PCM is an analog to the deterministic convergence of discretization-based Gaussian quadrature (used for numerical stability, see e.g. [315]).

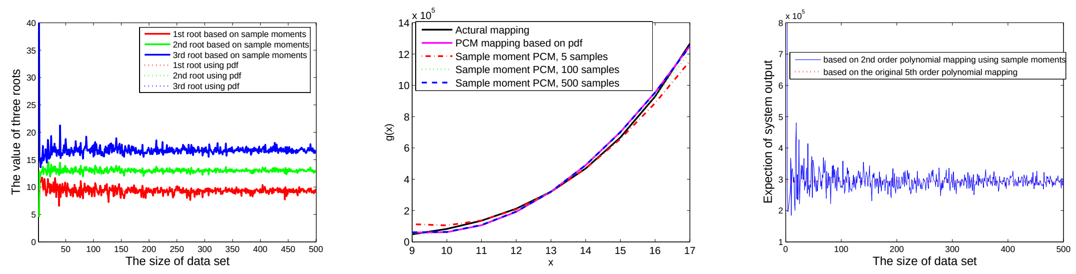


Fig. 21.2: a) The points obtained from sample moment-based PCM converge to those obtained from pdf-based PCM. b) Comparison of a degree-5 polynomial mapping, its 2nd-order pdf-based PCM approximation, and approximations obtained from sample moments; c) The output mean obtained from the sample moment-based PCM converges to the one obtained from pdf-based PCM, and hence in this case to the actual output mean.

Example 2 We apply sample moment-based PCM to the example from Section 21.2. Specifically, we assume that the pdf of X is unknown, and we only have a data set comprising independent samples drawn from the distribution. As seen from Figure 21.2, the PCM points generated by sample-moment PCM approach those from PDF-based PCM, for large data sets. We can also see from that the polynomial mapping obtained from sample-moment PCM approaches that obtained from the pdf-based PCM. Also, the output mean approaches that obtained from pdf-based PCM, and hence (in this case) approaches the true output mean. □

However, we sometimes have available only a (relatively) small set of parameter samples. In these limited-data cases, the sample moments are more likely to be away from the actual moments. In this case, we must thus decide between 1) using sample moment-based PCM, and 2) directly fitting the data with a typical probability distribution (e.g., uniform, Gaussian, or four-parameter Beta), and then applying (pdf-based) PCM to find the mapping. Which method to use depends on the size of the sample set and our a priori knowledge of the parameter's probability distribution, e.g., if the distribution is a typical one or not, see the classical statistics literature for details [316]. Generally, the second approach prevents overfitting of peculiarities in the data, and so tends to be more robust to errors when the data set is sufficiently small. The form of the probability distribution that should be used in the second approach is application-dependent. Gaussian, uniform, and four-parameter Beta distributions are commonly used, see e.g. [316–318].

In this case, after a type of distribution is selected, we can estimate the parameter(s) of the distribution (e.g. end points of the uniform distribution, four parameters of the Beta distribution) by using one of the following classical methods [316]: *1)* Maximum Likelihood estimation, *2)* method of matching moments, and *3)* least square regression of the cumulative distribution function. The validity of the assumed distribution form can be determined using hypothesis-testing methods such as the *chi-square test* [316].

We note that the intrinsic probability distribution may differ from the form that we have chosen, and hence we expect some error in fitting the data set with these distributions. Thus, we recommend sample moment-based PCM when sufficient data is available.

21.3.2 PCM with Knowledge of Only Some Low Moments

In some special cases, we may only know a few descriptive low moments of a random parameter. For example, operators of the power system may know through experience that a particular load has average magnitude of 10,000W with a variation 1000W, although no data has been formally recorded. In this setting, it is natural to fit the low moments with a typical probability distribution. Then we can use the moments generated from the pdf for finding the PCM mapping, as we have done in Section 21.3.1. Unfortunately, the typical distribution function with which to match moments is not easy to choose (in contrast to the small-dataset problem discussed in Section 21.3.1), since without sample data it is impossible to use hypothesis testing methods to determine the appropriateness of the selected pdf form. What we really care about, however, is how the change of the distribution affects the identified mapping. From this viewpoint, which distribution we choose affects the higher moments, and hence the locations of PCM points; we refer the reader to Example 3 below for an illustrative example comparing two distributional fits. Also of relevance, we will systematically study how the PCM points move with the higher moments in Section 21.3.3.

Example 3 For illustration, we compare the PCM points for uniform distribution and Gaussian distribution with the same mean and variance (Table 21.1). We see that the PCM points based on the Gaussian distribution are more spread out compared to those based on the Uniform distribution, which is sensible since the uniform distribution is bounded. This suggests that if we wish to take into account parameter values far away from the mean (albeit at the cost of less accurate estimates near the mean), it is better to use the normal distribution. Another advantage is that this mapping can better eliminate Runge's Phenomenon over the likely ranges of a parameter compared to the Uniform distribution-based PCM mapping, supposing the mapping is indeed normal distributed. Our performance analysis in the next section will make clear these tradeoffs in the choice of higher moments for PCM. □

21.3.3 Higher Moments' Impact on PCM

Whatever the methods used to obtain moment estimates, we expect some error in these estimates, which in turn will result in errors in the simulation points and hence in the mapping obtained as compared to pdf-based PCM. The dependence of the zeros of orthogonal polynomials on parameters of the weighting function (in our case, the probability distribution) has been widely studied [320]. We require extension of these results in the literature, for several reasons: 1)

Tab. 21.1: Comparison of PCM points between uniform distribution and Gaussian distribution, both with mean as 8 and variance as 8.

	Roots of 1st order Polynomial	Roots of 2nd Order Polynomial	Roots of 3rd Order Polynomial	Roots of 4th Order Polynomial
Uniform Distribution	8.000	5.172, 10.828	4.205, 8.000, 11.795	3.781, 6.334, 9.666, 12.219
Gaussian Distribution	8.000	5.172, 10.828	3.101, 8.000, 12.899	1.397, 5.901, 10.099, 14.603

typically, we do not have a parametrized form for the weighting function; 2) our interest is in the accuracy of the mapping rather than of the points themselves and so we must consider movement of all points. With these goals in mind, we have studied the impact of higher-moment variations on the PCM points and fit in two ways: 1) using root-locus methods and 2) through sensitivity arguments.

First, by applying the classical *root-locus method* [321] and generalizations thereof, we are able to obtain majorizations of PCM points in terms of their moments, and also to characterize the extremes of the root locations due to variabilities in certain moments. This approach allows us to study the impact of moment variabilities on PCM points, and hence the range of parameters for which the fit is accurate, in a sequential fashion: in particular, we find that PCM points move monotonically to the right toward the roots of a lower-order orthogonal polynomial with respect to the odd moments, while even-moment changes engender a more complex root movement with (possibly) a change in direction. Further, we find that the movement of the PCM points is particularly simple when the pdf is symmetric, i.e. the odd moments are 0. The results are summarized in the following theorems.

Specifically, let us begin by characterizing the dependence of the PCM points on the odd moments.

Theorem 21.5. *Consider that all moments of X up to $2n$ are fixed. When the moment ζ_{2n+1} moves from 0 to ∞, the $n+1$ PCM points that can be computed from these moments (i.e., the roots of h_{n+1}) all move to the right. The rightmost PCM point approaches infinity, and the remaining n PCM points converge to the roots of h_n.*

Proof: Let us show that the movement of the PCM points can be traced using a root locus, and hence prove the result. According to the recurrence for orthogonal polynomials, we have $h_{n+1} = (x - \frac{(xh_n, h_n)}{(h_n, h_n)})h_n - \frac{(h_n, h_n)}{h_{(n-1,n-1)}}h_{n-1}$, where (h_i, h_j) denotes the integral $\int h_i(x)h_j(x)f(x)dx$ [311]. Notice that h_{n+1} can be written as $(x - \frac{\zeta_{2n+1}+c_1}{c_2})h_n - \frac{c_2}{c_3}h_{n-1}$, where constants c_1 and c_2 are determined by moments up to $2n$, and constant c_3 is determined by moments up to $2n-2$. We can write the equation $h_{n+1} = 0$ into the root locus form $\frac{\zeta_{2n+1}}{-c_2}h_n(x) + \left(xh_n(x) - \frac{c_1}{c_2}h_n(x) - \frac{c_2}{c_3}h_{n-1}(x)\right) = 0$, where $\frac{\zeta_{2n+1}}{-c_2}$ is a root locus variable [321]. Hence, the movement of PCM points by changing ζ_{2n+1} can be studied using a root locus. To continue, we note that the roots of h_{n+1} are real and simple. They and those of h_n are interleaved [319]. From the root locus, when ζ_{2n+1} move from 0 to ∞, the left n roots of h_{n+1} move to the right starting from the zeros of $xh_n(x) - \frac{c_1}{c_2}h_n(x) - \frac{c_2}{c_3}h_{n-1}(x)$ to the zeros of h_n, and the rightmost one moves to ∞. $\square$

Theorem 21.5 shows the ranges of the PCM points with the highest moment flexible to move. With ζ_{2n+1} approaching infinity, all of the roots move to the right, with one approaching infinity, and the rest approaching the n PCM points of the $n-1$ degree mapping. Since a large ζ_{2n+1} indicates a large asymmetry, it makes sense that the PCM points shift toward the most likely taken ranges. Also, it can be easily seen that when ζ_{2n+1} moves from 0 to $-\infty$, the rightmost n PCM points move to the left to the zeros of h_n, and leftmost one moves to $-\infty$.

Let us next study the dependence on the even moments, when the PDF is assumed even.

Theorem 21.6. *Consider that all odd moments up to $2n+1$ are 0 and the even moments are fixed except the highest one, ζ_{2n}. When the moment ζ_{2n} moves from its minimum possible value to ∞, one of the $n+1$ PCM points (roots of h_{n+1}) starts from 0, and the remaining points start from the roots of h_n. The right- and left- most PCM points move toward $\pm\infty$ respectively, and the rest approach the roots of h_{n-1}.*

Proof: Similarly to the proof of Theorem 21.5, we can write the expression $h_{n+1} = 0$ into a root locus form. Since all the odd moments are 0, both $f(x)$ and h_n^2 are even functions, and hence the equation $\frac{(xh_n,h_n)}{(h_n,h_n)} = 0$ holds. Therefore, $h_{n+1}(x)$ can be written as $xh_n - \frac{\zeta_{2n}+c4}{c5}h_{n-1}$, where both c_4 and c_5 are determined only by the moments up to $2n-2$. The root locus form is thus $\frac{\zeta_{2n}}{c_5}h_{n-1}(x) + (\frac{c_4}{c_5}h_{n-1}(x) - xh_n(x)) = 0$. The root locus form clearly shows that when ζ_{2n} approaches infinity, the PCM points approach the roots of h_{n-1} and $\pm\infty$. Now let us consider the root locations for small ζ_{2n}. To do so, we note that ζ_{2n} has a lower bound, which is determined by the non-negativity of $E\left[(a_1x^n + a_2x^{n-1} + ... + a_{n+1})^2\right]$ for any $a_i \in R$. When ζ_{2n} is at this lower bound, we can find $a_i \neq 0$ to make $E\left[h_n'^2\right] = 0$, where $h_n' = a_1x^n + a_2x^{n-1} + ... + a_{n+1}$. $E\left[h_n'^2\right] = 0$ holds only when $h_n' = 0$ since $h_n'^2 \geq 0$. Thus, h_n' is the nth order orthogonal polynomial h_n, since $(h_n', x^i) = 0$, for $i = 0, ...n-1$. So when ζ_{2n} is at this lower bound, we have $(h_n, h_n) = 0$ and hence according to the root locus, the roots are those of xh_n. Root locus rules yield that the roots move from those of polynomial xh_n to those of h_{n-1} and $\pm\infty$, without changing direction. $\Box$

In summary, for an even probability distribution, the root locus allows the study of PCM points' movement with variations of moments. The PCM points are always symmetric with respect to 0. Increasing the highest even moment drives the PCM points apart, starting from 0 and those of the

$n-1$ degree polynomial mapping toward those of the $n-2$ degree mapping and $\pm\infty$.

Finally, let us study the dependence on even moments for the general case; for convenience, we set the highest odd moment to zero, with the understanding that Theorem 21.5 allows us to subsequently characterize the dependence on it.

Theorem 21.7. *Consider that all moments up to $2n-1$ are fixed, the $(2n+1)$th moment is 0, and the $(2n)$th moment ζ_{2n} is flexible to move*[†]*. When ζ_{2n} moves from 0 to ∞, the PCM points change direction at most once. Eventually, the right- and left- most PCM points move toward $\pm\infty$ respectively, and the remaining PCM points approach the roots of h_{n-1}.*

Proof: Under the condition that the highest moment ζ_{2n+1} is 0, we can rewrite the recurrence as $h_{n+1} = (x - \frac{q\zeta_{2n}+c_1}{\zeta_{2n}+c_2})h_n(x) - \frac{\zeta_{2n}+c_3}{c_4}h_{n-1}$, where constants c_i, $i = 1..4$ are determined by moments up to $2n-1$, and q is a constant. Denoting $\zeta_{2n}+c_2$ as a new variable ζ'_{2n}, h_{n+1} can be rewritten as $h_{n+1} = f_{n+1}(x) - \frac{f_n(x)}{\zeta'_{2n}} - f_{n-1}(x)\zeta'_{2n}$, where $f_{n+1}(x) = (x-q)h_n(x) + \frac{c_2-c_3}{c_4}h_{n-1}(x)$, $f_n(x) = (c_1 - qc_2)h_n(x)$, and $f_{n-1}(x) = \frac{h_{n-1}(x)}{c_4}$. Hence the roots of $h_{n+1}(x)$ are the roots of the equation $f_{n-1}(x)\zeta'^2_{2n} - f_{n+1}(x)\zeta'_{2n} + f_n(x) = 0$. From the equation, we can see that each root x is associated with at most two ζ_{2n}. Hence when ζ_{2n} increases, the movement of the roots may only change direction only once. The limit of the locations when $\zeta_{2n}\to\infty$ can be studied from $h_{n+1} = (x - \frac{q\zeta_{2n}+c_1}{\zeta_{2n}+c_2})h_n(x) - \frac{\zeta_{2n}+c_3}{c_4}h_{n-1}$. When $\zeta_{2n}\to\infty$, the second term dominates. Hence the roots approaches those of h_{n-1} and $\pm\infty$. $\square$

Second, we use sensitivity notions to specify the relationship between moment errors and errors in the PCM points, fit, and output statistics. Using these sensitivity results together with moment-

[†] We can make this assumption that the moment is 0 WLOG because Theorem 21.5 can subsequently be used to study the movement of the PCM points due to changes in the odd moments.

estimation error expressions from classical statistics, we can estimate error percentages for PCM points and/or output characteristics, and hence decide whether PCM of a certain order is viable. The sensitivity characterization follows naturally from existing work, and so we omit the details. Briefly, we use a clever description of the PCM points as solutions to an eigenvalue equation, see e.g. [315]. From this reformulation as an eigenanalysis, we can determine sensitivity of the points to moment errors using classical eigenvalue sensitivity notions. The moment errors can be related to the size of the available data set using standard statistics notions, and hence percentage errors in the PCM points and in the fit can roughly be related to characteristics of the data sets. It is worth noting that these error expressions often depend on the moments of X themselves, in which case sample moments must be used to estimate the errors.

21.4 PCM when Parameters Depend on Regressors: Brief Overview

In many applications, an input parameter's probability distribution is dependent on one or more known regressors. For example, load values in a power network may depend on the season, and chemical concentrations in a biological regulatory network may depend on other measured concentrations. In consequence, the locations of PCM points and hence the PCM mappings are also dependent on these regressors. In these applications, we need to identify the mapping between a parameter and the system output for likely parameter values, conditioned on the value(s) of the particular regressor(s). In this case, we can apply PCM based on the conditional probability distribution of the parameter given the regressors, however these conditional PDFs are typically unknown and instead must be inferred from data. In this section, we assume that regressor-parameter pair sample data is available, and suggest strategies for regression and subsequent application of PCM for data sets of various fidelities. Because our focus in this chapter is on mapping-identification, our discussion of regression here is brief and informal: we wish to expose the strategies for applying PCM without concerning ourselves with the details of proof regarding regression techniques.

Large Data Set Suppose we have a large data set of randomly sampled regressor-parameter pairs (i.e., sufficient data at each discrete regressor value or over a small range of continuous regressor values). Then we can calculate PCM points for a known regressor value in two steps.

1) We collect the parameter samples at or in a small range around the regressor value of interest.

2) We use the sample moment-based PCM method (see Section 21.3.1) to obtain the polynomial mapping.

When the size of the data set becomes arbitrarily large over small ranges of the regressor, the PCM mapping obtained using this method approaches the one obtained from the inherent conditional probability distribution with probability 1 (given some weak conditions on the continuity of the conditional distribution). However, we may in general need a very large data set for accurate mapping identification.

Moderate-Sized Data Set In most applications, the conditional distribution of the parameters and hence the parameter moments change smoothly with the regressors. Hence, we can find a low-order relationship between the regressors and parameter moments, as follows:

1) We use classical regression methods to find the relationship $\hat{m}_i(r)$ between the regressor r and the ith sample moment (or ith power) of the parameter, see [317] for details. Notice that polynomials or other bases can be used for the fit, and that typically higher moments require more basis functions.

2) For a particular regressor value r_k that we wish to study, we read off all the sample moments $\hat{m}_i(r_k)$, and use the sample moment-based PCM methods to come up with the PCM mapping.

Small Data Set Let us consider the case where we only have a small set of regressor-parameter sample data pairs. With such limited information, it is most effective to fit the data assuming a typical distribution form for each regressor value, and a low-order dependence of the distribution parameters (typically moments) on the regressor. The typical distribution that we choose is application-specific.

We use the following procedure to finding the PCM mapping in this case:

1) We choose a typical distribution form that reasonably describes the parameter's conditional distribution in this application. Let us assume that the distribution is specified by its first q moments.

2) We calculate the ith power of the parameter (denoted by m_i) for each regressor value r_j, for $i = 1, \ldots, q$.

3) We use regression methods to fit the data from step 2 with basis functions, thus obtaining approximations $\hat{m}_i(r)$ for the sample moments. Hence, we can approximate the conditional distribution of the parameter given the regressor (denoted as $f(x|r)$).

4) For a particular regressor value r_k that we want to study, we calculate all the moments needed for PCM from the distribution $f(x|r_k)$, and use the moment-based PCM methods to obtain the mapping.

21.5 Applications

In this section, we introduce several applications of PCM in identifying mappings between uncertain parameters and simulation outputs. In the interest of space, we only pursue the air traffic management application in some detail, and briefly introduce the other ones.

Prediction of Delays in Air Traffic Flow Management There is abundant literature on optimizing air traffic flow management strategies to minimize traffic delay under capacity constraints (e.g., limited controller resources, limited runway resources, etc) [2, 93, 291]. These optimizations are typically done using highly simplified and deterministic models for traffic flow. Hence, in order to practically evaluate a strategy, accurately finding the corresponding traffic delay is critical. However, the prediction of delay under a management strategy is not a easy task, for two reasons. First, the United States national airspace system is so large and strongly interconnected that the delay can often only be predicted using a time-consuming simulation tool, e.g., the Future ATM Concepts Evaluation Tool (FACET) [322]. Second, delay is highly dependent on uncertainties such as weather conditions. Severe weather events (whose timings are often uncertain) can cause significant delay since capacity constraints fall significantly compared to those under normal weather conditions [312]. Here, we suggest using PCM to find mappings between uncertain weather parameters and delays, and hence predict the distribution of delays, without running extensive simulations over all possible weather conditions. We illustrate this approach for delay prediction with a canonical example concerning congestion at San Francisco International Airport (SFO).

SFO is well known for its morning fog during the summer months. The fog usually settles in around midnight and remains in place until clearing during the morning or early afternoon. Under normal weather conditions, aircraft are allowed to land parallel on two close runways, however, when fog is present, this is considered unsafe and only one runway can be used, causing delay. Hence the fog clearing time is a key parameter affecting traffic delay. To minimize en route delays and backups, aircraft are held at their origins (through a mechanism known as a *ground delay program*, or GDP). The extent and time of the GDP is typically set based on a prediction of the fog clearing time, and so understanding the mapping between the actual fog clearing time and the

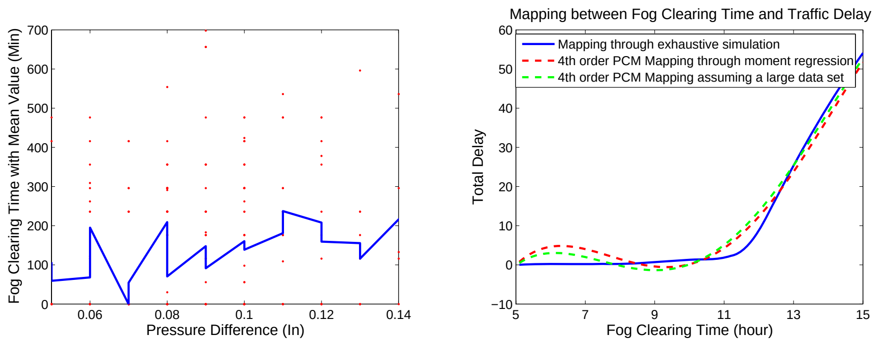


Fig. 21.3: a) The dataset showing fog clearing time at SFO versus the pressure difference between SFO and SAC. Each red point is a data collected. The blue line shows the average of clearing time associated with each particular pressure difference. b) Mapping between fog clearing time and traffic delay

consequent delay over the range of likely fog-clearing times is valuable.

A crude but effective method for predicting the fog clearing time at SFO on a particular day is based on the pressure difference between San Francisco and Sacramento (SAC) the day before. Thus, we can view the pressure difference as a known regressor for the fog-clearing time distribution[‡]. Figure 21.3 shows the relationship of clearing time at SFO (the parameter) with the pressure difference between SFO and SAC (the regressor) on the previous evening. Given that the pressure difference is 0.09in on a particular evening, we wish to determine the relationship between the fog clearing time and the resultant delay (the output of interest) for likely clearing times, using only a small number of simulations.

We have used the regression methods for large- and medium-sized data sets to estimate the conditional moments, and hence to find a 4th order PCM mapping between fog clearing time and the delay, when the pressure difference is 0.09in. The PCM-based mappings and the mapping found through running the simulation algorithm exhaustively are shown in Figure 21.3. PCM identifies

[‡] Many other factors can be chosen as regressors, e.g., the outputs of weather forecasting programs. In this illustration, we choose the pressure difference as a regressor.

the mapping quite well over likely parameter values.

Load Uncertainty Evaluation for Power System Fault Dynamics Simulations of electric power system dynamics are widely used, e.g. for characterizing system responses to faults. The relative prevalence of various load types (i.e., inductive vs. lighting loads) at the time of a fault is stochastic, and yet this ratio of load types can significantly impact the dynamic response. Thus, obtaining the mapping between load-type prevalences and certain features of the dynamic response, such as the minimum voltage reached at a particular bus during the transient, is very important for fault-evaluation studies. A complementary use of such mappings is in post-fault analysis, where the inverse mapping can be used to determine the load-type prevalences at the time of the fault.

Given the computational cost of simulations for electric power system dynamics, we suggest using PCM to identify the mapping with sparse simulation, albeit at the cost of only obtaining an accurate mapping at the likely parameter values. Like our other applications, we note that the probability distributions of the load-type parameters are not well known, and so the data-based approach developed here is needed.

Integrative Simulation in Cell Biology Detailed system-level simulation in biology (of cells, groups of cells, or whole organisms) can permit better understanding, and eventually design and control, of complicated biological behaviors. Such simulations (see, for instance, the *E-cell* and *Virtual Cell* programs [323, 324]) are typically very complicated and hence time-consuming. For instance, even to simulate a single cell's dynamics, we need to consider gene expression, molecule diffusion, signal transduction, biochemical reaction, etc [325, 326].

Understanding the impact of the many uncertain parameters in a biological system is necessary, to identify the key factors that modulate its behavior. For instance, in the epidermal growth factor receptor (EGFR) pathway related to tumor formation, the concentration of molecules like Shc, Sos, $Grb2$ and $EGFR$ affect the concentration of activated oncogene Ras, and eventually cell devision [327]. In order to understand the impact (the mapping between the parameters and critical concentration), we have to rely on simulations at various parameter values, and hence PCM is a valuable tool. As in the other applications, we note that the concentration parameters' ranges must be inferred from data, and so the comprehensive methodology developed here is needed.

BIBLIOGRAPHY

[1] Y. Wan, S. Roy, and A. Saberi, "Designing spatially-heterogeneous strategies for control of virus spread," *IET Systems Biology*, vol. 2, No. 4, pp. 184–201, 2008. Short version in *Proceedings of 46th IEEE Conference on Decision and Control*, New Orleans, Louisiana, 12-14 December, 2007.

[2] Y. Wan and S. Roy, "A scalable methodology for evaluating and designing coordinated air traffic flow management strategies under uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, in press. Short version in *Proceedings of AIAA 2007 Guidance, Navigation, and Control Conference*, Hilton Head, South Carolina, 20-23 August, 2007.

[3] Y. Wan and S. Roy, "Sensitivity of national airspace system performance to disturbances: modeling, identification from data, and use in planning," in *Proceedings of AIAA 2008 Guidance, Navigation, and Control Conference*, Honolulu, Hawaii, 18–21 August, 2008.

[4] S. Roy, J. Krueger, D. Rector, and Y. Wan, "A network model for activity-dependent sleep regulation," *Journal of Theoretical Biology*, vol. 253, pp. 462–468, 2008. Short version in *Proceedings of 2008 American Control Conference*, pp. 1400–1405, Seattle, Washington, 11-13 June, 2008.

[5] Y. Wan, S. Roy, and A. Saberi, "A new focus in the science of networks: toward methods for design," *Proceedings of the Royal Society A*, vol. 464, pp. 513–535, March, 2008.

[6] Y. Wan, S. Roy, X. Wang, A. Saberi, T. Yang, M. Xue, and B. Malek, "On the structure of graph edge designs that optimize the algebraic connectivity," in *Proceedings of 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 9–11 December, 2008.

[7] S. Roy, Y. Wan, and A. Saberi, "On time-scale designs for networks," *International Journal of Control*, in press. Short version in *Proceedings of 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 9–11 December, 2008.

[8] S. Roy, Y. Wan, and A. Saberi, "Majorizations for the dominant eigenvector of a nonnegative matrix," in *Proceedings of 2008 American Control Conference*, pp. 1965–1966, Seattle, Washington, 11-13, 2008.

[9] S. Roy and Y. Wan, "An explicit formula for differences between laplacian-eigenvector components using coalesced graphs," submitted to *Linear Algebra and its Applications*.

[10] Y. Wan, S. Roy, A. Saberi, and A. Stoorvogel, "A multiple-derivative and multiple-delay paradigm for decentralized controller design: introduction using the canonical double-integrator network," submitted to *Automatica*. Short version in *Proceedings of AIAA 2008 Guidance, Navigation, and Control Conference*, Honolulu, Hawaii, 18-21 August, 2008.

[11] Y. Wan, S. Roy, A. Saberi, and A. Stoorvogel, "The design of multi-lead-compensators for stabilization and pole placement in double-integrator networks," submitted to *IEEE Transactions on Automatic Control*. Short version submitted to *2009 American Control Conference.*

[12] Y. Wan, S. Roy, A. Saberi, and A. Stoorvogel, "Semi-global stabilization of double-integrator networks with actuator saturation," submitted to *2009 American Control Conference.*

[13] S. Roy, A. Saberi, and Y. Wan, "On multiple-delay static output feedback stabilization of LTI plants," in *Proceedings of 2008 American Control Conference*, pp. 419-423, Seattle, Washington, 11-13 June, 2008, also submitted to *International Joournal of Robust and Nonlinear Control.*

[14] Yan Wan, Sandip Roy, Anton Stoorvogel, and Ali Saberi, "On multiple-delay approximations of multiple-derivative controllers," submitted to *Automatica*. Short version submitted to *2009 European Control Conference.*

[15] A. Stoorvogel, S. Roy, Y. Wan, and A. Saberi, "A class of neutral-type delay differential equations that are effectively retarded," submitted to *IEEE Transactions on Automatic Control*. Short version submitted to *2009 American Control Conference.*

[16] Y. Wan, S. Roy, and A. Saberi, "Explicit precompensator design for invariant-zero cancellation," *International Journal of Control*, in press.

[17] Y. Wan, S. Roy, and A. Saberi, "A pre- + post- + feedforward compensator design for zero placement," submitted to *International Journal of Control*. Short version submitted to *2009 American Control Conference.*

[18] S. Roy, Y. Wan, A. Saberi, and B. Malak, "An alternative approach to designing stabilizing compensators for saturating linear time-invariant plants," submitted to *International Journal of Robust and Nonlinear Control*. Short version in *Proceedings of 47th IEEE Conference on Decision and Control*, Cancun, Mexico, 9–11 December, 2008.

[19] Y. Wan, S. Roy, and A. Saberi, "A stochastic automaton-based algorithm for flexible and distributed network partitioning," *International Journal of Distributed Sensor Networks*, vol. 4, no. 3, pp. 223–246, 2008.

[20] S. Roy, Y. Wan, and A. Saberi, "A flexible algorithm for sensor network partitioning and self-partitioning problems," *2nd International Workshop on Algorithmic Aspects of Wireless Sensor Networks, 33th International Colloquium on Automata, Languages, and Programming*, Venice, Italy, 9-16 June 2006. *Lecture Notes in Computer Science*, vol. 4240, pp.152–163, Springer, 2006.

[21] Y. Wan, S. Roy, and B. Lesieutre, "Uncertainty evaluation through mapping identification in intensive dynamic simulations," submitted to *IEEE Transactions on Systems, Man, and Cybernetics, Part A*. Short version in *Proceedings of AIAA 2008 Guidance, Navigation, and Control Conference*, Honolulu, Hawaii, 18–21 August, 2008.

[22] R. Pastor-Satorras and A. Vespignani, "Immunization of complex networks," *Physical Reviews E*, 65(3):036104, 2002.

[23] C. Wang, J. C. Knight and M. C. Elder, "On computer viral infection and the effect of immunization," in *Proceedings of 16th Annual Computer Security Applications Conference*, New Orleans, Louisiana, December, 2000.

[24] S. Riley et al., "Transmission dynamics of the etiological agent of SARS in Hong Kong: Impact of Public Health Interventions," *Science*, vol. 300, pp. 1961-1966, June 20, 2003.

[25] S. Roy, A. Saberi, and K. Herlugson, "Formation and alignment of distributed sensing agents with double-integrator dynamics," IEEE Press Monograph on *Sensor Network Operations*, April 2006.

[26] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," in *Proceedings of the IEEE*, January 2007.

[27] O. Mason and M. Verwoerd, "Graph theory and networks in biology," *IET Systems Biology*, vol. 1(2), pp. 89-119, 2007.

[28] S. Wang and E. J. Davison, "On the stabilization of decentralized control systems," *IEEE Transactions on Automatic Control*, vol. AC-18, pp. 473-478, October 1973.

[29] S. Roy, A. Saberi, and P. Petite,"Scaling: a canonical design problem for networks," in the *Proceedings of the 50th American Control Conference*, Minneapolis,MN, June 2006. Extended version in the *International Journal of Control* (by S. Roy and A. Saberi), vol. 80, no. 8, pp. 1342-1353, 2007.

[30] R. M. Anderson and R. M. May, *Infectious Diseases of Humans Dynamics and Control*, Oxford University Press, 1991.

[31] J. Arino and P. van den Driessche, "The Basic Reproduction Number in a Multi-city Compartmental Epidemic Model," *Positive Systems*, LNCIS, vol. 294, pp. 135-142, 2003.

[32] E. Diaz, A. Urdapilleta, G. Chowell and C. Castillo-Chávez, "Ring Vaccination as a Control Strategy for Foot-and-Mouth Disease", in *2005 MTBI Summer Program*, Los Alamos, NM, 2005.

[33] M. Lipsitch et al., "Transmission dynamics and control of severe acute respiratory syndrome," *Science*, vol. 300, pp. 1966-1970, June 20, 2003.

[34] D. J. Daley and J. Gani, *Epidemic Modeling: An Introduction*, Cambridge University Press, 1999.

[35] D. J. Watts, R. Muhamad, D. C. Medina, and P. S. Dodds, "Multiscale, resurgent epidemics in a hierarchical metapopulation model," in *Proceedings of the National Academy of Science of the United States*, vol. 102, no. 32, pp. 11157-11162, August 9, 2005.

[36] P. van den Driessche, and J. Watmough, "Reproduction numbers and sub-threshold endemic equilibria for compartmental models for disease transmission," *Mathematical Biosciences*, vol. 180, pp. 29-48, 2002.

[37] H. W. Hethcote, "An immunization model for a heterogeneous population," *Theoretical Population Biology*, vol. 14, no. 3, December 1978.

[38] R. M. May and R. M. Anderson, “Spatial heterogeneity and the design of immunization programs,” *Mathematical Biosciences*, vol. 72, pp. 83-111, November 1984.

[39] M. J. Keeling and P. Rohani, “Estimating spatial coupling in epidemiological systems: a mechanistic approach,” *Ecology Letters*, 2002(5), pp. 20-29, 2002.

[40] A. L. Llyod and R. M. May, “Spatial heterogeneity in epidemic models,” *Journal of Theoretical Biology*, vol. 179, pp. 1-11, 1996.

[41] B. Grenfell and J. harwood, “(Meta)population dynamics of infectious diseases,” *Trends in Ecology and Evolution*, vol. 12, no. 10, October 1997.

[42] B. T. Grenfell and B. M. Bolker, “Cities and villages: infection hierarchies in a measles metapopulation,” *Ecology Letters*, vol. 1, pp. 63-70, 1998.

[43] N. M. Ferguson, M. J. Keeling, W. J. Edmunds, R. Gani, B. T. Grenfell, R. M. Anderson and S. Leach, “Planning for smallpox outbreaks,” *Nature*, vol. 425, October 2003.

[44] L. A. Rvachev, “A mathematical model for the global spread of influenza,” *Mathematical Biosciences*, vol. 75, pp. 3-22, July 1985.

[45] R. F. Grais, J. H. Ellis and G. E. Glass, “Assessing the impact of airline travel on the geographic spread of pandemic influenza,” *European Journal of Epidemiology*, vol. 18, pp. 1065-1072, 2003.

[46] V. Colizza, A. Barret, M. Barthélemy and A. Vespignani, “The role of the airline transportation network in the prediction and predictability of global epidemics, ” *Proceedings of the National Academy of Sciences*, vol. 103, no. 7, pp. 2015-2020, February 14, 2006.

[47] L. Ufnagel, D. Brockmann and T. Geisel, “Forecast and control of epidemics in a globalized world,” *Proceedings of the National Academy of Sciences*, vol. 101, no. 42, October 19, 2004.

[48] V. Colizza, R. Pastor-Satorras and A. Vespignani, “Reaction-diffusion processes and metapopulation models in heterogeneous networks,” *Nature Physics*, vol. 3, April, 2007.

[49] O. Diekmann, J. A. P. Heesterbeek, and J. A. J. Metz, “On the definition and the computation of the basic reproduction ratio R_0 in models for infectious diseases in heterogeneous populations,” *Journal of Mathematical Biology*, vol. 28, pp. 365-382, 1990.

[50] O. Diekmann, J. A. P. Heesterbeck, *Mathematical Epidemiology of Infectious Diseases: Model Building, Analysis and Interpretation*, S. Levin, Ed., Wiley Series in Mathematical and Computational Biology, John Wiley & Sons, New York, 2000.

[51] T. W. Ng, G. Turinici, and A. Danchin, “A double epidemic model for the SARS propagation,” *BMC Infectious Diseases*, 3:19, September 10, 2003.

[52] B. Bolker and B. Granfell, “Space, persistence and dynamics of measles epidemics,” *Philosophical Transactions of the Royal Society of London. B*, vol. 348, pp. 309-320, 1995.

[53] D. J. D. Earn, P. Rohani and B. T. Grenfell, “Persistence, chaos and synchrony in ecology and epidemiology,” *Proceedings of the Royal Society of London. Series B*, vol. 265, pp. 7-10, 1998.

[54] P. Rohani, D. J. D. Earn and B. T Grenfell, “Opposite pattens of synchrony in sympatric disease metapopulations,” *Science*, vol. 286, October 29, 1999.

[55] M. J. Keeling, “Metapopulation moments: coupling, stochasticity and persistence,” *Journal of Animal Ecology*, vol. 69, pp. 725-736, 2000.

[56] N. G. Becker, K. Glass, L. Zhengfeng, and G. K. Aldis, *Mathematical Biosciences*, vol. 193, pp. 205-221, 2005.

[57] C. Beauchemin, J. Samuel, and J. Tuszynski, “A simple cellular automaton model for influenza A viral infections,” *Journal of Theoretical Biology*, vol. 232, pp. 223-234, 2005.

[58] C-Y. Huang et al, “Simulating SARS: small-world epidemiological modeling and public health policy assessments,” *Journal of Artificial Societies and Social Simulation*, vol. 7, no. 4, 2004.

[59] J. O. Kephart, S. R. White, “Directed-graph epidemiological models of computer viruses”, in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 343-359, Oakland, CA, 1991.

[60] Z. Dezsö and A.-L. Barabási, “Halting viruses in scale-free networks,” *Physical Review E*, 65:055103(R), 2002.

[61] R. Pastor-Satorras and A. Vespignani, “Epidemic dynamics in finite size scale-free networks, *Physical Review E*, 65:035108(R), 2002.

[62] R. Pastor-Satorras and A. Vespignani, “Epidemic spreading in scale-free networks,” *Physical Review Letters*, 86(14) pp. 3200-3203, 2001.

[63] M. Boguñá and R. Pastor-Satorras, “Epidemic spreading in correlated complex networks,” *Physical Review E*, 66(4):047104, 2002.

[64] A. L. Lloyd and R M. May, “How viruses spread among computers and people,” *Sicence*, vol. 292, no. 5520, pp. 1316-1317, May 2001.

[65] Y. Wang, D. Chakrabarti, C. Wang, C. Faloutsos, “Epidemic spreading in real networks: an eigenvalue viewpoint,” in *Proceedings of 22nd International Symposium on Reliable Distributed Systems*, Florence, Italy, October, 2003.

[66] D. Siljak, *Decentralized Control of Complex Systems,* Academic Press: Boston, 1994.

[67] J. P. Corfmat and A. S. Morse, “Decentralized control of linear multivariable systems,” *Automatica*, vol. 12, pp. 479-495, 1976.

[68] S. Roy, K. Herlugson, and A. Saberi, “A control-theoretic perspective on distributed discrete-valued decision-making in networks of sensing agents,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 945-957, pp. 55-62, 2001. 2006.

[69] S. Roy and A. Saberi,“On structural properties of the Lyapunov matrix equation for optimal diagonal Lyapunov functions,” submitted as a technical correspondence to the *IEEE Transactions on Automatic Control.*

[70] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press: New York, 1974.

[71] D. P. Bertsekas, *Nonlinear Programming (2nd ed.)*, Athena Scientific: Belmont MA, 1995.

[72] R. G. Gallager, *Discrete Stochastic Processes*, Kluwer Academic Publishers, Massachusetts, 1996.

[73] F. R. K. Chung, *Spectral Graph Theory*, American Mathematical Society Press: Providence, RI, 1997.

[74] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Classics in Applied Mathematics Series, SIAM, Philadelphia, 1994.

[75] S. Eubank, H. Guclu, V. S. A. Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkal and N. Wang, "Modelling disease outbreaks in realistic urban social networks," *Nature*, vol. 429, pp. 180-184, 2004.

[76] A. Ganesh, L. Massoulié and D. Towsley, "The effect of network topology on the spread of epidemics," in *Proceedings of IEEE Infocom*, Miami, FL, March 13-17, 2005.

[77] V. Blondel and J. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM Journal of Control and Optimization*, vol. 35, no. 6, pp. 2118-2127, 1997.

[78] D. Hershkowitz, "Recent directions in matrix stability," *Linear Algebra and its Applications*, Vol. 171, pp. 161-186, Jul., 1992.

[79] M. E. Fisher and A. T. Fuller,"On the Stability of matrices and the convergence of linear iterative processes," *Proceedings of the Cambridge Philosophical Society*, vol. 45, pp. 417-425, 1958.

[80] S. Boyd, "Convex optimization of graph laplacian eigenvalues," in *Proceedings of the International Congress of Mathematicians*, vol. 3, pp. 1311-1319, 2006.

[81] A. S. Lewis and M. L. Overton, "Eigenvalue optimization," *Acta Numerica*, vol. 5, pp. 149-190, 1996.

[82] S. Roy, B. Sridhar, and G. C. Verghese,"An Aggregate Dynamic Stochastic Model for an Air Traffic System," in *Proceedings of the 5th Eurocontrol/Federal Aviation Agency Air Traffic Management Research and Development Seminar*, Budapest, Hungary, June 2003.

[83] S. S. Allan, J. A. Beesley, J. E. Evans and S.G. Gaddy, "Analysis of Delay Causality at Newark International Airport," in *Proceedings of the 4th USA/Europe Air Traffic Management R&D Seminar*, Santa Fe, New Mexico, December 2001.

[84] D. Long, E. Wingrove, D. Lee, J. Gribko, R. Hemm and P. Kostiuk, "A Method for Evaluating Air Carrier Operational Strategies and Forecasting Air Traffic with Flight Delay," *Final Report for National Aeronautics and Space Administration Contract NAS2-14361*, 1999.

[85] E. R. Mueller and G. B.Chatterji, "Analysis of aircraft arrival and departure delay characteristics," *Proceedings of the Aircraft Technology Integration and Operations Technical Forum*, Los Angeles, CA, October 2002.

[86] D. C. Moreau and S. Roy,"A Stochastic Characterization of En Route Traffic Flow Management Strategies," in *Proceedings of the 2005 AIAA Guidance, Navigation, and Control Conference*, San Francisco, CA.

[87] J. W. Pepper, K. R. Mills and L. A. Wojoik, "Predictability and Uncertainty in Air Traffic Flow Management," 5*th USA/Europe Air Traffic Management R&D Seminar*, Budapest, June 2003.

[88] D. J. Brudnicki and A. L. McFarland, "User Request Evaluation Tool (URET) Conflict Probe Performance and Benefits Assessment," *Technical Report for the U. S. Government under Contract Number DTFA01-93-C-00001*, The MITRE Corporation,1997.

[89] T. Hoang, T. Farley and T. Davis, "The Multi-Center TMA System Architecture and Its Impact on Inter-facility Collaboration," in *Proceedings of the AIAA Aircraft Technology, Integration and Operations (ATIO) Conference*, Los Angeles, CA, October 2002.

[90] K. Roy, and C. J. Tomlin, "Enroute Airspace Control and Controller Workload Analysis using a Novel Slot-based Sector Model," in *Proceedings of the 2006 American Control Conference, Minneapolis*, June 2006.

[91] S. Landry, T. Farley, J. Foster, S. Green, T. Hoang, G. L. Wong, "Distributed Scheduling Architecture for Multi-Center Time-based Metering," in *Proceedings of the AIAA Aircraft Technology, Integration and Operations (ATIO) Conference*, Denver, CO, November 2003.

[92] T. Farley, J. D. Foster, T. Hoang and K. K. Lee,"A Time-based Approach to Metering Arrival Traffic to Philadelphia," in *Proceedings of the First FIAA Aircraft Technology, Integration, and Operations Forum*, Los Angeles, California, October 2001.

[93] A. M. Bayen, R. L. Raffard, and C. J. Tomlin, "Adjoint-Based Constrained Control of Eulerian Transportation Networks:Application to Air Traffic Control," *Proceedings of the 2004 American Control Conference*, Boston, Massachusetts, June 2004.

[94] A. M. Bayen, R. L. Raffard and C. J. Tomlin,"Eulerian Network Model of Air Traffic Flow in Congested Areas," *Proceedings of the 2004 American Control Conference*, Boston, Massachusetts, June 2004.

[95] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, 3rd ed., Wiley: New York, 1998.

[96] A. M. Bayen, C. J. Tomlin, Y. Yez and J. Zhang, "An Approximation Algorithm for Scheduling Aircraft with Holding Time," In *Proceedings of the 43th Aircraft Technology, Integration, and Operations Forum*, Los Angeles, California, October 2001.

[97] D. Bertsimas, and S. Stock, *The Air Traffic Flow Management Problem with Enroute Capacities*, Working Paper, Sloan School of Management, Massachusetts Institute of Technology, August 1994.

[98] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 2, pp. 110-120, Jun. 2001.

[99] R. O. Hoffman and M. O. Ball, "The rate control index for traffic flow," *IEEE Transactions of Intelligent Transportation Systems*, vol. 2, no. 2,

[100] B. Sridhar, T. Soni, K. S. Sheth, and G.B. Chatterji, "An aggregate flow model for air traffic management," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Providence, RI, August 2004.

[101] P. van Tulder, M. Berge, B. Repetto, A. Haraldsdottir, and D. Moerdyk, "Airline schedule recovery in collaborative flow management and weather forecast uncertainty," in *Proceedings of the 2004 Digital Avionics Systems Conference*, Salt Lake City, UT, October, 2004.

[102] M. E. Berge, C. A. Hopperstad, and A. Haraldsdottir, "Airline schedule recovery in collaborative flow management with airport and airspace capacity constraint," in *Proceedings of the 5th US/Europe Air Traffic Management Research and Development Seminar*, Budapest, June 2003.

[103] H. Idris, J. P. Clarke, R. Bhuva, and L. Kang, "Queueing model for taxi-out time estimation," *Traffic Control Quarterly*, vol. 10, no. 1, pp. 1-22, 2001.

[104] H. Chen and Y. Zhao, "A new queueing model for aircraft landing process," in *Proceedings of the AIAA GNC, AFM, and MST Conference and Exhibit*, New Orleans, LA, August 1997.

[105] P. B. Mirchandani and N. Zou, "Queueing models for analysis of traffic adaptive signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no.'1, Mar. 2007.

[106] C. Wu and Y. Liu, "Queuing network modeling of driver workload and performance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, Sep. 2007.

[107] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 2002.

[108] B. Sridhar and P. K. Menon, "Comparison of linear dynamic models for air traffic management," in *Proceedings of the 16th International Federation of Automatic Control (IFAC) World Congress*, no. 02187, Prague, July 2005.

[109] B. Sridhar and S. Swei, "Computation of aggregate delay using center-based weather impacted traffic index," at *National Airspace System Performance Workshop*, Asilomar Conference Grounds, Pacific Grove, CA, Sep. 2007.

[110] B. Sridhar, "Modeling, optimization, and software in air traffic management," *Presentation at Washington State University*, Feb. 2007.

[111] J. Krozel, R. Jakobovitz, and S. Penny, "An algorithmic approach for airspace flow programs," *Air Traffic Control Quarterly*, vol. 14, no. 3, 2006.

[112] S. A. Martinez, G. B. Chatterji, D. Sun, A. M. Bayen, "A weighted-graph approach for dynamic airspace configuration, " in *Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, 20 - 23 August 2007.

[113] K. Bilimoria, B. Sridhar, and G. Chatterji, "Effects of conflict resolution maneuvers and traffic density on free flight," *AIAA Guidance, Navigation, and Control Conference*, San Diego, CA, Jul. 1996.

[114] A. Klein, R. Jehlen, and D. Liang, "Weather index with queueing component for National Airspace System performance assessment," 7th FAA/Eurocontrol ATM Seminar, Barcelona, Spain, Jul. 2007.

[115] J. Krozel, "Capacity estimation for level flight with convective weather constraints," submitted to *Air Traffic Quarterly*.

[116] R. Hoffman, "Probabilistic scenario-based event planning for traffic flow management," in *Proceedings of AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, South Carolina, 20 - 23 August 2007.

[117] J. Krozel, C. Lee, and J. S. B. Mitchell, "Turn-constrained route planning for avoiding hazardous weather," *Air Traffic Control Quarterly*, vol. 14, no. 2, pp. 159-182, 2006.

[118] P. Kopardekar, K. Bilimoria, and B. Sridhar, "Initial concepts for dynamic airspace configuration," in *Proceedings of 7th AIAA Aviation Technology, Integration and Operations Conference*, Belfast, Northern Ireland, 18 - 20 September 2007.

[119] R. Ehrmanntraut and S. McMillan, "Airspace design process for dynamic sectorisation, " 26th DASC, Dallas, Texas, 2007.

[120] J. A. Hadley and R. L. Sollenberger, "Dynamic resectorization of airspace boundaries between adjacent air route traffic control centers, " *FAA techinical report*, 2001.

[121] A. Borbely and D. Schneider, *Secrets of Sleep*, Basic Books: New York, 1986.

[122] J. M. Krueger and F. Obal Jr., "A neuronal group theory of sleep function," *Journal of Sleep Research*, vol. 2, pp. 63-69, 1993.

[123] S. H. Strogatz, *The mathematical structure of the human sleep-wake cycle*, Springer-Verlag: New York, 1986.

[124] A. A. Borbely, "A two process model of sleep regulation," *Human Neurobiol.*, vol. 1, pp. 195-204, 1982.

[125] P. Achermann and H. Kunz, "Modeling circadian rhythm generation in the suprachiasmatic nucleus with locally coupled self-sustained oscillators," *Journal of Biological Rhythms*, vol. 14, pp. 460-468, 1999.

[126] J. Krueger and F. Obal Jr., "Sleep function," *Frontiers in Bioscience*, vol. 8, pp. 511-519, 2003.

[127] J. Eggert and J. L. van Hemmen, "Modeling neuronal assemblies: theory and implementation," *Neural Computation*, vol. 13, pp. 1923-1974, 2001.

[128] V. S. Manorjan, I. Rajapakse, and J. M. Krueger, "Oscillations in a neuronal assembly—a phenomenological model," *International Journal of Computational and Applied Mathematics*, vol. 1, no. 1, 2005.

[129] D. L. Wang and D. Terman, "Global competition and local cooperation in a network of neural oscillators," *Physica D*, vol. 81, pp. 148-176, 1995.

[130] D. Terman, J. E. Rubin, A. C. Yew, "Activity patterns in a model for subthalamopallidal network of the basal ganglia," *The Journal of Neuroscience*, vol. 22, no. 7, pp. 2963-2976, 2002.

[131] A. Kane, *Activity Propagation in Two-Dimensional Neuronal Networks*, Ph.D. Dissertation, Graduate School in Mathematics, The Ohio State University, 2005.

[132] M. Bazhenov, I. Timofeev, M. Steriade, and T. Sejnowski, "Spiking-bursting activity in the thalamic reticular nucleus initiates sequences of spindle oscillations in thalamic networks, *Journal of Neurophysiology*, vol. 84, pp. 1076-1087, 2000.

[133] S. L. Hill and G. Tononi, "Modeling sleep and wakefulness in the thalamocortical system," *Journal of Neurophysiology*, vol. 4, no. 31, pp. 6862-6870, Aug. 2004.

[134] M. Massimini, F. Ferrarelli, R. Huber, S. K. Esser, H. Singh, and G. Tononi, "Breakdown of cortical effective connectivity during sleep," *Science*, vol. 309, pp. 2228-2232, 2005.

[135] M. T. Wilson, J. W. Sleigh, M. L. Steyn-Ross, and D. A. Steyn-Ross, "Cerebral cortex: using higher-order statistics in a mean-field model," presented at the Australian Institute of Physics Congress, Brisbane, Dec. 2006.

[136] J. J. Wright, P. A. Robinson, C. J. Rennie, and E. Gordon, "Toward an integrated continuum model of cerebral dynamics: the cerebral rhythms, synchronous oscillation and cortical stability," *Journal of Biological and Information Processing Sciences*, vol. 63, no. 1-3, pp. 71-88, 2001.

[137] A. Borbely and P. Achermann, "Sleep homeostasis and models of sleep regulation," *Journal of Biological Rhythms*, vol. 14, no. 6, pp. 557-568, 1999.

[138] J. Chow, *Time-scale modeling of dynamic networks with application to power systems*, Springer-Verlag: New York, 1983.

[139] P. V. Kokotovic, J. O'Reilly, and H. K. Khalil, *Singular Perturbation Methods in Control: Analysis and Design*, SIAM Academic Publishing: New York, 1986.

[140] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal of Applied Mathematics*, vol. 50, pp. 1645-1662, 1990.

[141] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71-82, 2007.

[142] R. M. Murray, K. J. Astrom, S. P. Boyd, R. W. Brockett, and G. Stein, "Future directions in control in an information-rich world," *IEEE Control Systems Magazine*, pp. 20-33, April 2003.

[143] G. Lafferriere, A. Williams, J. Caughman, and J. J. P. Veerman, "Decentralized control of vehicle formations," *Systems and Control Letters*, vol. 54, no. 9, pp. 899-910, Sep. 2005.

[144] J. J. Hopfield and A. V. M. Herz, "Rapid local synchronization of action-potentials – toward computation with coupled integrate-and-fire neurons," *Proceedings of the National Academy of Science*, vol. 92, no. 15, pp. 6655-6662, Jul. 18, 1995.

[145] S. Bottani, "Pulse-coupled relaxation oscillators – from biological synchronization to self-organized criticality," *Physical Review Letters*, vol. 74, no. 21, pp. 4189-4192, May 1995.

[146] A. Diaz-Guilera, C. J. Perez, and A. Arenas, "Mechanisms of synchronization and pattern formation in a lattice of pulse-coupled oscillators," *Physical Review E*, vol. 57, no. 4, pp. 3820-3828, Apr. 1998.

[147] S. H. Strogatz, "Coupled Oscillators and Biological Synchronization," *Scientific American*, vol. 269, no. 6, pp. 102-109, Dec. 1993.

[148] G. A. Pratt and J. Nguyen, "Distributed Synchronous Clocking," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 3, pp. 314-328, Mar. 1995.

[149] J. Slade, *Synchronization of Multiple Rotating Systems*, Master's Thesis, School of Electrical Engineering and Computer Science, Washington State University, July 2007.

[150] S. Strogatz, "From Kuramoto to Crawford: exploring the onset of synchronization in a population of coupled nonlinear oscillators," *Physica D*, vol. 143, pp. 1-20, 2000.

[151] A. Jadbabaie, N. Motee, and M. Barahona, "On the stability of the Kuramoto model of coupled nonlinear oscillators," *Proceedings of the 2004 American Control Conference*, Boston, MA, 2004.

[152] O. Simeone and U. Spagnolini, "Distributed timing synchronization in wireless sensor networks with coupled discrete-time oscillators," *EURASIP Journal on Wireless Communications and Networking*, Art. no. 57054, 2007.

[153] A. S. Hu and S. D. Servetto, "On the scalability of cooperative time synchronization in pulse-connected networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2725-2748, Jun. 2006.

[154] J. M. Krueger, D. M. Rector, S. Roy, G. Belenky, and J. Panksepp, "Brain organization of sleep," submitted to *Nature Neurosciences Reviews*.

[155] L. Ljung, *System Identification: Theory for the User*, Prentice Hall: Englewood Cliffs NJ, January, 1999.

[156] C. Asavathiratham, S. Roy, B. C. Lesieutre, and G. C. Verghese, "The influence model," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 52-64, 2001.

[157] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440-442, 1998.

[158] D. J. Watts, *Six Degrees: The Science of a Connected Age*, Norton Press: New York, 2003.

[159] D. M. Rector, I. A. Topchiy, K. M. Carter, and M. J. Rojas, "Local functional state differences between rat cortical columns," *Brain Research*, vol. 1047, pp. 45-55, 2005.

[160] F. Obal Jr. and J. M. Krueger, "Humoral mechanisms of sleep," in *The Physiological Nature of Sleep* (P. L. Parmeggiani and R. Velluti, eds.), Imperial College Press: London, pp. 23-44, 2005.

[161] Z. Wen, S. Roy, and A. Saberi, “On the disturbance response and external stability of a saturating static-feedback controlled double integrator,” to appear in *Automatica*.

[162] R. Albert and A.-L. barabasi, “Topology of evolving networks: local events and universality,” *Physical Review Letters*, vol. 85, no. 24, 2000.

[163] P. M. Anderson and A. A. Fouad, *Power System Control and Stability*, Hoboken, NJ: IEEE Press Power Engineering Series, 2^{nd} edition, 2003.

[164] P. Bak, C. Tang, and K. Wiesenfeld, “Self-organized criticality: an explanation of 1/f noise,” *Physical Review Letters*, vol. 59, pp. 381-384, July, 1987.

[165] J. M. Carlson and J. C. Doyle, “Highly Optimized Tolerance: A Mechanism for Power Laws in Designed Systems,” *Physical Review E*, vol. 60, no. 2, 1999.

[166] B. A. Carreras, V. E. Lynch, I. Dobson, and D. E. Newman, “Dynamics, Criticality and Self-organization in a Model for Blackouts in Power Transmission Systems,” in *Proceedings of the Hawaii International Conference on Systems Sciences*, Hawaii, Jan. 2002.

[167] M. Costa, J. Crowcroft, M. Castro, and A. Rowstron, *Can we contain internet worms?* Microsoft Research Technical Report MSR-TR-2004-83.

[168] P. Gevros and G. Crowcroft, “Distributed resource management with heterogeneous linear controls,” *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 45, no. 6, pp. 835-858, 2004.

[169] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Society for Industrial and Applied Mathematics (SIAM) Press: Phiadelphia, 1997.

[170] R. Hekmat, “Ad-hoc networks: fundamental properties and network topologies,” *Springer*, Cambridge MA, 2006.

[171] F. Kelly, “Fairness and stability of end-to-end congestion control,” *European Journal of Control*, vol. 9, pp. 159-176, 2003.

[172] R. Levins, “Some Demographic and Genetic Consequences of Environmental Heterogeneity for Biological Control,” *Bulletin of the Entomological Society of America*, vol. 15, pp. 237-240.

[173] M. A. de Menezes and A.-L. Barabasi, “Fluctuations in network dynamics,” *Physics Review Letters*, vol. 92, no. 2, 2004.

[174] M. Newman, D. Watts, and A.-L. Barabasi, *The Structure and Dynamics of Networks*, Princeton University Press: New Jersey, 2006.

[175] L. J. Perez-Arriaga, G. C. Verghese, F. L. Pagola, J. L. Sancha, and F. C. Schweppe, “Developments in selective modal analysis of small-signal stability in electric power systems,” *Automatica*, vol. 26, pp. 215-231, 1990.

[176] H. Wang, H. Xie, L. Qiu, Y. Yang, Z. Zhang, A. Greenberg, “COPE: Traffic Engineering in Dynamic Networks”, in *Proceedings of SIGCOMM 2006*, Pisa, Italy, Sep. 2006.

[177] D. J. Watts, *Small Worlds: The Dynamics of Networks Between Order and Randomness*, Princeton University Press: New Jersey, 2003.

[178] S. Boyd, P. Diaconis, J. Sun, and L. Xiao, “Fastest mixing markov chain on a path,” *The American Mathematical Monthly*, vol. 113, no. 1, pp. 70-74, January 2006.

[179] S. Boyd, P. Diaconis, and L. Xiao, “Fastest mixing markov chain on a graph,” *SIAM Review*, vol. 46, no. 4, pp. 667-689, December 2004.

[180] J. Sun, S. Boyd, L. Xiao, and P. Diaconis, “The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem,” *SIAM Review*, vol. 484, pp. 681-699, November 2006.

[181] A. Ghosh, S. Boyd, and A. Saberi, “Minimizing effective resistance of a graph,” *SIAM Review, problems and techniques section*, vol. 50, no. 1, pp. 37-66, February 2008.

[182] A. Ghosh and S. Boyd, “Upper bounds on algebraic connectivity via convex optimization,” *Linear Algebra and its Applications*, vol. 418, pp. 693-707, October 2006.

[183] A. Ghosh and S. Boyd, “Growing well-connected graphs,” in *Procedings of 45th IEEE Conference on Decision and Control*, pp. 6605-6611, December 2006.

[184] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems and Control Letters*, vol. 53, pp. 65-78, 2004. Shorter version appeared in *Proceedings of IEEE Conference on Decision and Control*, pp. 4997-5002, Hawaii, December 2003.

[185] M. Fiedler, “Absolute algebraic connectivity of trees,” *Linear and Multilinear Algebra*, vol. 423, no. 1, pp. 53-73, May 2007.

[186] N. M. M. de Abreu, “Old and new results on algebraic connectivity of graphs,” *Linear Algebra and Its Applications*, vol. 423, no. 1, pp. 53-73, May 2007.

[187] S. Kirkland and S. Pati, “On vertex connectivity and absolute algebraic connectivity for graphs,” *Linear and Multilinear Algebra*, vol. 50, no. 3, pp. 253-284, January 2002.

[188] R. B. Ellis III, *Chip-firing games with dirichlet eigenvalues and discrete green’s functions*, PhD Thesisin Mathematics, University OF California, San Diego, California, 2002.

[189] F. Göring, C. Helmberg, and M. Wappler, “Embedded in the Shadow of the Separator,” submitted to *SIAM Journal on Optimization*, September 2, 2005.

[190] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press: Oxford, 1965.

[191] J. Moro, J. V. Burke, and M. L. Overton, “On the Lidskii-Vishik-Lyusternik perturbation theory for eigenvalues of matrices with arbitrary Jordan structure,” submitted to the *SIAM Journal on Matrix Analysis and its Applications*.

[192] J. Kleinberg, “The small-world phenomenon: an algorithmic perspective,” in *Proceedings of the* 32nd *ACM Symposium on Theory of Computing*, Portland, OR, May 2000.

[193] B. Bollobas and F. R. K. Chung, “The diameter of a cycle plus a random matching,” *SIAM Journal on Discrete Mathematics*, vol. 1, pp. 328-333, 1988.

[194] I. W. Horowitz, *Synthesis of feedback systems*, Academic Press: New York, 1963.

[195] A. Saberi and P. Sannuti, "Time-scale structure assignment in linear multivariable systems with high-gain feedback," *International Journal of Control*, vol. 49, no. 6, 1989.

[196] A. Saberi, B. M. Chen, and P. Sannuti, *Loop Transfer Recovery: Analysis and Design*, Springer-Verlag, 1993.

[197] H. K. Ozcetin, A. Saberi, and P. Sannuti, "Almost Disturbance Decoupling Problem with Internal Stability Via State or Measurement Feedback – Singular Perturbation Approach," *International Journal of Control*, vol. 55, no. 4, pp. 901-944, 1992.

[198] Z. Lin and A. Saberi, "Semi-global exponential stabilization on linear systems subject to "input saturation" via linear feedbacks," *Systems and Control Letters*, vol. 21, pp. 225-239, 1993.

[199] G. N. Ramaswamy, G. C. Verghese, L. Rouco, C. Vialas, and C. DeMarco, "Synchrony, aggregation, and multi-area eigenanalysis," *IEEE Transactions on Power Systems*, vol. 10, no. 4, pp. 1986-1993, Nov. 1995.

[200] C. L. DeMarco and J. Wassner, "A generalized eigenvalue perturbation approach to coherency," *Proceedings of the 4th IEEE Conference on Control Applications*, pp. 611-617, 1995.

[201] P. Sridhar, T. Kahveci, and S. Ranka, "An iterative algorithm for metabolic network-based drug target idenficaition," in *Proceedings of the 2007 Pacific Symposium on Biocomputing*, Maui HI, Jan. 2007.

[202] P. Csermely, V. *Á*goston, and S*á*ndor Pongor, "The efficiency of multiple-target drugs: the network approach might help drug design," TRENDS in Pharmacological Science, Vol. 26, No. 4, April 2005.

[203] D. Colzolari, G. Paternostro, P. L. Harrington, C. Piermarpcchi, P. M. Duxbury, "Selective control of the poptosis signaling network in heterogeneous cell populations," *PLoS ONE*, Issue 6 June 2007.

[204] D. Cheverez-Gonzalez and C. L. DeMarco, "Laplacian structure in power network constraints and inherent zonal prices regions," in *Proceedings of the 2006 North American Power Symposium*, Carbondale IL, Sep. 2006.

[205] P. Sannuti and A. Saberi, "Special coordinate basis for multivariable linear system-finite and infinite zero structure, squaring down and decoupling," *International Journal of Control*, vol. 45, No. 5, pp. 1655-1704, 1987.

[206] A. Saberi, P. V. Kokotovic, and H. J. Sussmann, "Global stabilization of partially linear composite systems," *SIAM Journal on Control and Optimization*, vol. 28, no. 6, pp. 1491-1503, November 1990.

[207] J. Baillieul and P. J. Antsaklis, "Control and communication challenges in networked real-time systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 9-28, Jan. 2007.

[208] S. Roy, A. Saberi, and A. Stoorvogel, "Toward a control theory for networks," *International Journal of Robust and Nonlinear Control*, vol. 17, no. 10-11, p. 897, July 2007.

[209] A. Berman, M. Neuman, and R. J. Stern, *Nonnegative Matrices in Dynamic Systems*, Pure and Applied Mathematics Series, Wiley 1989.

[210] D. A. Spielman and S.-H. Teng, "Spectral partitioning works: planar graphs and element meshes," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, Burlington, VT, Oct. 1996

[211] A. Hagberg and d. A. Schult, "Rewiring networks for synchronization," *Chaos*, p. 037105-1, vol. 18, 2008.

[212] T. Biyikoglu, J. Leydold, and P. Stadler, *Laplacian Eigenvectors of Graphs:Perron-Frobenius and Faber-Krahn Type Theorems*, Springer, 2007.

[213] B. Ayazifar, *Graph spectra and modal dynamics of oscillatory networks*, Ph.D. thesis, Massachusetts Institute of Technology, 2002.

[214] R. Merris, "Laplacian graph eigenvectors," *Linear Algebra and its Applications*, vol. 278, pp. 221-236, 1998.

[215] Y. Wan, S. Roy, A. Saberi, and A. Stoorvogel, "A multiple-derivative and multiple-delay paradigm for decentralized controller design: uniform rank systems," submitted to *Automatica*.

[216] C. L. DeMarco, "Control structures for competitive market-driven power systems," *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando FL, Dec. 2001.

[217] B. D. O. Anderson and J. B. Moore, "Time-varying feedback laws for decentralized control," *IEEE Transactions on Automatic Control*, Vol. 26, No. 5, pp. 1133-1139, October 1981.

[218] J. P. Corfmat and A. S. Morse, "Stabilization with decentralized feedback control," *IEEE Transactions on Automatic Control*, Vol. 21, No. 1, pp. 679-682, Feb. 1976.

[219] J. K. Hale and S. M. V. Lunel, *Introduction to Functional Difference Equations*, Epring-Verlag, 1993.

[220] S.-I. Niculescu and W. Michiels, "Stabilizing a chain of integrators using multiple delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 5, pp. 802-807, May 2004.

[221] A. Saberi, A. Stoorvogel, and P. Sannuti, *Output Regulation and Control Problems with Regulation Constraints*, Springer-Verlag, 1999.

[222] A. Stoorvogel, A. Saberi, C. Deliu, and P. Sannuti, "Decentralized stabilization of time-invariant systems subject to actuator saturation," *Advanced Strategies in Control Systems with Input and Output Constraints*, LNCIS series (S. Tarbouriech et al eds.), Springer-Verlag, June 2006.

[223] E. Kreindler and A. Jameson, "Optimality of linear control systems," *IEEE Transactions on Automatic Control*, vol. 17, no. 3, pp. 349-351, June 1972.

[224] W. Michiels and T. Vyhlidal, “An eigenvalue based approach for stabilization of linear time-delay systems of neutral type,” *Automatica*, vol. 41, pp. 991-998, 2005.

[225] Z. Lin and H. Fang, “On asymptotic stabilizability of linear systems with delayed input,” *IEEE Transaction n Automatic Control*, vol. 52, no. 6, June 2007.

[226] C. W. Wu and L. Chua, “Application of kronecker products to the analysis of systems with uniform linear coupling, ” *IEEE Transactions on Circuits and Systems I*, vol. 42, no. 10, pp. 775-779, October 1995.

[227] C. W. Wu and L. Chua, “Application of graph theory to the synchronization in an array of coupled nonlinear oscillators, ” *IEEE Transactions on Circuits and Systems I*, vol. 42, no. 8, pp. 494-497, October 1995.

[228] J. A. Fax and R. M. Murray, “Information flow and cooperative control of vehicle formations,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465-1476, September 2004.

[229] A. Pogromsky and H. Nijmeijer, “Cooperative oscillatory behavior of mutually coupled dynamical systems,” *IEEE Transactions on Circuits and Systems*, Part I, vol. 48, no. 2, Feb. 2001.

[230] S. Roy, A. Saberi, and K. Herlugson, “Control-theoretic perspective on the design of distributed agreement protocols,” *International Journal on Robust and Nonlinear Systems*, Special Issue on Communicating-Agent Networks, published online in November 2006 (1.108, 6).

[231] Z. Duan, J. Wang, G. Chen, and L. Huang, “Stability analysis and decentralized control of a class of complex dynamical networks, ” *Automatica*, vol. 44, pp. 1028-1035, 2008.

[232] W. Ren, “On consensus algorithms for double-integrator dynamics,” *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1503-1509, Jul. 2008.

[233] Z. Lin and A. Saberi, “A semi-global low-and-high gain design technique for linear systems with input saturation – stabilization and disturbance rejection,” *International Journal of Robust and Nonlinear Control*, vol. 5, pp. 381-398, 1995.

[234] J.K. Hale and S.M. Verduyn Lunel, *Introduction to functional differential equations*, vol. 99 of Applied Mathematical Sciences, Springer Verlag, New York, 1993.

[235] V. L. Kharitonov, S.-I. Niculescu, J. Moreno, and W. Michiels, “Static output feedback stabilization: necessary conditions for multiple-delay controllers,” *IEEE Transactions on Automatic Control*, vol. 50, no. 1, pp. 82-86, Jan. 2005.

[236] A. Ilchmann and C.J. Sangwin, “Output feedback stabilization of minimum phase systems by delays”, Syst. & Contr. Letters, 52(3-4), pp. 233–245, 2004.

[237] X. Liu, Z. Lin, and B. M. Chen, “Symbolic realization of asymptotic time-scale and eigenstructure assignment design method in multivariable control,” *International Journal of Control*, vol. 79, no. 11, pp. 1471-1484, Nov. 2006.

[238] A. Saberi and P. Sannuti, “Squaring down by static and dynamic compensators,” *IEEE Transactions on Automatic Control*, vol 33, no 4, 1988.

[239] V. I. Syrmos, C. T. Abdallah, P. Dorato, and K. Grigoriadis, "Static output feedback–a survey," *Automatica*, vo. 33, no. 2, pp. 125-137, 1997.

[240] D. C. Youla, D. D. Bongiorno Jr., and C. N. Liu, "Single loop feedback stabilization of linear multivariable plants," *Automatica*, vol. 10, pp. 159-173, 1974.

[241] W. Michiels and D. Roose, *Global stabilization of multiple integrators with time-delay and input constraints*, Technical Report TW 325, Department of Computer Science, K.U.Lueven, May 2001.

[242] A. Papoulis, "Limits on bandlimited signals," *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1677-1686, Oct. 1967.

[243] J.B. Conway, *Functions of one complex variable*, vol. 11 of Graduate texts in mathematics, Springer Verlag, New York, 2nd Ed., 1995.

[244] V.L. Kharitonov, S.-I. Niculescu, J. Moreno, and W. Michiels, "Static output feedback stabilization: necessary conditions for multiple delay controllers", IEEE Trans. Aut. Contr., 50(1), pp. 82–86, 2005.

[245] H. Kokame, K. Hirata, K. Konishi, and T. Mori, "Difference feedback can stabilize uncertain steady states", IEEE Trans. Aut. Contr., 46(12), pp. 1908–1913, 2001.

[246] H. Kokame and T. Mori, "Stability preserving transition from derivative feedback to its difference counterparts", in Proceedings of the 15th IFAC World Congress, Barcelona, Spain, 2002.

[247] D.A. O'Connor and T.J. Tarn, "On stabilization by state feedback for neutral differential difference equations", IEEE Trans. Aut. Contr., 28(5), pp. 615–619, 1983.

[248] S. Bochner, "Beiträge zur theorie der fastperiodischen funktionen. I. Teil. Funktionen einer Variablen", Mathematische Annalen, 96(1), pp. 119–147, 1927. in German.

[249] B.Ja. Levin, *Distribution of zeros of entire functions*, vol. 5 of Translations of Mathematical Monographs, American Mathematical Society, Providence, RI, 1964. Translated from Russian.

[250] W. Michiels, *Stability and stabilization of time-delay systems*, PhD thesis, Katholieke Universiteit Leuven, Leuven, 2002. Advisors: D. Roose and R. Sepulchre.

[251] J. Douglas and M. Athans, "Multivariable poles, zeros, and pole-zero cancellations," in *The Control Handbook* (W. S. Levine, ed.), *IEEE Press*: Boca Raton, FL 1996.

[252] A. Saberi, B. M. Chen, and P. Sannuti, "Theory of LTR for non-minimum phase systems, recoverable target loops, and recover in a subspace part 1. Analysis," *International Journal of Control*, vol 53, no 5, 1067-1115, 1991.

[253] A. S. Morse, "Structural invariants of linear multivariate systems," *SIAM Journal of Control*, vol. 11, no. 3, pp. 446-465, 1973.

[254] A. Saberi, P. V. Kokotovic, and H. J. Sussmann, "Global Stabilization of Partially Linear Composite Systems," *SIAM Journal on Control and Optimization*, vol. 28, no. 6, pp. 1491-1503, 1990.

[255] S. Axler, "Down with Determinants!" *American Mathematical Monthly*, vol., pp. 139-154, 1995.

[256] A. Saberi, "Simultaneous stabilization with almost disturbance decoupling part I: uniform rank systems," in *Proceedings of 24th Conference on Decision and Control*, 1985.

[257] R. Lozano-Leal, "Robust adaptive regulation without persistent excitation," *IEEE Transactions on Automatic Control*, vol. 34, no. 2, pp. 1260-1267, Dec. 1989.

[258] D. S. Bayard and D. Boussalis, "Noncolocated structural vibration supression using zero annihilation periodic control," in *Proceedings of the 2nd IEEE Conference on Control Applications*, Vancouver, Canada, Sep. 1993.

[259] D. S. Bayard, "Extended horizon liftings for stable inversion of nonminimum phase systems," *IEEE Transactions on Automatic Control*, vol. 39. no. 6, pp. 1333-1338, June 1994.

[260] A. Ortega, G. Bartolini, and A. Ferrara, "On zero relocation in adaptive control of plants with structured uncertainties," in *Proceedings of the 20th International Conference on Industrial Electronics, Control, and Instrumentation*, Bologna, Italy, Sep. 1994.

[261] R. Ortega, "On periodic control of nonminimum phase plants," in *Proceedings of the 29th IEEE Conference on Decision and Control*, Honolulu, Hawaii, Dec. 1990.

[262] B. M. Chen, Z. Lin, and Y. Shamash, *Linear Systems Theory: A Structural Decomposition Approach*, Birkhauser: Boston, 2004.

[263] A. R. Teel, "Semi-global stabilizability of linear null controllable systems with input nonlinearities," *IEEE Transactions on Autoatic Control*, vol. 40, no. 1, January 1995.

[264] Z. Lin, A. A. Stoorvogel and A. Saberi, "Output regulation for linear systems subject to input saturation", *Automatica*, vol. 32, no. 1, pp. 29-47, January 1996.

[265] J. B. Pearson and C. Y. Ding, "Compensator design for multivariable linear system," *IEEE Transactions on Automatic Control*, vol. AC-14, no. 2, Apr. 1969.

[266] J. B. Pearson, "Compensator design for dynamic optimization," vol. 9, no. 4, pp. 473-482, Apr. 1969.

[267] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Distributed sensor networks – a review of recent research", *Journal of the Franklin Institute*, vol. 338, pp. 655-668, 2001.

[268] C. Asavathiratham, S. Roy, B. C. Lesieutre and G. C. Verghese. "The influence model," *IEEE Control Systems Magazine*, Dec. 2001.

[269] B. Chamberlain,"Graph partitioning algorithms for distributed workloads of parallel computations," *Technical Report UW-CSE-98-10-03*, University of Washington, Oct. 1998.

[270] R. B. Boppana, "Eigenvalues and graph bisection: an average case analysis," *IEEE FOCS*, pp. 280-285, 1987.

[271] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Systems Technical J.,* vol.49, pp 291-307, Feb. 1970.

[272] D. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, “Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning,” *Operations Research,* vol.37, no. 6, pp. 865–892, November-December 1989.

[273] C. Cordiero, H. Gossain, and D. P. Agrawal, “Multicast over wireless mobile ad hoc networks: present and future directions,” *IEEE Networks Magazine*, Jan./Feb. 2003.

[274] R. O. Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Transactions in Automatic Control*, vol. 49, pp. 1520-1533, Sep. 2004.

[275] B. Krisnamachari and S. S. Iyengar, “Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks,” *IEEE Transactions on Computers*, Vol. 53, No. 3, Mar. 1, 2004.

[276] S. D. Servetto and G. Barrenechia, “Constrained random walks on random graphs,” in *Proceedings of the* 1*st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, Sep. 2002.

[277] D. Kempe, J. Kleinberg, and A. Demers, “Spatial gossip and resource location protocols,” in *Proceedings of the* 33*rd ACM Symposium on Theory of Computing*, 2001.

[278] T. Liggett, *Interacting Particle Systems*, Springer-Verlag (Mathematical Reviews Series), New York, 1985.

[279] G. Grimmett, *Percolation,* 2*nd ed.*, New York, 1999.

[280] H. You, V. Vittal, and X. Wang, “Slow Coherency-based Islanding,” IEEE Transactions on Power Systems , Vol. 19, NO. 1, Feb. 2004.

[281] A. Pinar and B. Hendrickson, “Partitioning for complex objectives,” in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.

[282] M. Garey and D. S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, San Francisco: Freeman, 1979.

[283] “A compendium of NP optimization problems,” http://www.nada.kth.se/ ̃viggo/wwwcompendium/wwwcompendium.html.

[284] M. S. Khan, *Sequential and distributed algorithms for fast graph partitioning*, Master’s thesis, University of Victoria, Victoria, B.C., Canada, Aug. 1994.

[285] M. Stoer and F. Wagner, “A simple min-cut algorithm,” *Lecture Notes in Computer Science*, vol. 855, pp. 141-147, 1994.

[286] D. Karger and C. Stein, “A new approach to the minimum cut problem,” *Journal of the ACM*, vol. 43, no. 4, pp. 601-640, 1996.

[287] P. K. Chan, M. D. Schlag, and J. Y. Zien, “Spectral k-way ratio-cut partitioning and clustering,” *30th ACM/IEEE Design Automation Conference*, Dallas Texas, June 1993.

[288] C.L. DeMarco and J. Wassner, “A generalized eigenvalue perturbation approach to coherency,” *Proc. IEEE Conference on Control Applications*, pp. 611-617, Sept. 1995.

[289] S. Roy and B. Lesieutre. “Studies in network partitioning based on topological structure,” *32nd Annual North American Power Symposium*, Waterloo, Canada, Oct. 2000.

[290] R. D. Williams, “Performance of dynamic load balancing algorithms for unstructured mesh calculations,” *Concurrency: Practice and Experience*, 3 , 1991.

[291] F. Cao, J. R. Gilbert, and S. Teng, “Partitioning meshes with lines and planes,” *Technical Report CSL-96-01*, Xerox Palo Alto Research Center, Jan. 1996.

[292] M. Fiedler, “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory,” *Czechoslovak Mathematics Journal*, Vol. 25, No. 100, pp. 619-633, 1975.

[293] T. Nguyen Bui and B. R. Moon, “Genetic algorithm and graph partitioning,” *IEEE Transactions on Computers,* vol. 45, No. 7, July 1996.

[294] D. E. Goldberg,“Genetic algorithms in search,” *Optimization and Machine Learning*, Addison-Wesley, New York, 1989.

[295] B. Hajek, “Cooling schedules for optimal annealing,” *Mathematics of Operations Research*,Vol.13, pp. 311-329, May 1998.

[296] H. D. Simon, S-H. Teng, “How good is recursive bisection,” *SIAM Journal on Scientific Computing*, Vol. 18, No. 5, pp. 1436-1445, 1997.

[297] C. K. Cheng and Y. C. Wei, “An improved two way partitioning algorithm with stable performance,” *IEEE Trans. on Computer-Aided Design*, 10(12):1502-1511, Dec. 1991.

[298] C. H. Lee and C. I. Park, “An efficient k-way graph partitioning algorithm for task allocation in parallel computing systems,” *Proceedings of the 1st International Conference in System Integration*, pp. 748-751, 1990.

[299] D. H. Wolpert and W. G. Macready, “No free lunch theorems for search,” Technical Report, Santa Fe Institute, No.. 95-02-010, 1995.

[300] M. Mauve, J. Widmer, and H. Hartenstein, “A survey of position-based routing in mobile ad hoc networks,” *IEEE Networks Magazine*, vol. 6, pp. 30-39, Dec. 2001.

[301] A. Jadbabaie, “On geographic routing with location information,” submitted to *Proceedings of the IEEE Conference on Decision and Control*, The Bahamas, 2004.

[302] J. R. Hockenberry and B. C. Lesieutre, “Evaluation of uncertainty in dynamic simulations of power system models: the probabilistic collocation method,” *IEEE Transactions on Power Systems*, vol. 19(3), pp. 1483-1491, Aug. 2004.

[303] S. Roy, D. Ramamurthy, and B. C. Lesieutre, “Studies on the probabilistic collocation method and its application to power system analysis,” in *Proceedings of the 36th North American Power Symposium*, Moscow, Idaho 2004.

[304] J. R. Hockenberry, *Evaluation of Uncertainties in Dynamic, Reduced-Order Power System Models*, Ph.D. Thesis, *Massachusetts Institute of Technology*, Sep. 2000.

[305] S. S. Isukapalli,*Uncertainty Analysis of Transport-Transformation Models*, Ph.D. Dissertation, Program in Chemical and Biochemical Engineering, Rutgers University, 1999.

[306] M. A. Tatang, W. Pan, R. G. Prinn, and G. J. McRae, "An efficient method for parametric uncertainty analysis of numerical geophysical models," *Journal of Geophysical Research–Atmospheres*, vol. 102, no. D18, pp. 21925-21932, 1997.

[307] M. Webster, M. A. Tatang, and G. J. McRae, "Application of the probabilistic collocation method for an uncertainty analysis of a simple ocean model," *Tech. Rep. 4*, Joint Program on the Science and Policy of Global Change, MIT, Cambridge, MA, Jan. 1996.

[308] J. Sjoberg et. al., "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31(12), pp. 1691-1724, 1995.

[309] V. N. Vapnik, *Statistical Learning Theory*, John Wiley and Sons: Hoboken, NJ, 1998.

[310] K. Atkinson, *An Introduction to Numerical Analysis (2nd ed.)*, John Wiley and Sons, 1988.

[311] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*, Harcourt Brace Jovanovich, Publishers, 1984.

[312] T. R. Inniss and M. O. Ball, "Estimating one-parameter airport arrival capacity distributions for air traffic flow management," *Air Traffic Control Quarterly*, vol. 12, pp. 223-252, Aug. 2002.

[313] D. Xiu, D. Lucor, C.-H. Su, and G. E. Karniadakis, "Stochastic modeling of flow structure interactions using generalized polynomial chaos," *Journal of Fluids Engineering*, vol. 25, pp. 51-59, Mar. 2002.

[314] H. Brass, J.-W. Fischer, and K. Petras, "The Gaussian quadrature method," *Abh. Braunschweig Wiss. Ges.*, vol. 47, pp. 115-150, 1997.

[315] A. D. Fernandes and W. R. Atchley, "Gaussian quadrature formulae for arbitrary positive measures," *Evolutionary Bioinformatics Online*, vol. 2, pp. 261-269, 2006.

[316] G. W. Snedecor and W. G. Cochran, *Statistical Methods*, The Iowa State College Press, Ames, Iowa, 1956.

[317] S. Roy, "Modeling of data sets using a continuum of Beta distributions," Thesis, MIT, 1999.

[318] T. T. Soong, *Probabilistic Modeling, and Analysis in Science and Engineering*, New York: John Wiley and Sons, 1981.

[319] T. S. Chihara, *An introduction to orthogonal polynomials*, Gordon and Breach Science Publishers, 1978.

[320] A. Elbert, "Some recent results on the zeros of Bessel functions and orthogonal polynomials," *Journal of Computational and Applied Mathematics*, vol. 133, pp. 65-83, 2001.

[321] G. F. Franklin, J. D. Powell, and A Emami-Naeini, *Feedback Control of Dynamic Systems*, Prentice Hall, 2005.

[322] K. Bilimoria, B. Sridhar, G. Chatterji, K. Sheth, and S. Grabbe, "FACET: Future ATM concepts evaluation tool," in *Proceedings of the 3rd USA/Europe ATM 2001 R&D Seminar*, Naples, Italy, June 2001.

[323] K. Takahashi, et. al., "E-Cell 2: multi-platform E-Cell simulation system," *Bioinformatics* vol. 19(13), pp. 1727-1729, 2003.

[324] J. Schaff and L. M. Loew, "The Virtual Cell," in *Proceedings of Pacific Symposium on Biocomputing '99*, pp. 228-239, 1999.

[325] H. Kitano, "Systems biology: toward system-level understanding of biological systems," in Kitano (ed.) *Foundations of Systems Biology*, Cambrige: The MIT Press, pp. 1-36, 2001.

[326] K. M. Kyoda, M. Muraki and H. Kitano, "Construction of a generalized siulator for multicellular organisms and its application to SMAD signal transduction," in *Pacific Symposium on Biocomputing*, 5, pp. 314-325, 2000.

[327] U. S. Bhalla and R. Iyengar, "Emergent properties of networks of biological signaling pathways," *Science*, Vol. 283, pp. 381-387, 1999.