CONSENSUS OVER NETWORKED DYNAMICAL SYSTEMS
CONTROL, DESIGN, & INTERACTION
MATHIAS HUDOBA DE BADYN

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

DOCTOR OF PHILOSOPHY

University of Washington

June 2019

READING COMMITTEE:

Mehran Mesbahi, Chair
Kristi Morgansen
Lillian Ratliff

PROGRAM AUTHORIZED TO OFFER DEGREE:

William E. Boeing Department of Aeronautics and Astronautics
This dissertation focuses on three main aspects of control theory applied to networked dynamical systems – how to control networked systems, how to design networks to facilitate control and estimation for such systems, and how to reason about interacting with them. We examine how network symmetries and how the distribution of cooperative and antagonistic interactions facilitate control of the seminal consensus protocol, as well as measures that indicate how well a network facilitates consensus. Various tools for designing complex networks are developed in this dissertation, including decentralized update schemes for improving network edge weights and the time scales on which the agents in the network operate. The crux of the analysis of these algorithms lies in a beautiful connection between electrical network theory, the paradigm of series-parallel networks, and multi-agent consensus. Lastly, we examine how to design feedforward and feedback controllers for consensus networks that have state-dependent edge switching. The key tool is a continuum approximation of the agent dynamics, which allows for an optimal transport approach for feedforward control, and a density gradient feedback scheme.
Dedicated to my parents and my grandmother.
Some ideas and figures have appeared previously in the following publications:

INCLUDED PUBLICATIONS


I've never seen the Icarus story as a lesson about the limitations of humans.
I see it as a lesson about the limitations of wax as an adhesive.
— Randall Munroe

ACKNOWLEDGMENTS

I would first like to thank my parents for raising me, and for being scientific role models. I also would like to apologize to my mother for asking if I could replace her with Ms. Frizzle at the tender age of 3. She was really the true Ms. Frizzle all along, except with a Subaru Forester instead of a nonphysical school bus. Although the amount of clutch replacements on that vehicle does make me suspect it was used for some sort of offroad scientific exploration.

Secondly, I want thank the staff at the William E. Boeing Department of Aeronautics and Astronautics. From even my undergraduate days when I was applying to the AA department, Ed Connery and Wanda Frederick treated me with the same friendliness and respect that carried over when I eventually joined the university as a doctoral student. I wish Wanda all the best in her retirement. The other members of the staff were also fantastic to work with over the years. I’d like to thank Victor Aque, Jenny Park, Winnie Lin, Leah Panganiban, Pam McGrath, and all the other staff members whom I didn’t interact with personally, but I’m sure still contributed strongly to my success here.

I am incredibly grateful to Mehran Mesbahi for being my advisor, mentor, and for helping me travel the world. His hard work behind (literal) closed doors allowed me to pursue my dreams of becoming a scientist (a rocket one, at that!), and his calm, friendly demeanour was a great blanket, filling me with solace in the face of the soul-crushing academic world. I also want to thank the members of my qualifying exam, general exam, and dissertation committees. Mehran Mesbahi, Christopher Lum, Setthivoine You, Kristi Morgansen, Jon Wellner and Lillian Ratliff all helped me enormously in my growth as a student of control theory.

To the Canadian and American taxpayer, I thank you for giving me money. In particular, I was generously funded by the National Sciences and Engineering Research Council of Canada under grant CGSD2-502554-2017, the U.S. Army Research Lab and U.S. Army Research Office under contract number W911NF-13-1-0340, and the U.S. Air Force Office of Scientific Research under grant number FA9550-16-1-0022. This work could not have been completed with your help. In this list, I also have to thank the project managers and coordinators on the Network Control MURI Grant, which include Alfredo Garcia, Purush Iyer, and Ananthram Swami. I’m also very grateful to the Erwin Schrödinger International Institute for hosting me from May 3–May 19 of 2019 during the Optimal Transport Workshop.

For the Network Control MURI Grant, I have to thank all of my collaborators, and especially Raisa D’Souza for doing a fantastic job organizing this massive scientific endeavour. The team of course includes Mehran, but also Michael Roukes, Jim Crutchfield, Leonardo Duenas-Osorio, Anastasiya
Salova, Martin Rohden, Airlie Chapman, Matt Matheny, Jeff Emmenheiser, Warren Fon, Jordan Snyder, Kelly Finn, Siavash Alemzadeh, Afshin Mesbahi, Brianne Beisner, Brenda McCowan, and many others.

From the University of Washington library system, I would like to thank Lisa Bacis and all of the librarians that found books and scanned old articles for me through the online request portal. The UW Libraries are truly world-class, in great part due to the work of the staff.

I’ve made many friends at the University of Washington, some of whom contributed to my success, and some of whom may or may not have decontributed to my success. The members of the Robotics, Aerospace and Information Networks lab have most definitely been the former – Dillon Foight, Siavash Alemzadeh, Taylor Reynolds, Bijan Barzgaran, Jingjing Bu, Tom Miesen, Aditya Deole, Dian-Jing Chen, Soumya Vasisht, Caroline Schlemmer, Airlie Chapman, Afshin Mesbahi, Unsik Li, Eric Schoof, Robert Vasil, Kerney Wu, Pietro Pierpaoli, Saghaf Hosseini, Will Howerton, Dave Besson, Shariar Talebi, Newsha Rahimi, and Mengyuan Wang. My collaborators in the other control labs have also been instrumental to my success. I’d like to thank Utku Eren, Adam Tahir, Miki Szmuk, Sarah Li, Yue Yu, Sierra Adibi, Armand Awad, Emma Hansen, Max Spetzler, Alison Zongolowicz, Dan Calderone, Doris Voina, Skye Mceowen, Elliot Claveau, Chris Everson, Alice Liu, Ena Hodzic, Pablo Posada, Gustavo Fujiwara, Sean Phenisee, Tony Deleo, Jithin George, Ellie Forbes, Abhiram Aiwal, Dawei Lu, Osazonamen Igbinosun, Danylo Malyuta, Sean Rice and everyone else that I forgot here. A special place in my heart goes out to Armin Mortazavi, the Sports Champion; and Michael Grudich, the Fortran master – the two brothers I never had.

There are far too many professors to list who have significantly impacted my success, but I will try nonetheless to do this justice. Back in my early undergraduate days, Mark Van Raamsdonk was one of the first who taught me how the academic career path looked like, and helped me land my first research job. In that job, I must thank James Owen for being my advisor and helping me get my first paper published, and for introducing me to fine ales. My second research job was with Richard Marchand and Richard Sydora, who helped me author my first first-author paper, and I will be forever grateful. Joanna Karczmarek taught me so many things while I did my undergraduate physics thesis with her, but most of all she believed in me more than I did. Greg Martin’s patience and guidance as I did my undergraduate mathematics thesis with him helped me mature as a mathematician, at least a little bit.

When I entered graduate school, the faculty at the Aeronautics & Astronautics departments were the key to any success I have achieved. Apart from Mehran Mesbahi, the remainder of the controls faculty including Kristi Morgansen, Christopher Lum, and Behçet Açıkmeşe all helped me in my transition to a new field, and propelled me in a direction that helped me publish a number of papers. In electrical engineering, Maryam Fazel and Sam Burden taught me a great deal about optimization and reinforcement learning. These tools I will take with me for the rest of my career. I owe a huge debt of gratitude to Lillian Ratliff for being a constant source of advice about research goals, and the academic job market. Her comments on my application packet were paramount in my success in obtaining a job after graduating.
In my dual life in the mathematics department, I received a warm welcome, despite being a practitioner of the forbidden arts of engineering, from many of the students and faculty. In particular, I would like to thank Sara Billey for her aid in understanding combinatorics, as well as Dmitriy Drusvyatskiy and Soumik Pal for their work as my *de facto* advisors during my mathematics Master’s. Ioanna Dumitriu cemented my knowledge of linear algebra, which of course is the basis of all of control theory, and Jim Burke taught me about line search methods. I’m especially grateful to Jack Lee and John Palmieri for admitting me to the Master’s program in the first place.

I’m grateful to Si-Gung John Leong, Sifu Alan Houp and Si-Jie Lotus Houp at the Seattle Kung Fu Club for the wonderful time learning martial arts throughout the later years of my PhD. The Civil Air Patrol, Seattle Squadron, had the patience to sit through several of my presentations, appropriately titled “Droning About Drones”. They taught me a lot about how many of the formation control algorithms presented in this work can be used in practice in search-and-rescue operations, and hopefully one day save lives. I would like to thank the bartenders at Big Time Brewing and the Toronado, especially the late Matt Bonney. I’d pour one out in celebration of our times together, but given your former profession, that seems inappropriate. I only wish I had the chance to say goodbye.
# Contents

## I Networks

1 **Consensus on Networks**
   1.1 Networks and Networked Dynamical Systems ............. 2
   1.2 Why Consensus? ........................................ 3
      1.2.1 Consensus in Coordination [1] ....................... 3
      1.2.2 Consensus in Oscillator Networks [2] ............... 4
      1.2.3 Consensus in Opinions [3] .......................... 5
      1.2.4 Consensus in Natural Systems [4] .................... 5
   1.3 Overview of Thesis .................................... 6
   1.4 Contributions ........................................... 7
      1.4.1 Part 2 – Control .................................... 7
      1.4.2 Part 3 – Design ..................................... 8
      1.4.3 Part 4 – Interaction ................................. 10
   1.5 Preliminaries ............................................ 11
      1.5.1 Notation ............................................. 11
      1.5.2 Consensus and Algebraic Graph Theory ............. 12

## II Control

2 **Signed Consensus Networks**
   2.1 Mathematical Background ............................... 16
      2.1.1 Signed Networks .................................... 16
      2.1.2 Automorphisms/Interlacing/Equitable Partitions ... 16
   3 **Signed Control**
      3.1 Problem Statement .................................... 18
      3.2 Methods ................................................ 20
      3.2.1 Signed Graph Controllability ...................... 20
      3.2.2 Stabilizability and Output Controllability ....... 26
      3.3 Nonlinear Controllability of Signed Networks ....... 29
      3.4 An Aside on Structural Balance ..................... 34

## III Koopman Analysis on Signed Networks

4 **Koopman Analysis on Signed Networks**
   4.1 Koopman Operator Theory ............................... 38
   4.2 Identification of Bipartition in Structurally Balanced Graphs with the Koopman Operator and EDMD .......... 45

## IV Examples

5 **Examples**
   5.1 Bipartite Consensus .................................... 46
   5.2 MIMO Perturbed Laplacian .............................. 47
   5.3 Influenced Consensus ................................... 47
   5.4 Unsigned Symmetry is Not Sufficient for Uncontrollability ......................................................... 48
   5.5 Identification of Bipartite Structure .................. 49

## VI Distributed State Estimation

6 **Distributed State Estimation** ............................ 51
6.1 Motivation for Distributed State Estimation .......................... 51
6.2 General Kalman Filtering ............................................. 52
  6.2.1 Gradient Descent for the Error Covariance Update:
    LinearCase ................................................................ 52
  6.2.2 Nesterov-Accelerated Methods .......................... 52
  6.2.3 Adaptive Learning Rate Methods .................. 53
  6.2.4 Filter Properties ................................................. 54
  6.2.5 Example ............................................................... 55
6.3 Distributed Kalman Filtering ............................................. 56
  6.3.1 Example ............................................................... 57
6.4 Gradient Descent for Kalman Filter Covariance Update .... 58
6.5 Conclusions and Future Works ........................................ 61

III DESIGN
7 INTRODUCTION ................................................................. 64
  7.1 Mathematical Preliminaries ........................................... 68
    7.1.1 Submodular Optimization ........................................ 68
8 GRAPH GROWING ............................................................ 70
  8.1 Graph Growing with Submodular Optimization .......... 70
    8.1.1 Single Leaf Addition ............................................. 75
    8.1.2 Multi-Leaf Addition ............................................ 78
    8.1.3 Fully Whiskered Graph ......................................... 80
  8.2 Examples ................................................................. 82
    8.2.1 Optimization Algorithms: Adding Leaves ........ 85
    8.2.2 Optimization Algorithms: Adding Clusters ........ 88
  8.3 Algorithm Implementation ............................................. 89
9 MATRIX-WEIGHTED GRAPHS ........................................... 92
  9.1 Leader-Follower Consensus Network Models .................. 93
    9.1.1 Electrical Networks ............................................ 95
    9.1.2 Series-Parallel Networks ...................................... 97
  9.2 System-Theoretic Analysis on Series-Parallel Graphs ..... 99
    9.2.1 Noise Rejection and Adaptive Weight Design .......... 101
    9.2.2 Synthesis of Optimal $\mathcal{H}_2$ Controllers .......... 106
  9.3 Example ................................................................. 111
  9.4 Proof of Proposition 5: .............................................. 111
10 TIMESCALED CONSENSUS ............................................ 115
  10.1 Leader-Follower Consensus Models with Time Scales .... 115
    10.2 Performance and Design Problems .................. 120
      10.2.1 $\mathcal{H}_2$ Performance .................................. 120
      10.2.2 Timescale Design For $\mathcal{H}_2$ Resilience .......... 123
      10.2.3 Decentralized Gradient Updates for Optimal $\mathcal{H}_2$
        Performance .......................................................... 126
      10.2.4 Decentralized Gradient Updates on Weights for Opt-
        imal $\mathcal{H}_2$ Performance ....................................... 127
11 EXAMPLE: FLOCKING VIA SECOND-ORDER CONSENSUS .... 130
  11.0.1 Scalar State Case .............................................. 130
  11.0.2 Matrix-Valued Double Integrator Consensus ........ 132
  11.0.3 Numerical Example .............................................. 134
# CONTENTS

## 12 Network Entropy

12.1 Introduction ............................................. 140
12.2 Network Model ........................................... 140
   12.2.1 Mathematical Notation and Graph Theory .......... 141
   12.2.2 Network Models ..................................... 141
12.3 Loop Entropy and Spanning Trees ....................... 142
   12.3.1 Counting Loops .................................... 144
   12.3.2 Loop Entropy and Network Gramians .............. 145
12.4 Kolmogorov-Sinai Entropy ................................ 147
   12.4.1 Bounds on the Kolmogorov-Sinai Entropy .......... 149
   12.4.2 Kolmogorov-Sinai Entropy of Adjacency-driven Networks .................. 150
12.5 Examples of Entropies for Simple Graphs ............... 150
12.6 Conclusion .............................................. 151

## 13 Interaction

13.1 Mathematical Preliminaries .............................. 154
   13.1.1 Mathematical Notation & Measure Theory .......... 155
   13.1.2 Optimal Mass Transport .................................. 155
   13.1.3 Density Control and Kernel Density Estimation .... 157
13.2 Problem Statement ........................................ 158
13.3 Feedback Control of State-Dependent Networked Dynamical Systems .......................... 160
13.4 Density Estimation for Kernels With Compact Support .... 161
13.5 Examples ................................................ 164
13.6 Conclusion ............................................... 164

## V Conclusion and Future Work

14 Concluding Remarks ........................................ 168
15 Future Work .................................................. 170
   15.1 Control ................................................. 170
   15.2 Design ................................................... 171
   15.3 Interaction ............................................. 172

BIBLIOGRAPHY .................................................. 175
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Equitable partition of a sample graph, (b) quotient of the graph in (a).</td>
<td>17</td>
</tr>
<tr>
<td>3.1</td>
<td>A network topology with input symmetry</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of a signed automorphism</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>Cartoon of the Koopman operator approach: the composition operator 'lifts' a nonlinear manifold $\mathcal{M}$ into a linear, infinite-dimensional manifold on which observables of the state $x \in \mathcal{M}$ evolve.</td>
<td>43</td>
</tr>
<tr>
<td>5.1</td>
<td>Solid lines: unsigned consensus. Dashed lines: signed consensus on structurally balanced graph.</td>
<td>46</td>
</tr>
<tr>
<td>5.2</td>
<td>(a) Uncontrollable and structurally balanced graph with $\pi_f$ containing nontrivial cells in $\pi$. (b) The same system with one change in the signs. No structural balance and controllable.</td>
<td>47</td>
</tr>
<tr>
<td>5.3</td>
<td>(a) Structurally balanced graph with a leader symmetry about node 4. (b) Structurally unbalanced graph with a leader symmetry about node 4.</td>
<td>48</td>
</tr>
<tr>
<td>5.4</td>
<td>Underlying structurally balanced signed graph for EDMD example. Dashed edges indicate negative edges.</td>
<td>49</td>
</tr>
<tr>
<td>5.5</td>
<td>Evolution of Dynamics 3.7 with $f(\cdot) = \sin(\cdot)$ on the graph in Figure 5.4. Left (red) shaded region corresponds to $\tilde{v}_1^2$ and the right (blue) shaded region corresponds to $\tilde{v}_1^3$.</td>
<td>50</td>
</tr>
<tr>
<td>6.1</td>
<td>Numerical test of Algorithm 1. Top: Error from true state from data, accelerated filter and Kalman filter. Middle: $y_1 = x_1$ for data, accelerated filter and Kalman filter. Bottom: Average steady-state error.</td>
<td>56</td>
</tr>
<tr>
<td>6.2</td>
<td>Sensor network graph with 178 nodes and 186 edges.</td>
<td>58</td>
</tr>
<tr>
<td>6.3</td>
<td>True state of system at $t = 1.2s$</td>
<td>59</td>
</tr>
<tr>
<td>6.4</td>
<td>Sample measurement $\tilde{y}$ over the entire system at $t = 1.2s$</td>
<td>59</td>
</tr>
<tr>
<td>6.5</td>
<td>Sample system estimate at $t = 1.2s$ from a single sensor</td>
<td>59</td>
</tr>
<tr>
<td>8.1</td>
<td>Whiskering a graph by adding a leaf to every node</td>
<td>71</td>
</tr>
<tr>
<td>8.2</td>
<td>Adding a leaf and a path of length 2 to every node</td>
<td>72</td>
</tr>
<tr>
<td>8.3</td>
<td>Three examples of graph whiskering, original graph is solid, added nodes are hollow. (a) Single leaf. (b) Multi-leaf. (c) Fully whiskered.</td>
<td>78</td>
</tr>
<tr>
<td>8.4</td>
<td>Original graph (solid) and added leaf (hollow) with control input to first node.</td>
<td>82</td>
</tr>
</tbody>
</table>
Figure 8.5  Multi-leaf whiskering example. Original graph (solid) and added leaves (hollow) with control input to first node. ................................................................. 84

Figure 8.6  Max-\(\lambda_2\) algorithm results for adding leaves (top) and path clusters (bottom), using the convex relaxation (\(\times\)), exhaustive search (\(\ast\)) and perturbation heuristic (\(+\)). ................................................................. 89

Figure 8.7  Seed graph, and final graph after 9 iterations of the leaf-adding problem using the SDP relaxation (8.14), exhaustive search over problem (8.13) and the perturbation heuristic. ................................................................. 90

Figure 8.8  Seed graph, and final graph after 9 iterations of the path-cluster-adding problem using the SDP relaxation (8.16), exhaustive search over problem (8.15) and the perturbation heuristic. ................................................................. 90

Figure 9.1  Top: leader-follower network setup. Right: electrical “grounding” of the leader set \(R\), and current vector \(e_s = e_s \otimes I_k\) injected into the network via node \(v_s\). \(y^i_s\) is the voltage dropped from \(v_i\) to \(R\). Bottom: Identification of grounded leader set into a single node. ................................................................. 94

Figure 9.2  Decomposition trees of two graphs. .............................. 97

Figure 9.3  Decomposition trees of the all-input TTSP in Figure 9.5 ................................................................. 98

Figure 9.4  Nested infimal convolution/sum computations over a series-parallel graph. Beige circles (always on the left) denote sources, red squares (always on the right) denote sinks at each step. ................................................................. 100

Figure 9.5  Left: An all-input TTSP graph. Right: Parallel joins across \(R\) to \(R\) used to compute \(Y_s^2\). ................................................................. 105

Figure 9.6  Series and parallel joins of weighted 1-paths. .............................. 108

Figure 9.7  Edges weights for example at iteration 1 ................................................................. 111

Figure 9.8  Edges weights for example iteration 3 ................................................................. 112

Figure 9.9  Edges weights for example iteration 5 ................................................................. 112

Figure 9.10  Edges weights for example iteration 7 ................................................................. 112

Figure 9.11  Convergence of gradient descent weight update for optimal \(H_2\) performance on the graphs in Figures 9.7-9.10 ................................................................. 113

Figure 10.1  Tree graph for six agents, \(T\). Each agent has an associated scaling parameter, \(\epsilon_i\), with subscript denoting the agent number, and each edge is similarly labeled \(w_j\). Agent 3 has the highest degree. .............................. 123

Figure 10.2  Random graph topology for 10 agents. The node number and time scale assigned by (P1) are printed in each node, which shows the slowest time scales are assigned to nodes with highest degree. ................................................................. 124

Figure 10.3  Spanning tree of the graph in Figure 10.2 used to calculate the optimal time scale distribution. ................................................................. 125
Figure 10.4  Spanning path graph of the graph in Figure 10.2. When used in place of the spanning tree in Figure 10.3, the same distribution of time scales results.

Figure 11.1  Relative formation of the agents, defined by discrete points on a spiral.

Figure 11.2  Cost of weight selection versus iteration.

Figure 11.3  Visualization of edge weights at $k = 1$.

Figure 11.4  Visualization of edge weights at $k = 2$.

Figure 11.5  Visualization of edge weights at $k = 3$.

Figure 11.6  Visualization of edge weights at $k = 4$.

Figure 11.7  Visualization of edge weights over iterations $k = 1$ (top left), 2 (top right), 3 (bottom left) & 4 (bottom right). Each of the 3 independent parameters of the $2 \times 2$ matrix-valued weight is visualized in a multi-graph.

Figure 11.8  Edge states over time subjected to gust at $10 \leq t \leq 20$; top two plots are the edge states in the $x, y$ directions respectively with the weight update, and the bottom two plots are the edge states in the $x, y$ directions respectively with no weight update.

Figure 11.9  Variance from consensus for a representative edge $[x_e(t) - x_e(t_f)]^2$ for the case with the weight update and without.

Figure 12.1  The two flow graphs of the transfer function of a 4-cycle controlled from a single node.

Figure 12.2  Adding self-loops to stabilize the adjacency dynamics.

Figure 12.3  Left: Adding an edge in a cycle increases entropy by adding spanning trees. Middle: Adding an edge to a star graph adds two spanning trees. Right: Creating a cycle out of a path adds $n - 1$ spanning trees.

Figure 13.1  Block diagram of density control scheme.

Figure 13.2  Illustration of (proximity-based) state-dependent constraints on the KDE procedure. Dotted lines indicate samples the center agent cannot measure.

Figure 13.3  Optimal kernels with unconstrained support, and support constrained to $[-2, 2] \setminus [1/4, 3/4]$, with second moment $a = 5^{-1/2}$. The unconstrained kernel solution is exactly given by the Equation (13.14).

Figure 13.4  Left: Initial density $\rho_0$. Right: Target density $\rho_1$.

Figure 13.5  Optimal density profiles over time, and superimposed agent states using the feedback density control law.

Figure 13.6  Optimal density profiles over time, and superimposed agent states with only the feedforward control.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>Examples of Loop Entropies</td>
<td>151</td>
</tr>
<tr>
<td>12.2</td>
<td>Examples of KS Entropies</td>
<td>151</td>
</tr>
</tbody>
</table>
Part I

NETWORKS
Humans as a species are, by nature, social. From the dawn of civilization, humans have banded together and formed social groups, with complex interactions both between groups and between individual human beings. As civilization and technology progressed over time, this intrinsic desire to create bonds between each other and to stay connected has been reflected in the very core of things we have created. The wonders of the modern age encompass miraculous technologies such as the Internet, wireless communication, and global transportation in the form of roads and airline traffic. Although not all of the effects of these technologies on human society and the planet at large are positive, their very existence is the result of a desire to understand the physical laws of the world, and to close the the distance between individual humans.

At the core of human interaction is the notion of a social network – pairwise interactions between humans. Such interactions can be considered friendly or antagonistic, or something more akin to the (in)famous Facebook relationship status “It’s Complicated”. Humans are clearly not the only socially interacting living beings, nor the only natural or synthetic system that is described by a network. The goal of this dissertation is to exposit a general framework for studying interactions of dynamic agents on networks – how the structure of the network affects the dynamics and behaviour of people, processes and information propagating over the networks.

Networks are comprised of a system of individual agents, and interactions among those agents. In recent years, a paradigm of studying distributed systems as an emergent structure out of a network has attracted significant research interest. The individual agents in a network can make local, autonomous, and decentralized decisions in order to coordinate to some global objective. For large networks, this allows for distributing out computational resources across many agents, thus requiring lower computational power in each agent. Furthermore, large networks are also robust to individual agent failures – many agents means greater redundancy.

Such networked dynamical systems can be synthetic – for example, cyber-physical systems such as power grids or the Internet, or even the Internet of Things. A topic of significant interest and potential impact is autonomous robotics, ranging from swarms of unmanned aerial vehicles for disaster response or space exploration, to autonomous cars coordinating together on a traffic network. Natural networked dynamical systems include the flocking of animals, such as geese, fish or buffalo, or abstract notions of distributed systems such as genetic regulatory networks.

All of these networked dynamical systems share the key feature of having several notions of dynamics. Each agent has their own decision-making dynamics, but this is influenced by a higher-order protocol that dictates a
dynamical interaction, or information sharing, between agents. As a mathematical object, networks are a powerful analysis tool for separating these two notions.

In this thesis, we examine the latter – the notion of information-sharing protocols over networks that agents use to augment their internal decision-making, or their self-dynamics. The most common and useful such protocol is called consensus. This begs the question addressed in the next section.

1.2 Why Consensus?

Consensus is a popular algorithm for allowing a networked dynamical system to agree on a common value of some parameter utilized in their self-dynamics. It is a simple protocol in principle – each agent averages the value of the parameter that they hold with that of their neighbours. However, consensus has a rich theoretical literature. Despite its simplicity, it is a protocol that is very much affected by the intrinsic structure of the network it is running on, and is thus amenable to analysis using graph theory.

Because of its control-theoretic pedigree, it is one of the premier protocols of choice when attempting to study the interplay between dynamics, or the study of differential equations, with the discrete world – that of combinatorics, graph theory and network science. Furthermore, many natural systems actually have dynamics that are captured by consensus as a first-order approximation.

As part of the motivation of the work in this thesis, we outline some common examples below.

1.2.1 Consensus in Coordination [1]

Consider the problem of $n$ agents trying to flock, i.e. move in the same heading and velocity. Suppose, for simplicity, that the agents are double-integrator Newton particles:

$$
\dot{q}_i = p_i \\
\dot{p}_i = u_i,
$$

where $1 \leq i \leq n$ denotes the agent, and each $q_i, p_i \in \mathbb{R}^3$. A choice of control that will achieve flocking is the following:

$$
u_i = -\nabla V(q) - L_G p_i,$$

where $L_G$ is the graph Laplacian, and $V(q)$ is a potential function of the form

$$V(q) = \frac{1}{2} \sum_i \sum_{j \neq i} \psi_\alpha(||q_j - q_i||).$$

Here, $\psi_\alpha$ is a function describing the interaction between agents – it may increase or decrease as a function of distance $||q_j - q_i||$ between agents $i$ and $j$. 
The term $-\mathcal{L}_G p$ is what is known as consensus. The graph Laplacian $\mathcal{L}_G$ both encodes the connectivity structure of the network, but also encapsulates the notion of distributed averaging – the quantity $p_i$ is averaged continuously over time with the neighbours of $i$.

1.2.2 Consensus in Oscillator Networks [2]

A nanoelectromechanical oscillator is a small, piezoelectric membrane approximately 100µm in size, consisting of a clamped aluminum nitride ultra-thin plate. Such oscillators can be electrically coupled, and tuned to precise resonant frequencies with high quality factors. Consider a system of nanoelectromechanical oscillators, electrically coupled in a ring (each oscillator is connected to two others). Then, the dynamics of the $j$th oscillator can be described by a complex-valued amplitude $A_j$ satisfying

$$\frac{dA_j}{dt} = -\frac{A_j}{2} + \frac{A_j}{2|A_j|} + i \left[ \omega_j A_j + \alpha |A_j|^2 A_j \right] - \frac{i\beta}{2} (A_{j-1} + A_{j+1}).$$

The coupling between oscillator $j$, and its neighbours $(j-1)$ and $(j+1)$ is given by the last term $\frac{i\beta}{2} (A_{j-1} + A_{j+1})$.

By considering the phasor $A_j = a_j e^{i\phi_j}$, Equation (1.1) can be decomposed into amplitude ($a_j$) and phase ($\phi_j$) dynamics:

$$\frac{da_j}{dt} = \frac{1 - a_j}{2} - \frac{\beta}{2} \left[ a_{j+1} \sin(\phi_{j+1} - \phi_j) + a_{j-1} \sin(\phi_{j-1} - \phi_j) \right]$$

$$\frac{d\phi_j}{dt} = \omega_j + \alpha a_j^2 - \frac{\beta}{2a_j} \left[ a_{j+1} \cos(\phi_{j+1} - \phi_j) + a_{j-1} \cos(\phi_{j-1} - \phi_j) \right].$$

In Equation (1.2), the term

$$[a_{j+1} \sin(\phi_{j+1} - \phi_j) + a_{j-1} \sin(\phi_{j-1} - \phi_j)]$$

is known as Kuramoto coupling, after the famed Kuramoto oscillator model.

If oscillator $j$ is connected to more than two other oscillators, we can write this term as

$$\sum_{k \in N_j} [a_k \sin(\phi_k - \phi_j)],$$

where $N_j$ is the set of neighbours of oscillator $j$. When the oscillators are phase synchronized, in that $\phi_j \approx \phi_i$, then $\sin(\phi_k - \phi_j) \approx \phi_k - \phi_j$, and so this term becomes

$$\sum_{k \in N_j} [a_j (\phi_k - \phi_j)] = -\mathcal{L}_G \phi,$$

where $\phi$ is the stacked vector of the phases. Thus, in the natural dynamics of certain physical oscillator networks, the graph Laplacian captures the
steady-state notion of synchronization – one dynamic interpretation of ‘agreement’.

1.2.3 Consensus in Opinions [3]

Suppose that \( n \) individuals are discussing their opinion on a certain topic. For example, some individuals may prefer cats over dogs, and others dogs over cats, and some may be neutral\(^1\). One way of encoding this is by assigning a numerical value \( x_i \) over a spectrum of opinions to each agent, with one extreme representing cat people, and the other dog people.

One model of studying how opinions propagate in such a circumstance is known as the opinion dynamics with bounded confidence. This is a discrete-time process, in which each agent only averages their opinion with neighbours whose opinions are \( \epsilon \)-close to theirs:

\[
x_i(t+1) = \frac{1}{|N_i(t)|} \sum_{j \in N_i(t)} x_j(t).
\]

Here, \( N_i(t) \) is the set of neighbours of agent \( i \), which is given by

\[
N_i(t) = \{ j : |x_i(t) - x_j(t)| \leq \epsilon_i \}.
\]

Note that this set of neighbours is time-dependent: as opinions change over time, two agents who find themselves at too large of a disagreement will cease contact with each other. This causes them to no longer share information, and thus they no longer influence each other.

The dynamics in Equation (1.3) can be interpreted as a discrete-time consensus protocol. Since Equation (1.3) is nothing but a linear combination of some of the agent states \( x_j \), this can be interpreted as a matrix-vector multiplication. The matrix will have a zero-structure corresponding to the connection structure of a dynamic graph – two agents are connected if and only if their opinions are \( \epsilon_i \) (or \( \epsilon_j \)) close.

1.2.4 Consensus in Natural Systems [4]

In an earlier subsection, we introduced the flocking problem – what control signals should agents compute and actuate in order to achieve consensus on heading and velocity? The inverse question is also appropriate: given a natural system that flocks, say of birds or fish, what dynamics accurately capture the observed behaviour?

---

\(^{1}\) Such an example may sound contrived, but there are many other pressing discussions of this form that include some notion of bipartisanship. For example, politics, or the debate between vim versus emacs.
Cucker and Smale proposed a flocking model based on distributed averaging – at time $t$, each bird $i$ does a discrete-time averaging process on velocity (here $v_i$ is a scalar):

$$v_i(t + 1) - v_i(t) = \sum_{k=1}^{k} a_{ij} (v_j(t) - v_i(t)).$$

The weight $a_{ij}$ is computed as a function of the distance between birds – a bird may wish to average with its closer neighbours more readily than neighbours farther away, both for deconfliction purposes as well as the simple fact that velocity is likely judged more accurately at close distances.

Cucker and Smale proposed the function

$$a_{ij} = \frac{K}{(\sigma^2 + |x_i - x_j|)^\beta},$$

where $\sigma > 0$, $\beta \geq 0$ and $K$ are all tuning parameters.

The dynamics in Equation (1.4) can then be written in terms of the graph Laplacian and the stacked velocity vector $v$:

$$\begin{align*}
x(t + 1) - x(t) &= v \\
v(t + 1) - v(t) &= -L_G(x)v.
\end{align*}$$

A continuous-time process may be inferred by taking appropriate limits between timesteps as:

$$\begin{align*}
\dot{x} &= v \\
\dot{v} &= -L_G(x)v.
\end{align*}$$

This is, again, a consensus algorithm, albeit with a \textit{state-dependent} graph Laplacian.

1.3 \textbf{Overview of Thesis}

In this thesis, we use consensus as a prototypical example of a networked dynamical system on which the effects of the network topology on the dynamics becomes apparent. The research discussed in this thesis all relates to answering questions regarding how the network can be used, abused or tuned when designing distributed control systems.

The structure of the network allows for a combinatorial number of additional parameters to tune when considering control design for distributed systems. For example, one can consider the choice of the actual network structure as a design choice, as well as the strength or type of interaction between individual agents. The notion of feedback control now can take into account the state of neighbouring agents. Even individual parameters for how the agent takes the information from its neighbours into account when computing its control action can be considered a control design problem.
As such, it is easy to lose focus in the large breadth of questions one can ask in this research area. In this thesis, we separate the task of examining the effect of the network on the system into three questions.

1. How does the network structure affect the controllability of a networked dynamical system?

2. How can one design a network \textit{a priori} to facilitate control? What are the right metrics on which to measure this performance?

3. How can one interact with a networked dynamical system? How do you hand a joystick to an unskilled operator and tell them to guide a swarm of robots?

\textbf{1.4 contributions}

The three questions above are each addressed in a separate part of this thesis – control, design and interaction. The motivations for each part, and the contributions and publications therein are discussed below.

\textbf{1.4.1 Part 2 – Control}

The most natural question to ask is if given a network, does that network render a networked dynamical system controllable, or uncontrollable. This question of course relies on how one intends the network to be influenced. One way to influence a network is to identify a certain number of agents near the boundary of the network, and either take over some part of their state, or simply inject some control signal into the system represented by that agent.

Previous work in the literature has identified that for consensus networks, symmetries fixing control input nodes renders the system uncontrollable. The first contribution of this part is to examine how this controllability condition changes when you consider new classes of interagent interactions. The motivation behind these interactions is the study of social networks. In social networks, one (hopefully) has some friends, and also potentially some enemies. One can draw a graph representing such a social network by assigning each interaction edge a scalar weight, with a positive weight denoting a friendly (cooperative, or excitatory) interaction, and a negative weight denoting an unfriendly (anti-cooperative, or inhibitory) interaction.

The consensus protocol on such a network exhibits interesting behaviour, in particular the network will 'cluster' into two groups each achieving consensus on their own.

We show that a certain distribution of negative edge weights, coupled with a network symmetry fixing the input node, renders the resulting system uncontrollable. The distribution is referred to as 'structural balance', and essentially means that for any three agents that have existing relationships, either the mathematical encodings of the phrases 'the friend of a friend is a friend', or the 'enemy of an enemy is a friend' is satisfied. We also derive a similar uncontrollability condition for nonlinear variants of consensus.

In structurally balanced networks, the signed consensus protocol exactly achieves the aforementioned clustering. It turns out that the two clusters
that are formed all have positive (friendly) interactions between agents in the same cluster, but only negative (unfriendly) interactions exist across clusters. Given some time-series data of a nonlinear signed networked dynamical system, it is of interest to be able to identify the exact agents in each cluster. We use novel algorithms related to the Koopman operator that allow one to reduce the problem of community detection to some simple linear algebra calculations on the data stream. In particular, we show that the sign structure of what is known in context as a Koopman mode exactly encodes which agents are in which cluster. This can be used, for example, to identify friend-enemy relationships in complex social networked systems.

Lastly, we examine the problem of state estimation on high-dimensional systems. For example, consider the problem of identifying the pressure or density of a weather system over a large geographical area from data collected by stations across that area. By discretizing the system into a grid, one can consider the pressure or density in each grid cell to be one element of a large state-space. State-estimation algorithms, such as Kalman filtering, are computationally intractable due to matrix inverses on high-dimensional systems. We expand on a method of Sutton to find the best diagonal matrix estimate of the covariance matrix used to compute the Kalman filter, allowing this computation to become tractable. We employ a consensus-based algorithm to allow each station to agree on the value of the state estimate, given that each station can only measure a finite region of the state space.

1.4.2 Part 3 – Design

The majority of this thesis focuses on the question of design – how do you build up a network from scratch to ensure that it ‘performs’ well? What does it even mean to ‘perform’ well?

One metric of performance is simply controllability – is the network controllable? This is a binary performance measure. The first contribution of this part is a algorithm called ‘whiskering’ which looks at a certain kind of
rooted graph product to build up a network from scratch so that controllability is preserved. We also examine the case where a few agents are being added to a network – which agents in the already existing network can you attach them to, and what is the best way to do this?

One measure of how a system ‘performs’ is known as the $\mathcal{H}_2$ performance. Roughly speaking, this measures how much ‘energy’ the system accumulates as it is being driven by random noise. A system that performs ‘well’ has a low $\mathcal{H}_2$ norm – for example, a swarm of UAVs can coordinate better if their positions and velocities are not disturbed much when they encounter turbulence. We examine the $\mathcal{H}_2$ norm in several contexts.

First, we use the paradigm of series-parallel networks. These are a class of networks that lack a homeomorphism to $K_4$, the complete graph on 4 nodes. Many difficult (read – NP-hard) combinatorial problems, such as the minimum vertex cover and maximum matching problems, have linear-time solutions on series-parallel networks. The key is a $O((\log n)$ decomposition algorithm that takes a series-parallel network and decomposes it into small atomic elements, and ‘series’ & ‘parallel’ operations on those elements – think of adding resistors in series or parallel.

This allows us to do two things. First, we can compute the $\mathcal{H}_2$ norm of a consensus network very quickly by exploiting this decomposition. The standard solution of computing the $\mathcal{H}_2$ norm requires a matrix inverse, making this calculation $O(n^\omega)$ where $\omega$ is the matrix multiplication complexity (roughly 2.3 as of now). Our method allows us to do this in $O(kn)$ to $O(k\log n)$ complexity, where $k$ is the number of input nodes. Secondly, a similar algorithm also allows the network to distributively update the edge weights in order to optimize the $\mathcal{H}_2$ norm.

The second context in which we study the $\mathcal{H}_2$ norm is on networks with multiple time scales. These are systems in which some states are ‘faster’ than others – in the context of consensus networks, this ‘speed’ is the rate at which an agent integrates information from its neighbours. We consider edge consensus, which is a decomposition of the consensus problem into examining information flow along edges between agents, rather than on the agents themselves. The reason we do this is because we cannot analytically work with the $\mathcal{H}_2$ norm of time scaled consensus on the agent states – the effective graph Laplacian in this case is not a symmetric matrix, and there are no standard techniques (other than looking at edge consensus) to deal with this additional complexity.

What we show is that the $\mathcal{H}_2$ norm satisfies a separation principle – when the noise takes a certain form, the contribution of the time scales and the contribution of the edge weights are completely decoupled from each other. When considering the design problem – what edge weights and what time scales optimize the $\mathcal{H}_2$ norm –, this means we can design parameters of the agent independently from the parameters of its interconnections (given a constant graph structure).

The last contribution of this part is to examine a notion of complexity of graphs, which we call ‘network entropy’ (although strictly speaking, it is not formally ‘entropy’ in the information-theoretic sense – it is better called pseudoentropy). We show that this notion of entropy encodes some notion of ‘signal mixing’ in consensus – as information flows from agent to agent,
cycles in the network increase the mixing of the signal. This is related to some of the performance measures discussed above, in particular the $H_2$ norm.

INCLUDED PUBLICATIONS


1.4.3 Part 4 – Interaction

In the final part of this thesis, we examine some notions of controlling networked dynamical systems. The original motivation for this work was to look at state-dependent networked dynamical systems – systems in which the graph itself evolves as a function of the underlying process. It turns out that such systems are amenable to controller design if we allow some uncertainty in their states.

In particular, if you start to care about the overall behaviour of the networked system as a whole rather than the individual agent states, you can do something called density control. Here, you assign a density function in space that describes the overall distribution of agents in that space. We then examine notions of feedback and feedforward control, as well as an density
estimation procedure, that allows the agents in a networked dynamical system to coordinate among themselves to distribute themselves according to this target density.

The tool we use is optimal mass transport, or the rigorous study of moving one pile of dirt to another under minimal effort. Jokes aside, consider two measures in some ambient probability space, and then consider all maps $T$ whose pushforward under one measure is the other measure. One can assign a cost to this map by integrating over the initial measure, and then the task is to find the map $T$ which is optimal under this cost.

For quadratic costs and absolutely continuous measures, one can pose an equivalent formulation discussed by Brenier and Benamou that seeks to find an incompressible flow that takes one measure to the other in finite time. The incompressible flow constraint is essentially the continuity equation – or the Liouville equation of a single integrator! For an arbitrary system, one just swaps out the continuity equation with the appropriate Liouville equation, and one gets an optimal control problem as a result. We discuss this in the context of networked dynamical systems.

**Included Publications**


**1.5 Preliminaries**

In this section, we outline the mathematical notation, as well as some elementary algebraic graph theory. Further notation and mathematics will be developed in the subsequent chapters; this section simply provides the universal notation used in every chapter of this thesis.

**1.5.1 Notation**

We consider $\mathbb{R}_+$ and $\mathbb{R}_{++}$ as the sets of nonnegative and positive real numbers, respectively. A column vector with $n$ elements is referred to as $v \in \mathbb{R}^n$ where $v_i$ or $[v]_i$ both represent the $i$th element in $v$. The matrix $M \in \mathbb{R}^{p \times q}$ contains $p$ rows and $q$ columns with $[M]_{ij}$ denoting the element in the $i$th row and $j$th column of $M$. The square matrix $N \in \mathbb{R}^{n \times n}$ is symmetric if $N^T = N$.

The identity matrix, $I_n$, is the diagonal $n \times n$ square matrix with ones on its diagonal and zeros otherwise. For $w \in \mathbb{R}^n$ the diag($w$) is an $n \times n$ matrix with $w$ on its diagonal and zero elsewhere. The unit vector $e_i$ is the column vector with all zero entries except $[e_i]_i = 1$. The column vector of all ones is denoted as $1$. The cardinality of a set $S$ is denoted as $|S|$. The annihilator of a set $S$ is defined as

$$S^\perp = \{v^* \in \mathbb{R}^n : \langle v, v^* \rangle = 0 \text{ for all } v \in S\},$$
where $\langle \cdot, \cdot \rangle$ is the inner product in the Euclidean space. The column space of a matrix $M$ is denoted by $\mathcal{R}(M)$. We define $\mathcal{R}(P)$ to be $A$-invariant if there exists $C$ such that $AP = PC$. We say $A$ is similar to $B$ if there is an invertible matrix $R$ such that $R^{-1}AR = B$; two similar matrices share the same spectral properties. The leading principal submatrix of order $k$ of $X$ is the square submatrix of $X$ formed by deleting the last $n - k$ rows and columns; let $A[K]$ denote the principal submatrix of $A$ obtained by deleting the rows and columns of $A$ corresponding to the elements in the set $[m] \setminus K$. We denote the Moore-Penrose pseudoinverse of a matrix $A$ as $A^\dagger$. The symbol `$\geq$' defines the positive semi-definite ordering: for $n \times n$ PSD matrices $A, B$ we have that

$$A \geq B \iff A - B \succeq 0_{n \times n}.$$ 

The function $h$ is even if $h(-x) = h(x)$ and is odd if $h(-x) = -h(x)$. The function $f$ is of class $C^r$ if the derivatives $f, f', \ldots, f^{(r)}$ exist and are continuous. The function $g \in C^\infty$, otherwise called smooth, has derivatives of all order. Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a vector field and let $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ be a smooth mapping. Then, $F$ is $\varphi$-invariant if $(D\varphi(x))F(x) = F(\varphi(x))$ for all $x \in \mathbb{R}^n$ with $D\varphi(x)$ the Jacobian matrix of $\varphi$ at $x$. The Lie bracket of two vector fields $f$ and $g$ is denoted by $[f, g]$. Given a mapping $\gamma : M \to M$, the fixed point set of $\gamma$ is denoted by $\text{Fix}(\gamma) = \{x \in M | \gamma(x) = x\}$. The operator $\pi_{n-1} : \mathbb{R}^n \to \mathbb{R}^{n-1}$ is defined to be the projection onto the first $(n-1)$-coordinates.

### 1.5.2 Consensus and Algebraic Graph Theory

A multi-agent system with $n$ agents is characterized by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges, and $\mathcal{W} \in \mathbb{R}^{n \times n}$ consists of weights assigned to edges. We call $A \in \mathbb{R}^{n \times n}$ the adjacency matrix where $A_{ij} = W_{ij} \neq 0$. The degree matrix $D \in \mathbb{R}^{n \times n}$ is a square diagonal matrix where $D_{ii} = \sum_{j \in \mathcal{N}(i)} A_{ij}$. The graph Laplacian is then defined as $\mathcal{L}_G = D - A$. Then the consensus dynamics, with $x \in \mathbb{R}^n$ the state vector, is defined as

$$\dot{x}_i = -\sum_{j \in \mathcal{N}_i} W_{ij} (x_i - x_j),$$

or in matrix-vector form, as

$$\dot{x} = -\mathcal{L}_G x.$$

A path of length $r$ in $\mathcal{G}$ is given by a sequence of different nodes $v_{i_0}, v_{i_1}, \ldots, v_{i_r}$ such that for $k = 0, 1, \ldots, r-1$, the nodes $v_{i_k}$ and $v_{i_{k+1}}$ are neighbors. When the terminal nodes are equal, the path is called a cycle [5]. A leaf, or whisker, is a node with degree 1.
Part II

CONTROL
Networked dynamical systems have been the forefront of active research for the past two decades. Some well-studied examples of networked systems are social networks and dynamics of opinions [3], flocking [6], autonomous robotics [7], [8], quantum networks [9], autonomous flight [10], traffic control [11], multi-agent systems [12], [13], and gene networks [14]. Several networked systems include both cooperative and antagonistic interactions, such as certain classes of social dynamics [15], [16].

Consensus algorithms have been used in many scientific and engineering applications, including those listed above. A large amount of research has been dedicated to looking at the control of consensus [5]. The work by Rahmani et al. showed that certain symmetries of networks characterized by automorphisms of the topology of the network cause uncontrollability [17]. This was generalized by Chapman and Mesbahi who showed signed fractional automorphisms generate necessary and sufficient conditions for uncontrollability and unstabilizability of linear systems [18]. Further work examined methods of generating network topologies, for either performance improvements, as in [19] and [20], or those that are controllable for consensus, such as in [21] and [22], [23].

Consensus algorithms on networks with antagonistic interactions were first considered by Altafini [24], [25]. The network property of structural balance, first considered in the study of social networks ([15], [26], [27]) was identified in Altafini’s work as the property inducing bipartite consensus in which the agents converge to two disjoint clusters instead of a uniform consensus. Graph-theoretic properties of signed Laplacian dynamics were studied by Pan et al. [28]. Further research by Pan et al. has looked at identifying the bipartite structure of structurally balanced graphs using data from signed Laplacian dynamics and dynamic mode decomposition [29], adding to the works done by Harary and Kabell [30], and Facchetti et al. [31]. Recent contributions by Clark et al. have studied the leader selection problem in signed consensus [32].

The generalization to nonlinear consensus algorithms has been studied in numerous settings. Behaviour of nonlinear consensus protocols was considered by Srivastava et al. [16]. The extension of these consensus protocols to signed networks was studied by Altafini [24]. Moreover, the generalization of symmetry arguments for controllability was examined by by Aguilar and Gharesifard [33].

The (literal) dual side of control is estimation. One of the most successful estimation algorithms is called the Kalman filter. The Kalman filter is an algorithm that uses the known dynamics of a system to remove noise from measurements of that system. When considering large-scale dynamical systems, implementation of the standard or extended Kalman filters can be computationally difficult. In such cases, the Kalman filter requires the inversion of very large matrices at each timestep. This may cause the Kalman
signed consensus networks

filter to run slower than the dynamical process it is trying to measure, or to severely reduce the temporal resolution of the measurements.

There are many systems for which measurements are taken by a network of sensors and are of high dimension; examples of such systems include weather models and can be found in [34] and [35]. Previous methods for circumventing this problem include decomposing the dynamical system being measured into several subsystems and distributing the subsystems over the sensor network as in [34], or using Monte-Carlo methods for estimating the error covariance, such as in [36]. Previously, Sutton proposed to modify the Kalman filter error covariance update with a gradient descent method for the purpose of minimizing memory consumption, albeit for a specific instantiation of a SISO linear system [37].

Distributed Kalman filtering seeks to estimate the state of a system by distributing the tasks of measuring the system and subsequently filtering the data to many agents, who then collectively assemble the state estimate. Such algorithms utilizing consensus to provide a global estimate were presented by [38] and [39], and later extended by [40]. Performance of distributed Kalman filters using graph-theoretic quantities were studied by [41].

There has been a recent interest in applying data-driven methods to control of networks, for example the Koopman operator. The Koopman operator is a dynamical framework in which one considers the propagation of observables of the state, rather than the state itself. The Koopman operator is linear, even for a non-linear system, but the trade-off is that the vector space of observables is generally infinite dimensional [42]. This formalism lends itself well to a data-driven approach, allowing one to approximate the Koopman operator by collecting data [43]. Research by Pan et al. has looked at identifying the bipartite structure of signed networks using data-driven methods [29], furthering work done by Facchetti et al. [31], and Harary and Kabell [30].

The contributions of the chapters in this part are as follows. We conduct a controllability analysis of signed Laplacian consensus using symmetry arguments developed by Rahmani et al. [17], for the individual single-input-single-output (SISO) and multi-input-multi-output (MIMO) cases of consensus dynamics with leader nodes. In particular, we identify the property of structural balance that when combined with symmetry causes uncontrollability of signed consensus dynamics. The key feature of structurally balanced graphs that allows this analysis is that they admit a \textit{gauge transformation} that allows the permutation matrix corresponding to the graph symmetry to be extended to a signed permutation matrix, which we show leads to uncontrollability when corresponding to a symmetry about input nodes. We then use tools developed by Chapman and Mesbahi in [18] to derive controllability and stabilizability conditions for influenced signed consensus dynamics.

Next, we show that the property of structural balance, when combined with symmetries in the underlying graph, as well as certain symmetries of the nonlinear dynamics, causes uncontrollability in the context of the accessibility problem. In particular, we consider the same network flows studied in [16], [24], [33]; however we extend the controllability analysis to signed dynamics. Subsequently, we extend the bipartite identification problem considered by Pan et al. in [29] to the case of signed nonlinear consensus net-
works. In particular, we use a Koopman operator-theoretic approach alongside Extended Dynamic Mode Decomposition (EDMD) to extract a ‘Koopman mode’ whose sign structure reveals the bipartite structure.

Lastly, we examine estimation problems on large sensor networks. In particular, we extend the gradient descent algorithm for estimating the Kalman filter error covariance to the general MIMO linear system as a proposed solution to the problem of running a Kalman filter on a high-dimensional system. We improve the gradient descent using Nesterov acceleration and adaptive learning rate methods. We apply the methods above to distributed Kalman filtering on a sensor network.

This part is organized as follows. We outline the problem statements in §3.1. In §3.2, we show that structural balance combined with symmetry about inputs leads to uncontrollability, and derive the corresponding stabilizability conditions. The nonlinear case is discussed in §3.3.

In Chapter 4, we outline the basics of Koopman operator theory, and how we apply it for community detection in nonlinear structurally-balanced networked dynamical systems. We discuss distributed Kalman filtering in §6.1. Relevant examples are shown in Chapter 5, and the paper is concluded in §6.5.

2.1 Mathematical Background

2.1.1 Signed Networks

A signed graph $\mathcal{G}_s$ is a graph that admits negative weights. We define the signed graph Laplacian as $L_{\mathcal{G}_s} = D_s - A_s$, where $[A_s]_{ij} = \pm A_{ij}$, and where the degree matrix is given by $D_{ii} = \sum_{j \in N(i)} |A_{ij}| = \sum_{j \in N(i)} |W_{ij}|$. The generalization of consensus dynamics to signed graphs was introduced in [25], and the corresponding dynamical system is given by

$$\dot{x}_i = -\sum_{j \in N(i)} [W_{ij}] (x_i - \text{sgn}(W_{ij})x_j),$$

where sgn represents the sign function. A gauge transformation is a change of orthant order via a matrix $G_t \in \text{diag}(\sigma) : \sigma = [\sigma_1, \ldots, \sigma_n], \sigma_i = \pm 1$. We know $G_t = G_t^T = G_t^{-1}$. Let the gauge-transformed Laplacian be given by $L_{\mathcal{G}_s} = G_t L_{\mathcal{G}_s} G_t = D - G_tA_s G_t$, where

$$(L_{\mathcal{G}_s})_{ij} = \begin{cases} \sum_{k \in N(i)} |A_{ik}| & j = i \\ -\sigma_i \sigma_j A_{ij} & j \neq i. \end{cases}$$

2.1.2 Automorphisms/Interlacing/Equitable Partitions

An automorphism of the graph $\mathcal{G}$ is a permutation $\psi$ of its nodes such that $\psi(i) \psi(j) \in E$ if and only if $ij \in E$. Let the permutation matrix $\Psi$ be such that $[\Psi]_{ij} = 1$ if $\psi(i) = j$ and zero otherwise. Then $\psi$ is an automorphism of $\mathcal{G}$ if and only if $\Psi A(\mathcal{G}) = A(\mathcal{G}) \Psi$ (see [5]). The permutation $\psi$ induces a mapping $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ such that $[\varphi(x)]_i = x_{\psi(i)}$. Let the permutation
matrix $J$ be such that $[J]_{ij} = 1$ if $\psi(i) = j$ and zero otherwise. Therefore, the permutation matrix $J$ is simply the Jacobian matrix of $\varphi$, in that $J = D\varphi$.

Suppose $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times m}$ are both symmetric and $m \leq n$. Then the eigenvalues of $B$ interlace the eigenvalues of $A$ if for $i = 1, 2, \ldots, m$, $\lambda_{n-m+i}(A) \leq \lambda_i(B) \leq \lambda_i(A)$ where $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ are the eigenvalues of $A$ in a non-increasing order [44].

In a similar way that we defined the functions $\phi$ and $\varphi$ for the graph automorphism, consider the function $g : \mathcal{V} \to \mathcal{V}$ encoding the action of the gauge on the nodes. This induces a function $\mathcal{g} : \mathbb{R}^n \to \mathbb{R}^n$ such that $[\mathcal{g}(x)]_i = \sigma_i x_i$. The gauge transformation is then the Jacobian of this function, in that $G_t = D\mathcal{g}$.

The cell $C$ is a subset of the graph nodes $\mathcal{V}$. A nontrivial cell is a cell with more than one node. A partition is a grouping of $\mathcal{V}$ into different cells. An $r$-partition $\pi$ of $\mathcal{V}$ with cells $\{C_i\}_{i=1}^r$ is equitable if each node in $C_i$ has the same number of neighbors in $C_j$, for all $i, j$. We call $\pi$ a nontrivial equitable partition (NEP) if it contains at least one nontrivial cell. Let $b_{ij}$ be the number of neighbors of a node in $C_i$ of a node in $C_j$. The quotient of $G$ over $\pi$, denoted by $G/\pi$, is the directed graph with the cells of an equitable $r$-partition $\pi$ as its nodes and $b_{ij}$ edges directed from $C_i$ to $C_j$. The adjacency matrix of the quotient is specified by $[A(G/\pi)]_{ij} = b_{ij}$. A characteristic vector $p_i \in \mathbb{R}^n$ of a cell $C_i$ has 1’s in components associated with $C_i$ and 0’s elsewhere. A characteristic matrix $P \in \mathbb{R}^{n \times r}$ of a partition $\pi$ of $\mathcal{V}$ is defined as $[p_i]_i^{r=1}$.

For example, for the equitable partition in figure 2.1a we get

$$A(G/\pi) = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 1 & 1 \\ 0 & 2 & 0 \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}. $$

![Figure 2.1](a) Equitable partition of a sample graph, (b) quotient of the graph in (a)
3

SIGNED CONTROL

3.1 PROBLEM STATEMENT

The analysis of this paper consists of several main parts, which examine two different notions of influencing control on networks, as well as linear and nonlinear controllability. In the first part, we examine the notion of uncontrollability due to symmetry and interlacing which was initially derived in [17] for signed consensus networks. The notion of control in this case is taking over the state of one or several nodes in the network, and using their edges to inject signals into the system. In the second part, we consider the case where the nodes are controlled by injecting a single-integrator signal to some nodes of the graph.

Signed consensus networks are of interest because the negative weights induce a phenomenon known as clustering, where agents will not converge to an agreement subspace, but rather converge to opposite equilibria (an example of clustering is shown in Section 5.1). The condition that causes clustering was identified by [24] as structural balance. Surprisingly, we show that this topological feature is exactly the condition that produces uncontrollability of signed networks.

Lemma 1. (See [24]) The following statements are equivalent:

1. The signed graph \( G \) is structurally balanced;
2. There exists a gauge \( G^t \) such that \( G^t A G^t \) has only positive entries (i.e. the transformed graph becomes unsigned);
3. All cycles in \( G \) are positive;
4. The signed Laplacian \( L_{Gs} \) has a zero eigenvalue;
5. There exists a bipartition of \( V \) such that the edge weights on the edges within the same set are positive, and the edges connecting the two sets are negative.

One may recall that the unsigned consensus dynamics have an agreement subspace spanned by the eigenvector 1 corresponding to the zero eigenvalue. For signed consensus, the zero eigenvalue explicitly corresponds to disagreement. Then one may think that structural balance is therefore not a desirable quality for signed consensus, but it turns out that if the graph is not structurally balanced, the consensus dynamics become trivial in some sense, and converge to zero.

Theorem 1. (See [24]) If \( L_{Gs} \) is structurally unbalanced in a signed graph \( G_s \), then \( \lim_{t \to \infty} x(t) = 0 \) where \( x \in \mathbb{R}^n \) is the state vector with the corresponding dynamics \( \dot{x} = -L_{Gs} x \). Otherwise, if \( L_{Gs} \) is structurally balanced, then \( \lim_{t \to \infty} x(t) = (1/n) (1^T G_s x(0)) G_s I \).
3.1 Problem Statement

Previous work by Rahmani et al. in [17] showed that symmetry with respect to a single input and interlacing for multiple input are sufficient for uncontrollability, and this was generalized by Chapman and Mesbahi in [18] to show that fractional symmetry with respect to the inputs is sufficient and necessary for uncontrollability.

In this paper, we aim to show that structural balance is the property that combined with symmetry and interlacing leads to uncontrollability. There are examples of signed networks that are input symmetric but controllable, and we will discuss these in Section 5.3. Before proceeding to our main results, we summarize the various dynamics considered in the paper.

There are several variants of consensus dynamics with respect to how inputs are injected into nodes. Given a connected signed graph \( G_s \), we can select one node and use that specific node to inject our input signal \( u \). This corresponds to partitioning the Laplacian as follows (see, for example, [5]):

\[
\mathcal{L}_{G_s} = \begin{bmatrix}
A^f_s & B^f_s \\
B^f_s^T & A^i_s
\end{bmatrix},
\]

where \( f \) and \( i \) denote the floating and input parts of the network respectively. Then, the dynamical system for signed consensus networks is

\[
\dot{x} = -A^f_s x - B^f_s u,
\]

where for a single-input-single-output (SISO) system we have \( A^f_s \in \mathbb{R}^{(n-1) \times (n-1)} \), \( B^f_s \in \mathbb{R}^{(n-1) \times 1} \), and \( A^i_s \) is a scalar. Similarly, one can define the same dynamics as (3.1) for the multiple-input-multiple-output (MIMO) system. In that case, \( A^f_s \in \mathbb{R}^{n_f \times n_f} \) and \( B^f_s \in \mathbb{R}^{n_f \times n_i} \) where \( n_f \) and \( n_i \) are the numbers of the nodes in floating and input subgraphs respectively. The floating signed graph is denoted by \( G^f_s \).

**Lemma 2.** [Popov-Belevitch-Hautus (PBH) Test] The system described in (3.1) is controllable if and only if none of the eigenvectors of \( A^f_s \) are simultaneously orthogonal to all columns of \( B^f_s \).

We use lemma 2 further to discuss controllability of the SISO case in Section 3.2.1.1, and the MIMO case in Section 3.2.1.2.

The second variant of controlling consensus networks is to simply inject signals into nodes, without taking over the state of the node. We can therefore define the influenced signed consensus dynamics with \( q \) inputs and \( p \) outputs as

\[
\dot{x} = -\mathcal{L}_{G_s} x + B(I)u, \quad y = C(O)x,
\]

(3.2)
where $I \subseteq N$ is the set of input nodes, $O \subseteq N$ the set of output nodes, and $B(I), C(I)$ are the matrices

$$B(I) = \begin{bmatrix} e_{i_1} & \cdots & e_{i_q} \end{bmatrix}, \quad C(O) = \begin{bmatrix} f_{j_1} & \cdots & f_{j_p} \end{bmatrix}^T,$$

in which $e_i$ is the indicator vector for nodes $i \in I$, and $f_j$ for nodes $j \in O$. The control signal vector is $u \in \mathbb{R}^q$. Both (output) controllability and (output) stabilizability of these dynamics is considered in Section 3.2.2.

For examining nonlinear consensus, we follow the same conventions as in [33]. Consider the controlled dynamical system

$$\dot{x} = F(x, u) \quad (3.3)$$

where $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a smooth mapping. The accessible set $\mathcal{A}(x_0, T)$ of the system (3.3) from $x_0$ at time $T$ is the set of all end-points $\theta(T)$ where $\theta : [0, T] \to \mathbb{R}^n$ is a trajectory of (3.3). The accessible set of (3.3) from $x_0$ up to $T$ is defined as $\mathcal{A}(x_0, \leq T) := \cup_{0 \leq \tau \leq T} \mathcal{A}(x_0, \tau)$. Then, the nonlinear dynamical system (3.3) is said to be accessible from the initial point $x_0$ if for every $T > 0$ the set $\mathcal{A}(x_0, \leq T)$ contains a non-empty interior.

### 3.2 Methods

#### 3.2.1 Signed Graph Controllability

##### 3.2.1.1 The SISO Case

A standard result by [17] shows that for a SISO consensus network, a symmetry about the input node is sufficient for uncontrollability. We extend this result for the signed consensus networks considered by [24] and [29]. In particular, we show that structural balance and input symmetry for the unsigned graph is sufficient for uncontrollability.

**Remark 1.** This result shows signed Laplacian is in some sense more robust to symmetries about the input nodes. In particular, there are examples of unsigned graphs that are symmetric about an input and therefore uncontrollable, but whose signed counterparts exhibit controllability despite the symmetry. Therefore, there is not enough conditions to claim the uncontrollability of the system.

Now, we state and prove a lemma which helps us provide the main result of this section.

**Lemma 3.** Assume the unsigned graph $\mathcal{G}$ enjoys input symmetry and the signed network $\mathcal{G}_s$ is structurally balanced. Then, the following statements hold

1. There exists $J'$ such that $J' \mathcal{L}_s = \mathcal{L}_s J'$
2. For the same \( J' \) as part 1, \( J'^TB'_s = B'_s \)

3. If \( v \) is the eigenvector corresponding to the eigenvalue \( \lambda \) of \( A'_s \), Then, \( J'v \)
   (and hence \( v - J'v \)) would also be the eigenvectors corresponding to \( \lambda \)

**Proof.** From the second statement of lemma 1 there exists \( G_t \) such that

\[
G_tL_{\mathcal{G}_s}G_t = L_{\mathcal{G}},
\]

(3.4)

and hence

\[
A'_s = G' A' G' \quad \text{and} \quad B'_s = \sigma_n G' B' \]

It follows

\[
G_t \begin{bmatrix} A'_s & B'_s \\ B'_s^T & A'_s \end{bmatrix} G_t = \begin{bmatrix} G' A'_s G' & G' B'_s \sigma_n \\ \sigma_n B'_s^T G' & \sigma_n A'_s \sigma_n \end{bmatrix} = \begin{bmatrix} A' & B' \\ B'^T & A' \end{bmatrix},
\]

where \( G' = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{n-1}) \). Without loss of generality, we can assume \( \sigma_n = 1 \) since (3.4) also holds for \(-G\). This gives

\[
B' = G' B'_s.
\]

From [17] we know that if \( \mathcal{G} \) has the input symmetry structure, then there exists the permutation matrix \( J \) such that

\[
JA' = A' J.
\]

Let \( J' = G' J G' \),

\[
J' A'_s = G' J G' A' G' = G' J A' G' = G' A' J G' = G' A' G' J G' = A'_s J',
\]

which proves part (1).

We also have

\[
J'^TB'_s = J'^T G' B' = G' J'^T G' B' = G' J'^T B' = -G' J'^T A' = -G' A' J'^T 1 = B'_s,
\]

which gives the result of part (2).

For the last part of the proof by definition we know \( A'_s v = \lambda v \). Then

\[
A'_s J' v = J' A'_s v = \lambda J' v,
\]
implying that \( J'v \) is also an eigenvector for the same eigenvalue. Hence, assuming orthonormal eigenvectors for \( A'_s \), then \( v - J'v \) would also be an eigenvector corresponding to \( \lambda \). □

The matrix \( J' \) introduced for a signed floating graph in lemma 2 is in some sense correspondent to the permutation matrix \( J \) in the unsigned case. In fact, the only difference between \( J \) and \( J' \) are the elements with the same rows or columns as the negative elements in \( G' \). For example, for the graph of Figure 3.1, with node 5 as the input node, the matrices \( J, J' \), and \( G' \) are found to be

![Figure 3.1: A network topology with input symmetry](image)

\[
J = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}, \quad
G' = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & -1
\end{bmatrix}
\]

\[
J' = \begin{bmatrix}
0 & 0 & 0 & -1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0
\end{bmatrix}.
\]

Therefore, the first two parts of lemma 3 demonstrate the correlation between the signed and unsigned consensus networks using the gauge transformation \( G_t \).

Now we have all the tools we need to provide the main theorem of this part.

**Theorem 2.** The signed network system \( G_s \) is uncontrollable if it is input symmetric and structurally balanced

*Proof.* We use the results of lemma 3 and the PBH test to show the uncontrollability of \( G_s \).

Let \((\lambda, v)\) be a pair of eigenvalue and eigenvector for \( A'_s \) so that \( A'_s v = \lambda v \). Then, from part 3 of lemma 3 we know that \( v - J'v \) is also an eigenvector for \( A'_s \). Then, from part 2 of lemma 3 we have

\[
(v - J'v)^T B'_s = v^T B'_s - v^T J'^T B'_s = v^T B'_s - v^T B'_s = 0,
\]
which implies that the system is not controllable according to the PBH test of controllability.

**Remark 2.** A signed symmetry implies the existence an unsigned symmetry of $G$. The converse is true when $G$ is structurally balanced.

### 3.2.1.2 The MIMO Case

In this section, we examine how the notion of structural balance is interposed in the controllability analysis of multiple input signed networks. The results in this section are extensions to [17]. To this end, we leverage the machinery of interlacing and equitable partitions on graphs.

First, we restate and modify two fundamental lemmas from [44] to the signed case and then provide the analysis which leads to sufficient conditions on the uncontrollability of the system.

**Definition 1.** Let $G_t$ be the gauge transformation as in (3.4). Then, $P'$ is the signed characteristic matrix defined as $P' = G_tP$.

**Lemma 4.** Let $\pi$ be a partition of the structurally balanced signed graph $G_s$, with adjacency matrix $A_s$ and signed characteristic matrix $P'$. Then, $\pi$ is equitable if and only if the column space of $P'$ is $A_s$-invariant.

**Proof.** (Necessity) assume $\pi$ is equitable. From lemma 9.3.1 in [44] if $\pi$ is equitable then $AP = P\hat{A}$ with $\hat{A} = A(G_s/\pi)$. Then, it follows from the second statement in lemma 1

$$P\hat{A} = AP = G_tA_sG_tP \quad \Rightarrow \quad A_sP' = P'\hat{A}.$$

(Sufficiency) From lemma 9.3.2 in [44], $\pi$ is equitable if there exists $B$ such that $AP = PB$. Then, if such $B$ exists, we get

$$PB = AP = G_tA_sG_tP \quad \Rightarrow \quad A_sP' = P'B.$$

**Lemma 4** shows how the gauge transformation is injected into the analysis of signed networks. Indeed, based on the knowledge of the second statement in lemma 1, we use the structural balance to turn the signed system into the extensively developed un-signed consensus dynamics.

The next lemmas are provided for the sake of completeness.

**Lemma 5.** (See [17]) Given a symmetric matrix $A \in \mathbb{R}^{n \times n}$, let $S$ be a subspace of $\mathbb{R}^n$. Then, $S^\perp$ is $A$-invariant if and only if $S$ is $A$-invariant.
**Lemma 6.** (See [44] Theorem 9.5.1) Let $\Phi \in \mathbb{R}^{n \times n}$ be a real symmetric matrix, and let $R \in \mathbb{R}^{n \times m}$ be such that $R^T R = I_m$. Set $\Theta = R^T \Phi R$ and let $v_1, v_2, \ldots, v_m$ be an orthogonal set of eigenvectors for $\Theta$ such that $\Theta v_i = \lambda_i(\Theta) v_i$, where $\lambda_i(\Theta) \in \mathbb{R}$ is an eigenvalue of $\Theta$. Then, $\Phi R = R \Theta$ if the interlacing between $\Phi$ and $\Theta$ is tight.

**Remark 3.** As discussed in [17], we can now find an orthogonal decomposition of $\mathbb{R}^n$ using the signed characteristic matrix $P'$ as

$$\mathbb{R}^n = \mathcal{R}(P') \oplus \mathcal{R}(Q'),$$

where $\mathcal{R}(Q') = \mathcal{R}(P')^\perp$. Then, an orthonormal basis for $\mathbb{R}^n$ can be formed as

$$T = \begin{bmatrix} P' & Q' \end{bmatrix},$$

(3.5)

where $P'$ and $Q'$ represent normalized $P'$ and $Q'$ respectively and satisfy $P'^T Q' = 0$ and $Q'^T Q' = I_{n-r}$.

**Lemma 7.** Given a connected signed graph $G_s$, the system (3.1) is uncontrollable if and only if $L_{G_s}$ and $A_s$ share at least one common eigenvalue.

Lemma 7 is a derivation from lemma 7.9 in [17]. Since this is a general result depending on the Laplacian and its leading principal submatrix (floating graph), and the PBH test, the same holds for the signed case.

From this point, the goal is to show that for some specific graph partition and the structural balance of the network, $L_s$ and $A_s^f$ share similar eigenvalues which leads to uncontrollability. The next two lemmas assert that $L_{G_s}$ and $A_s^f$ are similar to some block diagonal matrices.

**Lemma 8.** Suppose a structurally balanced signed graph $G_s$ has an NEP $\pi$ with $P'$ and $Q'$ as in (3.5). Then, the signed Laplacian $L_{G_s}$ is similar to the block diagonal matrix

$$L_{G_s} = \begin{bmatrix} L_{P'} & 0 \\ 0 & L_{Q'} \end{bmatrix},$$

where $L_{P'} = P'^T L_{G_s} P'$ and $L_{Q'} = Q'^T L_{G_s} Q'$.

**Lemma 9.** Let $G_s^f$ be a signed floating graph, and $A_s^f$ be defined as in (3.1) and $P'$ and $Q'$ be as in (3.5). If there exists an NEP $\pi_f$ in $G_s^f$ and a $\pi$ in the original structurally balanced signed graph $G_s$ such that all the nontrivial cells in $\pi_f$ are also cells in $\pi$, then $A_s^f$ is similar to the block diagonal matrix

$$A_s^f = \begin{bmatrix} A_{P'} & 0 \\ 0 & A_{Q'} \end{bmatrix},$$

with $A_{P'} = P'^T A_s^f P'$ and $A_{Q'} = Q'^T A_s^f Q'$. 

The proofs are similar to lemmas 7.11, 7.12, and 7.14 in [17] and is skipped for succinctness. One just needs to consider the role of structural balance and the fact that the signed characteristic matrix $P'$ needs to be replaced for $P$ due to the insertion of gauge transformation.

We are now well-equipped to address the main result of the section. In the following theorem, we see that in certain circumstances, the two block diagonal matrices share identical blocks and thus’ common eigenvalues. This leads to the uncontrollability of the system by lemma 7.

**Theorem 3.** Given a connected structurally balanced signed graph $G_s$ with the floating graph $G^f_s$, the system (3.1) is uncontrollable if there exist NEPs on $G_s$ and $G^f_s$, $\pi$ and $\pi_f$, such that $\pi_f$ contains all nontrivial cells of $\pi$.

The main scheme of the proof is similar to theorem 7.15 in [17]. We repeat the proof to show how the new orthogonal basis formed by $P'$ and $Q'$ work in the new setup of signed networks.

**Proof.** As a result of structural balance, let $P'$ and $Q'$ be defined as in (3.5). Following the convention in [17], let $\pi \cap \pi_f = \{C_1, C_2, \ldots, C_n\}$ with $|C_i| \geq 2$, $i = 1, 2, \ldots, r_1$. Let the nontrivial cells contain the first $n_1$ nodes. Since $\pi_f$ contains all nontrivial cells of $\pi$, it follows

\[
P' = \begin{bmatrix} P'_1 & 0 \\ 0 & I_{n-n_1} \end{bmatrix}_{n \times r} \quad \text{and} \quad P'_f = \begin{bmatrix} P'_1 & 0 \\ 0 & I_{n_f-n_1} \end{bmatrix}_{n_f \times r_f},
\]

where $P'_1 \in \mathbb{R}^{n_1 \times r_1}$ contains the nontrivial part of the signed characteristic matrices. Let $P'$ and $P'_f$ be the normalization of $P'$ and $P'_f$ and define $Q'$ and $Q'_f$ as in (3.5). Then

\[
Q' = \begin{bmatrix} Q'_1 \\ 0 \end{bmatrix}_{n \times (n_1-r_1)}, \quad Q'_f = \begin{bmatrix} Q'_1 \\ 0 \end{bmatrix}_{n_f \times (n_1-r_1)},
\]

where $Q'_1 \in \mathbb{R}^{n_1 \times (n_1-r_1)}$ satisfies $Q'_1 P_1 = 0$. It follows that $Q'_f = R^T Q'$ with $R = [I_{n_f}, 0]^T$. Then, by lemmas 8 and 9, we get

\[
L_{Q'} = Q'^T L_{G_s} Q' = Q'^T R^T L_{G^f_s} R Q'_f = Q'^T A'^s Q'_f = \mathcal{A}'_{Q'}
\]

This implies that $L_{G_s}$ and $A'^s$ share a block matrix and thus have at least one equal eigenvalue. Therefore, by lemma 7 the system is uncontrollable. $\square$

Theorem 3 gives sufficient conditions for uncontrollability of the signed network system. However, this by no means is a necessary condition. e.g. a system can be uncontrollable and structurally unbalanced simultaneously.
3.2.2 Stabilizability and Output Controllability

Recent developments in controllability have extended the idea of using symmetry to characterize controllability of linear systems.

**Theorem 4** (Symmetry Controllability Test [18]). Consider the general linear dynamics

\[ \dot{x} = Ax + Bu, \quad y = Cx, \]

with \( A \in \mathbb{R}^{n \times n} \), \( B \in \mathbb{R}^{n \times m} \) and \( C \in \mathbb{R}^{p \times n} \). For \( A \) diagonalizable and \( C \) full row rank, consider the conditions on a square matrix \( P \):

a. \( P \neq I \), \( AP = PA \) and \( PB = B \)

b. \( CP = ZC \) for some \( Z \)

c. \( \frac{1}{2}(P + P^T) \preceq I \) and \( PA + (PA)^T \preceq A + A^T \)

Then, there exists \( P \in \mathbb{C}^{n \times n} \) such that

1. \( (a) \iff (A, B) \) uncontrollable

2. \( (a) \& (b) \iff (A, B, C) \) is output uncontrollable

3. \( (a) \& (c) \iff (A, B) \) is unstabilizable

4. \( (a) \& (b) \& (c) \iff (A, B, C) \) is output unstabilizable

Using Theorem 2 in [18], we can now characterize the controllability, output controllability, stabilizability and output stabilizability of the influenced signed consensus dynamics

\[ \dot{x} = -L_{G_s} x + B(I)u, \quad y = C(O)x, \]

where \( I \subseteq N \) is the set of input nodes, \( O \subseteq N \) the set of output nodes, and \( B(I), C(I) \) are the matrices

\[ B(I) = \begin{bmatrix} \begin{array}{ccc} e_{i_1} & \cdots & e_{i_q} \\ \end{array} \end{bmatrix}, \quad C(O) = \begin{bmatrix} \begin{array}{ccc} f_{j_1} & \cdots & f_{j_p} \\ \end{array} \end{bmatrix}^T, \]

in which \( e_i \) is the indicator vector for nodes \( i \in I \), and \( f_j \) for nodes \( j \in O \).

Our main result is the following theorem, which identifies structural balance as the key feature for uncontrollability of the dynamics (3.2), on top of symmetry of the underlying unsigned graph. The crux of the argument is that the gauge transformation allows a signed symmetric automorphism to satisfy the criteria in Theorem 2 of [18], allowing a variant of the result in Corollary 5 of [18].

**Theorem 5.** Let \( J \) be a non-trivial signed fractional automorphism of \( L_G \). Suppose further that \( L_{G_s} \) is structurally balanced with gauge \( G_t \). Consider dynamics (3.2), and the following conditions on \( J_s := G_t J G_t \) with \( B_s := G_t B(I) \) and \( C_s := C(O) G_t \).
3.2 Methods

a. \( J_s B_s = B_s \)

b. \( C_s(R)J_s C_s((V \setminus R)^\top) = 0, \ C_s(R)J_s C_s(R)^\top = Z \neq I \).

c. \( J_s v_i = v_i \) for all \( v_i \sim \lambda_i(L_{\mathcal{G}_s}) > 0 \).

Then,

1. \( (a) \iff (-L_{\mathcal{G}_s}, B) \) uncontrollable

2. \( (a) \& (b) \iff (-L_{\mathcal{G}_s}, B, C) \) is output uncontrollable

3. \( (a) \& (c) \iff (-L_{\mathcal{G}_s}, B, C) \) is output unstabilizable

4. \( (a) \& (b) \& (c) \iff (-L_{\mathcal{G}_s}, B, C) \) is output unstabilizable

The results hold when \( B \rightarrow B_s \) or \( C \rightarrow C_s \).

The proof of Theorem 5 is similar to the proof of Corollary 5 in [18], with several technical differences which we discuss here. In particular, our result only requires a fractional automorphism \( J \) of the underlying unsigned graph; the matrix \( J_s \) does not need to be a signed fractional automorphism, which in this sense generalizes Corollary 5 of [18].

The following lemmas establish the equivalence of controllability under a gauge transformation of the \( B \) and \( C \) matrices for structurally balanced \( L_{\mathcal{G}_s} \), and some useful identities that will elucidate the role of the gauge transformation in Theorem 5.

**Lemma 10.** Let \( (L_{\mathcal{G}_s}, B(I)) \) be the pair in the dynamics (3.2), and let \( B_s(I) = G_t B(I) \) for any gauge transformation \( G_t \) (regardless of whether \( L_{\mathcal{G}_s} \) is structurally balanced). Then, \( (-L_{\mathcal{G}_s}, B(I)) \) is controllable if and only if \( (-L_{\mathcal{G}_s}, B_s(I)) \) is controllable.

Furthermore, letting \( C_s(O) = C(O)G_t \), we have that \( (-L_{\mathcal{G}_s}, B(I), C(O)) \) is output controllable if and only if \( (-L_{\mathcal{G}_s}, B_s(I), C_s(O)) \) is output controllable.

**Lemma 11.** Consider the dynamics in (3.2). Suppose \( L_{\mathcal{G}_s} \) is structurally balanced with gauge \( G_t \).

1. Suppose that there is an automorphism \( J \) such that \( J L_{\mathcal{G}_s} = L_{\mathcal{G}_s} J \). Then, \( J_s L_{G_t} = L_{G_t} J_s \), where \( J_s = G_t J G_t \).

2. Suppose \( J \) is input symmetric. Then, \( J_s B_s = B_s \).

3. Suppose that there exists \( Z \neq I \) such that \( Z C(O) = C(O)J \). Then, \( Z C_s(O) = C(O)J_s \).

**Proof.** By Lemma 10, \( J_s B_s = B_s \) is equivalent to Theorem 4(a).
Suppose that there exists \( Z \neq I \) such that \( ZC(O) = C(O)J \), to establish condition Theorem 4(b). Note that \( C(O)C(O)^T = I \) and \( C(O)C(N \setminus O)^T = 0 \), and \( [C(O)^T, C(N \setminus O)^T] \) is unitary. We can compute

\[
0 = C(O)J - ZC(N \setminus O) = (C(O)J - ZC(N \setminus O))[C(O)^T, C(N \setminus O)^T] = [C(O)JC(O)^T - ZC(R)C(R)^T, C(O)JC(N \setminus O)^T - ZC(R)C(N \setminus O)^T],
\]

yielding \( C(R)JC(N \setminus R)^T = 0, C(R)JC(R)^T = Z \neq I \). By Lemma 11(3), this is equivalent if we interchange \( J \to J_1 \) and \( C \to C_s \).

Now, let’s assume 5(a), and assume that

\[
-J_sL_{G_s} - (L_{G_s}J_s)^T \geq -L_{G_s} - L_{G_s}^T
\]

in order to establish Theorem 4(c). Since \((J_s + J_s^T)/2 \leq I\), we can see that

\[
\frac{\lambda_i(L_{G_s})}{2} v_i^T (J_s + J_s^T) v_i \geq v_i^T L_{G_s} v_i = \lambda_i(L_{G_s}),
\]

with equality if \( J_s v_i = v_i \), and hence \( \lambda_i(L_{G_s}) > 0 \), which holds for the stable modes of \(-L_{G_s}\). \( \square \)

We now prove Lemma 10.

**Proof.** Note that the column space of the controllability matrix of \((-L_{G_s}, B)\)

\[
C(B(I)) := \begin{bmatrix} B & -L_{G_s}B & \cdots & (-L_{G_s})^{n-1}B \end{bmatrix}
\]

is spanned by columns of the form \((-L_{G_s})^m e_i\) for \(0 \leq m \leq n-1\). The action of a gauge \( G_t \) on \( B(I) \) is to multiply each column of \( B(I) \) by \( \pm 1 \), and so the column space of the controllability matrix of \((-L_{G_s}, B_s)\)

\[
C(B(I)) := \begin{bmatrix} G_tB & \cdots & (-L_{G_s})^{n-1}G_tB \end{bmatrix}
\]

is spanned by columns of the form \( \sigma_i(-L_{G_s})^m e_i \) for \(0 \leq m \leq n-1\), and \( \sigma_i = \pm 1 \). Clearly,

\[
\text{span}\{(-L_{G_s})^m e_i\} = \text{span}\{\sigma_i(-L_{G_s})^m e_i\}
\]

and so \( \text{rank}[C(B(I))] = \text{rank}[C(B_s(I))] \). The same argument applies for output controllability, considering the output controllability matrix

\[
C(B(I), C(O)) := \begin{bmatrix} CB & \cdots & C(-L_{G_s})^{n-1}B \end{bmatrix}.
\]

\( \square \)

We now prove Lemma 11.
3.3 Nonlinear Controllability of Signed Networks

Proof.

\[ J_s \mathcal{L}_{G_s} = G_t J_t G_t \mathcal{L}_{G_t} \]
\[ = G_t J_t \mathcal{L}_{G_t} \]
\[ = G_t \mathcal{L}_{G_t} J_t \]
\[ = G_t \mathcal{L}_{G_t} G_t J_t \]
\[ = \mathcal{L}_{G_s} J_s. \]

We know \( JB = B \). Hence, \( J_s B_s = G_t J_t G_s B = G_s J_s = B_s. \)

\[ ZC(O)G_t = C(O)J_t G_t = C(O)G_t J_t G_t = C_s(O)J_s \]

\[ \square \]

Using the equivalences established in these two lemmas, the proof of Theorem 5 follows as the proof of Corollary 5 in [18], but using \( J_s \) instead of the signed fractional automorphism \( P. \)

3.3 Nonlinear Controllability of Signed Networks

In this section, we extend previous work [33] to analyze the controllability of nonlinear consensus protocols to the case where these protocols run on a signed network. We consider three types of nonlinear consensus protocols, following the nomenclature in [16], [24], [33].

- **Absolute Nonlinear Flow**
  \[ \dot{x}_i = -\sum_{j \in N_i} [f(x_i) - \text{sgn}(a_{ij})f(x_j)] \]  \hspace{1cm} (3.6)

- **Relative Nonlinear Flow**
  \[ \dot{x}_i = -\sum_{j \in N_i} f(x_i - \text{sgn}(a_{ij})x_j) \]  \hspace{1cm} (3.7)

- **Disagreement Nonlinear Flow**
  \[ \dot{x}_i = -f \left( \sum_{j \in N_i} x_i - \text{sgn}(a_{ij})x_j \right) \]  \hspace{1cm} (3.8)

To make this section self-contained, we provide two main theorems from [33] which we use later to demonstrate uncontrollability.

**Theorem 6.** Let \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) and \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \) be a flow on \( \mathcal{G} \). Assume \( \phi \) is a non-identity symmetry on \( F \). Then, for any leader \( l \), the leader-follower network flow on \( \mathcal{G} \) induced by \( l \) is not accessible from the origin in \( \mathbb{R}^{n-1} \).

**Theorem 7.** Let \( \mathcal{G} = (\mathcal{V}, \mathcal{E}) \) and let \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \) be the dynamics in any of (3.6)-(3.8). Also, assume \( \phi \) be an automorphism of \( \mathcal{G} \). Then \( F \) is \( \phi \)-invariant.
Remark 4. The more general case of (3.6)-(3.8) holds when each node has its own smooth nonlinear function $f_i : \mathbb{R} \to \mathbb{R}$. However, as shown in [33], $f_i = f_{\phi(i)}$ is a necessary and sufficient condition for all the subsequent controllability analysis to work. Since we assume the general function $f$ for all nodes, this condition is automatically satisfied.

The behavior of the dynamics (3.6)-(3.8) clearly depends on the choice of $f : \mathbb{R} \to \mathbb{R}$. In [24], several classes of functions were considered. First, the class of translated positive, infinite sector nonlinearities $S$ is defined as

$$S := \left\{ f : [f(x) - f(x^+)](x - x^+) > 0 \text{ for } x \neq x^+, f(0) = 0, \int_{x^+}^x f(t)dt \to \infty \text{ as } |x| \to \infty \right\}.$$  

See [27] for properties of this class of functions. A subset $S_0 \subset S$ of these functions that will be used later is the untranslated ($x^+ = 0$) positive, infinite sector nonlinearities given by

$$S_0 := \left\{ f : f(x^+)x > 0 \text{ for } x \neq 0, f(0) = 0, \int_0^x f(t)dt \to \infty \text{ as } |x| \to \infty \right\}.$$  

The reason these classes of functions are interesting is that when combined with the dynamics introduced in (3.6)-(3.8), clustering occurs in a structurally balanced graph. This is summarized in the following theorem.

Theorem 8. (Theorems 3 & 4 in [24]) Consider a graph $G$. Assume either the dynamics (3.6) with $f \in S$ or the dynamics (3.7) with $f \in S_0$ running on $G$. Then,

$$\lim_{t \to \infty} x(t) = \frac{1}{n} \left( 1^T G_1 x(0) \right) G_1 1$$

if and only if $G$ is structurally balanced (with gauge transformation $G_t$).

According to this theorem, for certain classes of functions, the dynamics will converge to two different clusters. These clusters are exactly those corresponding to the bipartite consensus condition in Lemma 1(5).

In the following subsections, we elaborate on the controllability of the dynamics (3.6)-(3.8) and show that a notion of symmetry about the input node, as well as structural balance, lead to uncontrollability. For each case we consider even (symmetric) and odd (anti-symmetric) functions $f$. From [24] we know if the underlying signed graph $G$ is structurally balanced, then there exists a gauge transformation $G_t$ that acts as a similarity transformation on the adjacency matrix of $G$ in that $G_t A_s(G) G_t = A$ where $A$ is the adjacency matrix of unsigned $G$. We will show that $G_t$ defines a useful coordinate transformation that allows an immediate application of the uncontrollability test derived by Aguilar and Gharesifard [33].

Before that, we need the following definition to extend the notion of a graph symmetry to signed graphs.
Definition 2. Let \( \varphi \) be a non-identity automorphism on graph \( G \). Suppose that this graph is structurally balanced, with gauge transformation \( G_t \) induced by the function \( g : \mathbb{R}^n \rightarrow \mathbb{R}^n \) defined by \( g(x)_i = \sigma_i x_i \). Then, we define signed automorphism operator as \( \varphi' = g \circ \varphi \circ g \). Moreover, assume that \( J \) is the matrix representation of the permutation operator \( \varphi \), in that \( J = D\varphi \). Then, the analogous matrix \( J' = G_t J G_t \) is the matrix representation of the signed permutation operator \( \varphi' \), in that \( J' = D(g \circ \varphi \circ g) \).

Definition 2 implies that unlike the unsigned case, the signed automorphism contains sign alterations of edge weights while permuting the nodes. Hence, if \( \varphi(x_i) = x_r \), then \( \varphi'(x_i) = \sigma_i \sigma_r x_r \). For example, consider the graph in Figure 3.2.

![Figure 3.2: Example of a signed automorphism](image)

The corresponding gauge transformation and automorphisms are defined as

\[
G_t = \begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad J = \begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

\[
J' = G_t J G_t = \begin{bmatrix}
0 & -1 & 0 \\
-1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

One can note that while the structures of \( J \) and \( J' \) are similar, the signs of the elements are different. This also implies

\[
\varphi'(x_1, x_2, x_3) = (-x_2, -x_1, x_3) \Rightarrow \begin{cases}
\varphi'(x_1) = \sigma_1 \sigma_2 x_2 \\
\varphi'(x_2) = \sigma_1 \sigma_2 x_1 \\
\varphi'(x_3) = \sigma_3 \sigma_3 x_3
\end{cases}
\]

Recall that we use \( \phi(i) \) as the action of the automorphism on the index of a node rather than the more obscure notation \( \varphi(v_i) \). For example, in Figure 3.2 we have that \( \phi(1) = 2 \) and \( \phi(2) = 1 \).

3.3.0.1 Absolute Nonlinear Flow

The original definition of this kind of flow allows the function \( f \) to vary across the nodes [16]. In our problem setup, however, all such functions are assumed to be equal, and thus the dynamics resemble linear consensus in
that we can write \( \dot{x} = -L\mathcal{G}_x f(x) \), where \( f(x) \) is the function \( f \) applied entry-wise to the vector \( x \).

In the following theorem, we will show that for absolute nonlinear flow with odd functions \( f \), structural balance directly generalizes the uncontrollability conditions in [33]. For the case of even functions, we need to impose additional topological structure on the edge weights of the underlying graph.

**Theorem 9.** Consider a structurally balanced graph \( G \) with gauge transformation \( G_t \) and absolute nonlinear flow dynamics (3.6). Further suppose \( G \) has a non-trivial signed automorphism \( \phi' \). Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be a smooth odd function (for example, odd \( f \in S_0 \) with \( f \) smooth). Then, for any vertex \( j \in \text{Fix}(\phi') \) chosen as the leader, the leader-follower network is not accessible from the origin in \( \mathbb{R}^n \). Moreover, the same results holds for smooth even functions \( f \) if \( \phi' \) preserves edge signs, in that \( \text{sgn}(a_{ij}) = \text{sgn}(a_{\phi(i)\phi(j)}) \).

**Proof.** Following Equation (3.3), let \( F \) denote the network flow and assume the dynamics in (3.6). We will first note that for smooth odd functions \( f \), the dynamics of the system can become unsigned by a convenient coordinate transformation. Let \( z = G_t x \), or \( z_i = \sigma_i x_i \). Then, following [25] we get the equivalent dynamics

\[
\dot{z}_i = -\sigma_i \sum_{j \in N_i} f(\sigma_i z_i) - \text{sgn}(a_{ij}) f(\sigma_j z_j)
\]

\[
= -\sigma_i \sum_{j \in N_i} \sigma_i f(z_i) - \text{sgn}(a_{ij}) \sigma_j f(z_j)
\]

\[
= -\sum_{j \in N_i} f(z_i) - f(z_j),
\]

which is defined as the unsigned absolute nonlinear flow. Then, from Theorems 6 and 7 we conclude the system is inaccessible.

Now, suppose \( f \) is an even function. From (3.6) we have

\[
F_i(\phi'(x)) = -\sum_{l \in N_r} f(\sigma_i \sigma_r x_r) - \text{sgn}(a_{ij}) f(\sigma_j \sigma_l x_l)
\]

\[
= -\sum_{l \in N_r} f(x_r) - \text{sgn}(a_{ij}) f(x_l),
\]

where \( r = \phi(i) \) and \( l = \phi(j) \) and the property \( f(\sigma_i \sigma_j x) = f(x) \) of even functions is used.

On the other hand,

\[
F_{\phi(i)}(x) = F_r(x) = -\sum_{l \in N_r} f(x_r) - \text{sgn}(a_{rl}) f(x_l).
\]

Hence, \( F \) is \( \phi' \)-invariant if \( \text{sgn}(a_{ij}) = \text{sgn}(a_{rl}) \). \( \square \)

The condition on even functions in Theorem 9 can be interpreted as the sign symmetry of the graph, in that if the link between two nodes contains a negative weight, it will remain negative even after the signed automorphism. For instance, sign symmetry does not hold in the graph of Figure 3.2 since
3.3.0.2 Relative Nonlinear Flow

The main result of this section shows that unlike the absolute nonlinear flow, structural balance on top of the signed variant of the conditions in Theorem 6 imply that the network flow is not accessible from the origin for both even and odd functions, without any additional constraints on edge weight permutations.

**Theorem 10.** Consider a structurally balanced graph $G$ with gauge transformation $G_t$ and relative nonlinear flow dynamics. Further suppose $G$ has a non-trivial automorphism $\varphi'$. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a smooth odd or even function (for example, odd $f \in S_0$ with $f$ smooth). Then, for any vertex $j \in \text{Fix}(\varphi')$ chosen as the leader, the leader-follower network is not accessible from the origin in $\mathbb{R}^{n-1}$.

**Proof.** Let the same notations as in proof of Theorem 9 hold. We will show that both cases of odd and even functions $f$ lead to $\varphi'$-invariance of the flow $F$ and therefore the inaccessibility from the origin.

Let $f$ be an odd function. Changing the coordinates by $z = G_t x$ yields

$$
\dot{z}_i = -\sigma_i \sum_{j \in N_i} f(\sigma_i z_i - \text{sgn}(a_{ij}) \sigma_j z_j) \\
= -\sigma_i \sum_{j \in N_i} f(\sigma_i (z_i - \sigma_i \text{sgn}(a_{ij}) \sigma_j z_j)) \\
= -\sum_{j \in N_i} f(z_i - z_j),
$$

which is the unsigned relative nonlinear flow. Hence, $F$ is $\varphi'$-invariant and inaccessibility from the origin follows from Theorems 6 and 7.

For even function $f$, from (3.7)

$$
F_i(\varphi'(x)) = -\sum_{l \in N_r} f(\sigma_l \sigma_r x_r - \sigma_l \sigma_r \text{sgn}(a_{ij}) x_l) \\
= -\sum_{l \in N_r} f(\sigma_l \sigma_r (x_r - \sigma_l \sigma_r \sigma_j \text{sgn}(a_{ij}) x_l)) \\
= -\sum_{l \in N_r} f(x_r - \sigma_r \sigma_l x_l) \\
= -\sum_{l \in N_r} f(x_r - \text{sgn}(a_{rl}) x_l) \\
= F_r(x),
$$

where we have used the fact that $\sigma_l \sigma_j \text{sgn}(a_{ij}) > 0$ hence $\sigma_l \sigma_j = \text{sgn}(a_{ij})$ for all $i$ and $j \in N_i$. \(\Box\)
3.3.0.3 Disagreement Nonlinear Flow

Applying the same assumptions for this type of flow, we will prove similar results without proof for the sake of completeness.

Theorem 11. Consider a structurally balanced graph $G$ with gauge transformation $G_t$ and disagreement nonlinear flow dynamics. Further suppose $G$ has a non-trivial automorphism $\phi'$. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a smooth odd or even function (for example, odd $f \in \mathcal{S}_0$ with smooth). Then, for any vertex $j \in \text{Fix}(\phi')$ chosen as the leader, the leader-follower network is not accessible from the origin in $\mathbb{R}^{n-1}$.

Proof. The proof is almost identical to that of Theorem 10 with the same change of coordinates for the odd functions, and similar sign analysis for the even case. 

Remark 5. The analysis of this section demonstrates that for all of the three nonlinear flows (3.6)-(3.8) structural balance in addition to $\phi'$-invariance leads to system uncontrollability for even and odd functions $f$. The only exception is when the absolute nonlinear flow $f$ is even. In this case, an edge-sign symmetry condition is also required.

3.4 An Aside on Structural Balance

As stated in the introduction, it is of interest to study the properties of large graphs by examining smaller subgraphs, and how the property composes as you build the graph up. For example, Chapman et al. has studied controllability under Cartesian products of graphs [45]. A comprehensive summary of graph products can be found in one of [46]–[48]; the latter if one has a strong grasp of the Russian language.

Suppose that $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$. Then the Cartesian product graph $G_1 \square G_2$ has vertex set $V_1 \times V_2$, and edge set $E_{1 \times 2}$ such that $\{(v_1, v_2), (v_1', v_2')\} \in E_{1 \times 2}$ if and only if either $\{v_1, v_1'\} \in E_1$ and $v_2 = v_2'$, or $\{v_2, v_2'\} \in E_2$ and $v_1 = v_1'$. The weight of this edge is given by

$$w((v_1, v_2), (v_1', v_2')) = \delta_{v_1, v_1'} w_{v_2, v_2'} + \delta_{v_2, v_2'} w_{v_1, v_1'}.$$

The Cartesian product of graphs is commutative (there is an isomorphism between $G_1 \square G_2$ and $G_2 \square G_1$), and and associative (there is an isomorphism between $(G_1 \square G_2) \square G_3$ and $G_1 \square (G_2 \square G_3)$). We denote the Cartesian product of $n$ graphs $G_1, \ldots, G_n$ as

$$\square_{i=1}^n G_i := G_1 \square \ldots \square G_n.$$

We show below how structural balance is preserved under the composition of Cartesian products.
Theorem 12. Let $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$ be signed graphs. Then, the graph product
\[
\mathcal{G}^{[n]} := \square_{i=1}^n \mathcal{G}_i
\]
is structurally balanced if and only if for each $1 \leq i \leq n$, $\mathcal{G}_i$ is structurally balanced.

Furthermore, suppose that the $\mathcal{G}_i$’s are structurally balanced signed graphs with pairs of bipartition vertex sets $(V_1^i, V_2^i), (V_1^2, V_2^2), \ldots, (V_1^n, V_2^n)$. Then, the bipartition of the graph product in Equation (3.9) is given by $(V_1^{[n]}, V_2^{[n]})$ defined in the following way.

Let $l_n := a_1a_2 \ldots a_n$ be a 2-ary sequence of length $n$ consisting of integers $a_i \in \{1, 2\}$. There are $2^n$ such sequences. We call $l_n$ odd if $\sum_{i=1}^n a_i$ is odd, and even otherwise. For each such sequence $l_n$, we identify a bijection with a sequence of sets $V_i^{a_i}$ as follows:
\[
a_1a_2 \ldots a_n \leftrightarrow V_1^{a_1}V_2^{a_2} \cdots V_n^{a_n}.
\]

Lastly, let $S^n_e$ and $S^n_o$ be defined as the sets of even and odd such sequences respectively:
\[
S^n_e = \{l_n \in \{1, 2\}^n | l_n = a_1 \ldots a_n, l_n \text{ even}\}
\]
\[
S^n_o = \{l_n \in \{1, 2\}^n | l_n = a_1 \ldots a_n, l_n \text{ odd}\}.
\]

Then, the bipartition $(V_1^{[n]}, V_2^{[n]})$ is given by taking unions over products of bipartition vertex sets corresponding to even and odd sequences $l_n$ as follows:
\[
V_1^{[n]} = \bigcup_{l_n \in S^n_e} \left[ \prod_{i=1}^n V_i^{a_i} \right] = \bigcup_{l_n \in S^n_e} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_n^{a_n}, \tag{3.11}
\]
\[
V_2^{[n]} = \bigcup_{l_n \in S^n_o} \left[ \prod_{i=1}^n V_i^{a_i} \right] = \bigcup_{l_n \in S^n_o} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_n^{a_n}. \tag{3.12}
\]

Proof. The first statement can be proven by induction. The base case is given by Theorem 5 in [29]. For the inductive step, suppose that the graph product
\[
\mathcal{G}^{[n]} := \square_{i=1}^n \mathcal{G}_i
\]
is structurally balanced. Then, by Theorem 5 in [29], we know that if $\mathcal{G}_{n+1}$ is structurally balanced, then $\mathcal{G}^{[n+1]} := \mathcal{G}^{[n]} \square \mathcal{G}_{n+1}$ is structurally balanced. But, by the associative property of the Cartesian product, we can see that
\[
\mathcal{G}^{[n]} \square \mathcal{G}_{n+1} = \square_{i=1}^{n+1} \mathcal{G}_i,
\]
and so the result follows.
We can also prove Equations (3.11) and (3.12) by induction, using a bijective combinatorial proof. Consider the base case for \( n = 3 \): suppose that \( \mathcal{G}_1, \mathcal{G}_2 \) and \( \mathcal{G}_3 \) have bipartitions \((V_1, V_2), (U_1, U_2)\) and \((T_1, T_2)\) respectively. Consider the graph product \( \mathcal{G}_1 \square \mathcal{G}_2 \square \mathcal{G}_3 \).

By the definition of the Cartesian product, we have that \( \mathcal{G}_1 \square \mathcal{G}_2 \) has bipartition

\[
V'_1 = (V_1 \times U_1) \cup (V_2 \times U_2)
\]

\[
V'_2 = (V_1 \times U_2) \cup (V_2 \times U_1),
\]

and so the graph product \( \mathcal{G}_1 \square \mathcal{G}_2 \square \mathcal{G}_3 = \mathcal{G}_1 \square \mathcal{G}_2 \square \mathcal{G}_3 \) has bipartition

\[
T'_1 = (V'_1 \times T_1) \cup (V'_2 \times T_2)
\]

\[
= (V_1 \times U_1 \times T_1) \cup (V_2 \times U_1 \times T_1)
\]

\[
\cup (V_2 \times U_2 \times T_2) \cup (V_1 \times U_2 \times T_2)
\]

\[
T'_2 = (V'_1 \times T_1) \cup (V'_2 \times T_1)
\]

\[
= (V_1 \times U_1 \times T_1) \cup (V_2 \times U_1 \times T_1)
\]

\[
\cup (V_1 \times U_2 \times T_1) \cup (V_2 \times U_2 \times T_1).
\]

There are \( 2^3 = 8 \) 2-ary sequence \( l_n \), and notice that the terms in Equation (3.13) correspond by the bijection (3.10) to the 4 odd sequences and the terms in Equation (3.14) correspond to the 4 even sequences. This concludes the base case.

For the induction step, suppose for each \( 1 \leq m \leq n \) that the (structurally balanced) graph product

\[
\mathcal{G}^{[m]} := \square_{i=1}^n \mathcal{G}_i
\]

has bipartition \((V_1^{[m]}, V_2^{[m]})\) corresponding to Equations (3.11) and (3.12). Let \( \mathcal{G}_{n+1} \) be structurally balanced with bipartition \((V_1^{[n+1]}, V_2^{[n+1]})\). Then, by the definition of the Cartesian product, the graph product \( \mathcal{G}^{[n+1]} := \mathcal{G}^{[n]} \square \mathcal{G}_{n+1} \) has bipartition

\[
V_1^{[n+1]} = (V_1^{[n]} \times V_1^{[n+1]}) \cup (V_2^{[n]} \times V_2^{[n+1]})
\]

\[
V_2^{[n+1]} = (V_1^{[n]} \times V_2^{[n+1]}) \cup (V_2^{[n]} \times V_1^{[n+1]}).
\]

Let’s examine the two terms appearing on the right-hand-side of Equation (3.15). First, we can see that

\[
V_1^{[n]} \times V_1^{[n+1]} = \bigcup_{l_a \in S_a^n} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_n^{a_n} \times V_1^{n+1}
\]

\[
= \bigcup_{l_a \in S_a^n} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_n^{a_n} \times V_1^{n+1}.
\]

Hence, every term in the union corresponds by the bijection (3.10) to a 2-ary sequence of length \( n + 1 \) given by \( a_1 a_2 \ldots a_n 1 \). Since \( \sum_{i=1}^n a_n \) is even,
$1 + \sum_{i=1}^{n} a_i$ is odd. The second term on the right-hand-side of Equation (3.15) is similarly given by

$$V_2^n \times V_2^{n+1} = \bigcup_{l_n \in S_n} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_n^{a_n} \times V_2^{n+1},$$

whose terms correspond by the bijection (3.10) to 2-ary sequences of length $n + 1$ given by $a_1 a_2 \ldots a_2$. Since $\sum_{i=1}^{n} a_n$ is odd, $2 + \sum_{i=1}^{n} a_i$ is odd. These two terms correspond to all the ways of getting odd, length-$(n + 1)$, 2-ary sequences: by taking length-$n$ sequences and concatenating the appropriate term in $\{1, 2\}$ to the end of them. It follows by the bijection (3.10) that

$$V_1^{[n+1]} = \bigcup_{l_n \in S_{n+1}} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_{n+1}^{a_{n+1}}.$$  

A similar calculation by considering the two terms in the right-hand-side of Equation (3.16) yields

$$V_2^{[n+1]} = \bigcup_{l_n \in S_{n+1}} V_1^{a_1} \times V_2^{a_2} \times \cdots \times V_{n+1}^{a_{n+1}},$$

and so the result follows by induction.

For the converse direction, suppose that $G_1 \square G_2$ is structurally balanced. In order to derive a contradiction, suppose (without loss of generality) that $G_1$ is not structurally balanced. Then, $G_1$ has a negative cycle. By the definition of the Cartesian product, $G_1$ appears as a subgraph in $G_1 \square G_2$, and so the negative cycle in $G_1$ appears in $G_1 \square G_2$, a contradiction. A similar contradiction can be achieved for an induction step, yielding the converse direction. 

□
KOOPMAN ANALYSIS ON SIGNED NETWORKS

4.1 KOOPMAN OPERATOR THEORY

In the early 1930s, Koopman showed that nonlinear dynamical systems associated with Hamiltonian flows could be analyzed with an infinite-dimensional linear operator on the Hilbert space of functions on the state space of that dynamical system. The resulting field, called Koopman analysis, has been at the heart of recent data-driven efforts to characterize complex systems and there is a considerable interest in obtaining finite-dimensional representations of the Koopman operator that offer high-fidelity approximation of nonlinear systems.

For an introductory example, we show one rigorous way how a finite-dimensional ordinary differential equation can be written as an infinite-dimensional partial differential equation [49]. Suppose we have the nonlinear system of ODEs

$$\frac{d}{dt} \phi(x, t) = R(\phi(x, t)), \quad \phi(x, 0) = x$$  \hspace{1cm} (4.1)

and we want to write it as

$$u_t = Lu, \quad u(x, 0) = g(x)$$

for some choice of $u, L, g$. First, we need a lemma.

**Lemma 12.**

$$R (\phi(x, t)) = D_x \phi(x, t) R(x)$$

where $D_x \phi$ is the Jacobian of $\phi$ given by

$$D_{x_j} \phi_i(x, t) = \frac{\partial \phi_i}{\partial x_j}.$$  

**Proof.** Let $F = R (\phi(x, t)) - D_x \phi(x, t) R(x)$. We want to shown that $F$ is identically zero. First, at $t = 0$ we have

$$F(x, 0) = R (\phi(x, 0)) - D_x \phi(x, 0) R(x)$$

$$= R(x) - IR(x) = 0,$$
since \( D_x \phi(x, 0) = D_x x = I \). Now, look at

\[
\frac{\partial F}{\partial t} = \frac{\partial R}{\partial t} - \frac{\partial}{\partial t} \left[ D_x \phi(x, t) R(x) \right]
\]

\[
= [D_x R] \left( \phi(x, t) \right) \frac{\partial \phi(x, t)}{\partial t} - \left( D_x \frac{\partial \phi}{\partial t} \right) R(x)
\]

The IVP defined by

\[
\frac{\partial F}{\partial t} = [D_x R] \left( \phi(x, t) \right) F, \quad F(x, 0) = 0
\]

has the unique solution (given \( R, \phi \) are smooth) of \( F(x, t) = 0 \). \( \square \)

Now we can prove the main result.

**Claim 1.** Let \( g \) be an arbitrary smooth function. Then, when \( \phi(x, t) \) solves the problem (4.1), we have that \( u(x, t) = g(\phi(x, t)) \) is the unique solution to

\[
u_t = Lu, \quad u(x, 0) = g(x)
\]

where \( L \) is the Liouville operator given by

\[
L = \sum_i R_i \frac{\partial}{\partial x_i}.
\]

**Proof.** Note that \( u(x, 0) = g(x) \) immediately from the relation \( u(x, t) = g(\phi(x, t)) \). Then,

\[
\frac{\partial u}{\partial t} = \sum_i \frac{\partial g(\phi(x, t))}{\partial x_i} \frac{\partial \phi_i(x, t)}{\partial t}
\]

\[
= \sum_i R_i (\phi(x, t)) \frac{\partial g(\phi(x, t))}{\partial x_i}.
\]
Recall by the lemma that \( R_i(\phi(x,t)) = D_x \phi(x,t) R_i(x) = \sum_j \frac{\partial \phi_j(x,t)}{\partial x_j} R_j(x) \), and so we have

\[
\frac{\partial u}{\partial t} = \sum_i \left( \sum_j \frac{\partial \phi_i(x,t)}{\partial x_j} R_j(x) \right) \frac{\partial g(\phi(x,t))}{\partial x_i} \\
= \sum_j R_j(x) \left( \sum_i \frac{\partial g(\phi(x,t))}{\partial x_i} \right) \frac{\partial \phi(x,t)}{\partial x_j} \\
= \sum_j R_j(x) \frac{\partial g(\phi(x,t))}{\partial x_j} \\
= Lu
\]

\[\square\]

In summary, given a system of ODEs

\[
\frac{d}{dt} \phi(x,t) = R(\phi(x,t)), \quad \phi(x,0) = x
\]

with solution \( \phi(x,t) \), we can write it as

\[
\frac{\partial u}{\partial t} = \sum_i R_i(x) \frac{\partial g(\phi(x,t))}{\partial x_i}, \quad u(x,0) = g(x)
\]

with solution \( u = g(\phi(x,t)) \).

We can relate this to a probabilistic interpretation in the context of Hamiltonian mechanics.

Recall that the Hamiltonian \( \mathcal{H} \) is roughly the total energy of the system. A system is called Hamiltonian if it obeys Hamilton’s equations of motion, given by

\[
\frac{dp}{dt} = -\frac{\partial \mathcal{H}}{\partial q}, \quad \frac{dq}{dt} = \frac{\partial \mathcal{H}}{\partial p}
\]

where \( q \) denotes the position vector and \( p \) denotes the momentum vector.

Suppose we have a Hamiltonian system with \( n \) degrees of freedom, namely that \( q = (q_1, \ldots, q_n) \) and \( p = (p_1, \ldots, p_n) \). The system exists in a 2\( n \)-dimensional phase space \( \Gamma = (q,p) \). Since we cannot computationally hope to keep track of 2\( n \) equations of motion, what we will do is assume that the initial data \( p(0), q(0) \) are drawn from some probability distribution \( W(t) \) and look at the collection, or ensemble, of trajectories that are initially distributed by \( W(t) \). To this end, we have a few definitions:

1. **Microstate**: A vector \([q(t), p(t)] \in \Gamma\); drawn from \( W(t) \)
2. **Macrostate**: \( W(t) \); describes the ensemble at time \( t \)
3. **Sample Space**: \( \Omega \); the set of all macrostates
4.1 Koopman Operator Theory

Now, we derive the equations of motion for the macrostate. Define $u = (\dot{q}_1, \ldots, \dot{p}_n)$. Then, $u$ is an incompressible vector field:

\[
\nabla u = \sum_{i=1}^{n} \frac{\partial}{\partial q_i} \left( \frac{dq_i}{dt} \right) + \sum_{i=1}^{n} \frac{\partial}{\partial p_i} \left( \frac{dp_i}{dt} \right)
\]

\[
= \sum_{i=1}^{n} \frac{\partial}{\partial q_i} \frac{\partial H}{\partial p_i} - \frac{\partial}{\partial p_i} \frac{\partial H}{\partial q_i}
\]

\[
= \sum_{i=1}^{n} \frac{\partial^2 H}{\partial q_i \partial p_i} - \frac{\partial^2 H}{\partial p_i \partial q_i} = 0.
\]

Next, consider a volume $V$ in the phase space $\Gamma$, with probability density $W$. Then, the number of microstates in $V$ at time $t$ is on average given by

\[
\int_V W \, dq \, dp = \int_V W \, dV
\]

where $dp = dp_1 \ldots dp_n$ and $dq = dq_1 \ldots dq_n$. Now, microstates do not spontaneously appear or disappear. Therefore, the change in the number of microstates only comes from the inflow and outflow at the boundary of $V$. Therefore, by Green’s Theorem, we get

\[
\frac{d}{dt} \int_V W \, dq \, dp = - \int_{\partial V} W (u \cdot n) \, ds = - \int_V \nabla (Wu) \, dV.
\]

If the density $W$ is smooth, then we get

\[
\int_V \left( \frac{\partial W}{\partial t} + \nabla (Wu) \right) \, dq \, dp = 0.
\]

This implies that

\[
\frac{\partial W}{\partial t} + \nabla (Wu) = 0
\]

\[
= \frac{\partial W}{\partial t} + W \cdot \nabla u + u \cdot \nabla W
\]

\[
= \frac{\partial W}{\partial t} + u \cdot \nabla W.
\]

Now, we can specify the Liouville operator

\[
L = - \sum_{i=1}^{n} \left( \frac{\partial H}{\partial p_i} \frac{\partial}{\partial q_i} - \frac{\partial H}{\partial q_i} \frac{\partial}{\partial p_i} \right)
\]

which will give

\[
\frac{\partial W}{\partial t} = LW.
\]

Based on the previous discussion on converting a system of nonlinear ODEs with nonlinear right-hand-side function $R$ into a linear system of PDEs, we
can see by the definition of the Liouville operator here that $R_i(x = (q, p)$ is given by

$$R_i(x) = \left[ \frac{dq_i}{dt}, \frac{dp_i}{dt} \right].$$

In summary, given a (nonlinear) system of ODEs

$$\frac{d}{dt} \phi(x, t) = R(\phi(x, t)), \quad \phi(x, 0) = x$$

with solution $\phi(x, t)$, we can write it as the (linear) system of PDEs

$$\frac{\partial u}{\partial t} = \sum_i R_i(x) \frac{\partial g(\phi(x, t))}{\partial x_i}, \quad u(x, 0) = g(x)$$

with solution $u = g(\phi(x, t))$. In more convenient notation, this is simply

$$\frac{\partial u}{\partial t} = Lu$$

where $L$ is the Liouville operator

$$\sum_i R_i(x) \frac{\partial}{\partial x_i}.$$ 

To get the equations of motion for a probability distribution (or macrostate), we can specify the Liouville operator

$$L = -\sum_{i=1}^n \left( \frac{\partial H}{\partial p_i} \frac{\partial}{\partial q_i} - \frac{\partial H}{\partial q_i} \frac{\partial}{\partial p_i} \right)$$

$$= \sum_{i=1}^n \left[ \frac{dq_i}{dt}, \frac{dp_i}{dt} \right] \left[ \frac{\partial}{\partial q_i}, \frac{\partial}{\partial p_i} \right].$$

In other words, the function $R_i(x)$ is simply the Hamiltonian equations of motion

$$R_i(x) = \left[ \frac{dq_i}{dt}, \frac{dp_i}{dt} \right]$$

and the partial derivative with respect to $x$ is over the phase space variables $x = (q, p)$:

$$\frac{\partial}{\partial x} = \left[ \frac{\partial}{\partial q_i}, \frac{\partial}{\partial p_i} \right].$$
This will give the evolution of the probability density generating $x(t) = (q(t), p(t))$

$$\frac{\partial W}{\partial t} = LW.$$

In a manner analogous to the above discussion, the Koopman operator $\mathcal{K}$ ‘lifts’ a nonlinear ODE into a linear, infinite-dimensional PDE on the space of observables.

The Koopman operator is a powerful tool for the analysis and decomposition of nonlinear dynamical systems. This makes more sense if we assume that we have a linear approximation of a nonlinear dynamical system. Then, it becomes possible to not only investigate the behavior of the dynamical system as a linear system (where we know all the prominent definitions of controllability, stability, LQR, etc.), but also recover the state from the Koopman features in some cases.

Consider the following discrete-time nonlinear dynamical system

$$x_{k+1} = F(x_k),$$

where $x_k$ is the state of the system at timestep $k$, and $F : \mathcal{M} \rightarrow \mathcal{M}$ is a nonlinear dynamical map propagating the state. Then, the Koopman operator $\mathcal{K}$, is defined on functions of states $\psi : \mathcal{M} \rightarrow \mathbb{R}$ as follows:

$$\mathcal{K} \psi = \psi \circ F \quad \Rightarrow \quad \mathcal{K} \psi(x_k) = \psi(F(x_k)) = \psi(x_{k+1})$$

$$\Rightarrow \quad \psi(x_{k+1}) = \mathcal{K} \psi(x_k). \quad (4.2)$$

We call the function $\psi$ an observable, and the Koopman operator governs its evolution in time. Note that the action of the Koopman operator on $\psi$ given in Equation (4.2) is linear.

Although the action of $\mathcal{K}$ is linear, the space of functions $\psi$ is infinite-dimensional in general. Therefore, although one can envision some basis of functions describing all possible $\psi$ on the manifold of the dynamical system, one cannot in general write, say, a finite-dimensional matrix representation of $\mathcal{K}$. Moreover, the dynamical framework adds some additional complexity in terms of decomposing the Koopman operator into algebraic quantities. By this, we mean that not only are there notions of “eigenvectors” of this operator, there are also eigenfunctions.

We call the function $\varphi(x)$ an eigenfunction of $\mathcal{K}$ corresponding to the eigenvalue $\mu$ if $\mathcal{K} \varphi(x) = e^{\mu t} \varphi(x)$. For an observable function $\psi$ in the span of
Koopman eigenfunctions, one can write \( \psi(x) = \sum_{k=1}^{\infty} \psi_k \phi_k(x) \), where the \( \psi_k \)'s are the Koopman modes. For the case of full-state observable \( \psi(x) = x \) the states can be reconstructed

\[
x(t) = g(x(t)) = \sum_{k=1}^{\infty} e^{\mu_k t} \varphi_k(x_0) v_k,
\]

where we refer to \( \{\mu_k, \varphi_k, v_k\} \) as the Koopman triple.

As stated before, the Koopman operator is linear, but can be infinite-dimensional. To make this operator computationally usable, a numerical approximation of the Koopman operator can be obtained by EDMD, resulting in a finite-dimensional approximation \( \mathbf{K} \).

We describe the algorithm to find the Koopman finite-dimensional approximation \( \mathbf{K} \). First, the required quantities to input into the algorithm are

- Pairs of snapshots gathered from data in the form of
  \[
  X = [x_1 x_2 \ldots x_M], \quad Y = [y_1 y_2 \ldots y_M]
  \]
  where \( y_i = F(x_i) \) and \( M \) is the number of the measurements taken from the system.

- A clever choice of observables in the form of
  \[
  \Psi(x) = [\psi_1(x) \psi_2(x) \ldots \psi_{N_K}(x)]
  \]
  where \( N_K \) is a number approximately denoting the 'order' of the approximation. For example, one can take polynomials up to order \( N_K \).

- An appropriate weight matrix for constructing the modes later in the algorithm
  \[
  B = [b_1 b_2 \ldots b_{N_K}]
  \]

Then, we can obtain the finite-dimensional approximation as follows

\[
\mathbf{K} = \mathbf{G}^\top \mathbf{A}
\]

where \( \mathbf{A} \) and \( \mathbf{G} \) are computed as

\[
\mathbf{G} = \frac{1}{M} \sum_{m=1}^{M} \Psi(x_m)^\top \Psi(x_m)
\]

\[
\mathbf{A} = \frac{1}{M} \sum_{m=1}^{M} \Psi(x_m)^\top \Psi(y_m).
\]

Then, the approximate Koopman modes \( \{v_i\}_{i=1}^{N_K} \) are given by

\[
v_i = (w_i^\top B)^T,
\]

where \( w_i \) is the \( i \)th left eigenvector of \( \mathbf{K} \).
4.2 Identification of Bipartition in Structurally Balanced Graphs with the Koopman Operator and EDMD

In this section, we extend the data-driven approach by Pan et al. in using data-driven methods to identify the bipartite consensus for some of the non-linear network flows considered in the preceding section. Our main result asserts that the Koopman mode corresponding to the zero eigenvalue of the Koopman operator contains the sign structure indicating the bipartite consensus.

**Theorem 13.** Consider either the dynamics (3.6) with $f \in S$ or the dynamics (3.7) with $f \in S_0$. Recall that the full-state observable can be written in terms of the Koopman triple as

$$ x(t) = \sum_{j=1}^{\infty} e^{\lambda_j t} \varphi_j(x_0)v_j. \quad (4.3) $$

If the underlying graph is structurally balanced, then the following hold:

1. $\lambda_i \leq 0$ with a unique zero eigenvalue.

2. Let $\lambda_1 = 0$. Then, the sign structure of the corresponding Koopman mode $v_1$ corresponds to the bipartition of nodes denoted in Lemma 1(5).

**Proof.** By Theorem 8, if $G$ is structurally balanced, then we have that

$$ \lim_{t \to \infty} x(t) = \frac{1}{n} \left( 1^T G_t x_0 \right) G_t 1. $$

Hence, $\lambda_i \leq 0$ since otherwise the RHS of Equation (4.3) does not converge.

The sign structure of the vector $G_t 1$ corresponds to bipartition of nodes denoted in Lemma 1(5). Since $\lambda_1 = 0$ is unique, by setting $\alpha = (1/n) 1^T G_t x_0$ we have that

$$ \lim_{t \to \infty} x(t) = \lim_{t \to \infty} \sum_{j=1}^{\infty} e^{\lambda_j t} \varphi_j(x_0)v_j = \varphi_1(x_0)v_1 = \alpha G_t 1. $$

Since both $\alpha$ and $\varphi_1(x_0)$ are constants, we can see that $v_1 \propto G_t 1$, and the result follows. □

In §5.5, we show an example where we use EDMD to approximate the first mode of the Koopman operator to obtain the sign structure corresponding to the bipartite consensus.
In this section, we show some examples that highlight the necessity of combining $\varphi$-invariance, structural balance and leader-node symmetry for uncontrollability of signed networks. To see examples that show the necessity of combining $\varphi$-invariance and leader-node symmetry for unsigned graphs, we refer the reader to [33]. In particular, we show the difference between consensus under signed and unsigned consensus, for linear and nonlinear consensus. We then show an example of using EDMD to obtain the bipartite consensus structure of a nonlinear flow on a structurally balanced graph.

5.1 Bipartite Consensus

Signed Laplacians considered in this section exhibit clustering when structurally balanced. Consider the signed network in Figure 5.3a, and its unsigned counterpart. Note that the signed network is structurally balanced, since the product of the edge weights over all three cycles is positive. The consensus dynamics (with no control input) beginning at $x_0 = [1, 2, 3, 4]^T$ for these two networks are shown in Figure 5.1. As one can see, the signed consensus converges to two clusters rather than a single equilibrium point.

Figure 5.1: Solid lines: unsigned consensus. Dashed lines: signed consensus on structurally balanced graph
5.2 MIMO PERTURBED LAPLACIAN

In this example, we consider the MIMO case with two input signals injected onto nodes 4 and 5. The partition is equitable and \( \pi = \{ C_1, C_2, C_3, C_4 \} \) and \( \pi_f = \{ C_1, C_2 \} \). Moreover

\[
A = \begin{bmatrix}
0 & -1 & -1 & 0 & -1 \\
-1 & 0 & 1 & 1 & 0 \\
-1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}, \quad P' = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
A(G/\pi) = \begin{bmatrix}
0 & 2 & 0 & 1 \\
1 & 1 & 1 & 0 \\
0 & 2 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{bmatrix}
\]

The system in figure 5.2a is structurally balanced and \( \pi_f \) contains the non-trivial cell \( C_2 \) in \( \pi \) and thus the system is uncontrollable. However, changing one negative sign to positive as depicted in figure 5.2 makes the system structurally unbalanced and the system becomes controllable.

Figure 5.2: (a) Uncontrollable and structurally balanced graph with \( \pi_f \) containing non-trivial cells in \( \pi \). (b) The same system with one change in the signs. No structural balance and controllable.

5.3 INFLUENCED CONSENSUS

Previously, we discussed that symmetry alone is not sufficient for uncontrollability. In this section we show an example of a symmetric signed network that is controllable.

Consider the influenced consensus networks in Figure 5.3. As one can see, the network in Figure 5.3a is structurally balanced, but the one in Figure 5.3b is not. The Laplacians for these two networks are, respectively,
\[
L_1 = \begin{bmatrix}
2 & -1 & 0 & 1 \\
-1 & 3 & 1 & 1 \\
0 & 1 & 2 & 1 \\
1 & 1 & 1 & 3
\end{bmatrix},
L_2 = \begin{bmatrix}
2 & -1 & 0 & 1 \\
-1 & 3 & -1 & 1 \\
0 & -1 & 2 & 1 \\
1 & 1 & 1 & 3
\end{bmatrix}.
\]

The controllability matrices for these two networks are, respectively:

\[
C_1 = \begin{bmatrix}
0 & 1 & 4 & 16 \\
0 & 1 & 4 & 16 \\
0 & 1 & 4 & 16 \\
1 & 3 & 12 & 48
\end{bmatrix},
C_2 = \begin{bmatrix}
0 & 1 & 4 & 14 \\
0 & 1 & 6 & 32 \\
0 & 1 & 6 & 30 \\
1 & 3 & 12 & 52
\end{bmatrix}.
\]

As one can see, the network in Figure 5.3a is uncontrollable since \(C_1\) is rank-deficient, but the network in Figure 5.3b is controllable, and hence one can conclude that unsigned symmetry is not sufficient for uncontrollability.

5.4 **Unsigned Symmetry is Not Sufficient for Uncontrollability**

Here we show an example of a network flow on a signed graph which has a leader-node symmetry. We show that in one case, the graph is structurally balanced, and the induced flow is hence uncontrollable. By altering the sign on a single edge, structural balance is lost and the resulting network flow is controllable.

Consider the structurally balanced graph in Figure 5.3a, and the structurally unbalanced graph in Figure 5.3b. These have respective dynamics

\[
\dot{x} = -L_{\mathcal{G}1}x - 1u, \quad \dot{x} = -L_{\mathcal{G}2}x - 1u,
\]

with Laplacians

\[
L_1 = \begin{bmatrix}
2 & -1 & 0 \\
-1 & 3 & 1 \\
0 & 1 & 2
\end{bmatrix},
L_2 = \begin{bmatrix}
2 & -1 & 0 \\
-1 & 3 & -1 \\
0 & -1 & 2
\end{bmatrix}.
\]
The controllability matrices of these dynamics are respectively,
\[
C_1 = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix},
C_2 = \begin{bmatrix}
1 & 1 & -1 \\
1 & 3 & 11 \\
1 & 3 & 9
\end{bmatrix}.
\]

5.5 Identification of Bipartite Structure

In this subsection, we consider a numerical method to identify the bipartite consensus structure of nonlinear dynamics on a structurally balanced graph. We do this by exploiting Theorem 13, and numerically approximating the Koopman mode corresponding to the zero eigenvalue. The goal is to take data from an unknown dynamical system and numerically identify the two groups of agents between which only exist antagonistic interactions, i.e. the bipartition of nodes in Lemma 1(5).

Consider the dynamics (3.7) with \( f(\cdot) = \sin(\cdot) \). The underlying structurally balanced graph is pictured in Figure 5.4.

![Figure 5.4: Underlying structurally balanced signed graph for EDMD example. Dashed edges indicate negative edges.](image)

We use EDMD to approximate the first Koopman mode corresponding to the zero eigenvalue. We refer the reader to [43] for a detailed explanation of the EDMD algorithm; a summary of the relevant computations is shown in Algorithm 1.

**Algorithm 1: Extended Dynamic Mode Decomposition**

**Input:** Data \( \{[X_i, Y_i]\}_{i=1}^{n-1} = \{[X_i, X_{i+1}]\}_{i=1}^{n-1} \)

Dictionary \( \Psi_i = \psi_i(x_1, \ldots, x_n), 1 \leq i \leq N_K \)

Matrix \( B \in \{0, 1\}^{n \times N_k} \) s.t. \( \Psi_j = x_i \leftrightarrow B_{ij} = 1 \)

**Result:** Koopman Modes \( \{v_i\} \)

**Compute**

\( G = 1/(n-1) \sum_{m=1}^{n-1} \Psi(X_m)^*\Psi(X_m) \)

\( A = 1/(n-1) \sum_{m=1}^{n-1} \Psi(X_m)^*\Psi(Y_m) \)

\( K = G^+A \)

**Decompose** \( K \) **Into**

**Eigenvalues:** \( \mu_i \)

**Right Eigenvectors:** \( \xi_i \)

**Left Eigenvectors:** \( w_i \)

**Compute**

**Koopman Modes:** \( v_i = (w_i^* B)^T, 1 \leq i \leq N_K \)
In particular, we use a dictionary of functions of the form $\psi_k = \prod_{i=1}^6 H(x_i, j_i)$ where $H(x_i, j_i)$ is the $j_i$th order Hermite polynomial of $x_i$. We use all such possible $\psi_k$ for Hermite polynomials up to order 2. Simple combinatorics indicates that there are 729 such functions $\psi_k$ for a 6-node graph; hence $N_k = 729$ in Algorithm 1.

The dynamics are shown in Figure 5.5 for an initial condition of $x_0 = (-1.73, -0.38, -0.21, 0.56, -0.65, -0.32)$. The EDMD procedure in Algorithm 1 was applied to three sets of data with a time-step of 0.1, as depicted in Figure 5.5: the full data (from $0 \leq t \leq 10$), the data in the red shaded region ($0 \leq t \leq 3$) and the data in the blue shaded region $2 \leq t \leq 5$). The computed Koopman modes corresponding to $\lambda_1 \approx 1$ for these regions respectively are

$$
\tilde{v}_1 = \begin{bmatrix} 0.018 \\ 0.026 \\ 0.016 \\ -0.020 \\ -0.020 \\ -0.011 \end{bmatrix}, \quad \tilde{v}_2^1 = \begin{bmatrix} 0.019 \\ 0.021 \\ 0.015 \\ -0.018 \\ -0.020 \\ -0.007 \end{bmatrix}, \quad \tilde{v}_3^1 = \begin{bmatrix} 0.016 \\ 0.023 \\ 0.015 \\ -0.018 \\ -0.016 \\ -0.012 \end{bmatrix},
$$

which all contain sign structure corresponding to the bipartition depicted in Figure 5.4. Despite not using all available data, the EDMD procedure was able to extract the bipartition well before the dynamics converged (after which the bipartite structure is obvious from the data). This begs the question how early can we detect the bipartite structure of the underlying dynamics? We will address this question in future works.
6.1 Motivation for Distributed State Estimation

In this section, we lay out the mathematical notation used for the remainder of the chapter, and summarize the essential background on Kalman filtering.

Consider the noisy discrete-time controlled linear system

\[
\begin{align*}
x_{k+1} &= Ax_k + Bu_k + \gamma_k w_k \\
\tilde{y}_k &= C_k x_k + v_k,
\end{align*}
\]

where \( x_k \in \mathbb{R}^n \) denotes the state vector, \( u_k \in \mathbb{R}^m \) denotes the control vector, \( \tilde{y}_k \in \mathbb{R}^p \) is the output vector or measurement, and \( w_k, v_k \) are Gaussian white noise vectors of appropriate dimensions with covariances \( Q_k \) and \( R_k \) respectively. The subscript \( k \) refers to the timestep. The random vectors \( w_k \) and \( v_k \) represent system disturbances and sensor noise respectively, and the matrix \( \gamma_k \) describes how the disturbance propagates into the system. Since most matrix quantities will have a subscript \( k \) denoting the timestep, we will refer to the \( ij \)th entry of a matrix \( A_k \) with bracketed superscripts: \( A_k^{(ij)} \). Similarly, the \( i \)th entry of a vector \( a_k \) is denoted \( a_k^{(i)} \), the \( i \)th column of a matrix \( A_k \) is denoted \( A_k^{(i)} \) and the \( i \)th row is denoted \( A_k^{(i)} \).

The purpose of Kalman filtering is to provide an accurate estimate \( \hat{x}_k \) of the state \( x_k \) using the measurement \( \tilde{y}_k \) and the known information about the system and noise. The standard discrete-time linear Kalman filter yields the estimate \( \hat{x}_{k+1} \) and is given by

\[
\begin{align*}
\hat{x}_{k+1} &= A (\hat{x}_k + K_k [\tilde{y}_k - C \hat{x}_k]) \\
K_k &= P_k C^T \left[ CP_k C^T + R_k \right]^{-1} \\
P_{k+1} &= A (I - K_k C) P_k A^T + \gamma_k Q_k \gamma_k^T,
\end{align*}
\]

where \( K_k \) is the Kalman gain and \( P_k \) is the error covariance matrix. The problem for high-dimensional systems is the inverse when computing \( K_k \). In this paper, we assume that the sensor noise is uncorrelated, and so \( R_k \) is a diagonal positive-definite matrix. This leaves the term \( CP_k C^T \); if this term is diagonal, then the matrix inverse becomes a series of \( n \) scalar divisions. We propose to replace \( P_k \) with a diagonal estimate \( \hat{P}_k \), and one can also assume that \( C \) is a matrix that measures individual states of the system without redundancy. Therefore, a system with \( 1 \leq p \leq n \) outputs will have a measurement matrix of the form \( C = [\text{diag}(C^{(ii)})]_p 0_{n-p} \). This assumption is common for networked systems where one measures the state of individual nodes of the network, see for example [18]. In the next few sections, we will formalize these assertions in the context of our algorithm.
6.2 General Kalman Filtering

In this section, we extend the gradient descent estimate for the error covariance proposed by [37]. In §6.2.1, we discuss the linear Kalman filter, and then extend the error covariance estimate to an accelerated gradient descent in §6.2.2. The accelerated gradient descent benefits from a clever adaptive learning rate, which is discussed in §6.2.3. The final algorithm combining all these methods is summarized in Algorithm 2.

6.2.1 Gradient Descent for the Error Covariance Update: Linear Case

Gradient descent is an iterative algorithm that seeks to find the local minimum of a function $f$ by stepping in the direction of the largest gradient:

$$
\beta_{k+1} = \beta_k - \mu \nabla f(\beta_k),
$$

where $\mu$ is the learning rate. For the Kalman filter, we seek to find a diagonal matrix estimate $\hat{P}_k$ of $P_k$. We do this by assuming $\hat{P}_k$ is of the form $\hat{P}_k = \text{diag}(\hat{P}_k^{(ii)}) = \text{diag}(e^{\hat{\beta}_k})$, where $\hat{\beta}_k$ is a parameter undergoing gradient descent attempting to minimize the norm of the error $\delta_k = \tilde{y}_k - C\hat{x}_k$:

$$
\hat{\beta}_{k+1}^{(i)} = \hat{\beta}_k^{(i)} - \frac{1}{2} \mu \frac{\partial (\delta_k^T \delta_k)}{\partial \hat{\beta}_k^{(i)}}.
$$

Computing the gradient of $\delta_k^T \delta_k$ as outlined in Appendix 6.4 yields the following gradient descent equations for $\hat{P}$:

$$
\begin{align*}
\hat{\beta}_{k+1}^{(i)} &= \hat{\beta}_k^{(i)} + \mu \delta_k^T C^{(i)} h_k^{(i)} \\
h_{k+1}^{(i)} &= h_k^{(i)} \left(1 - k_k^{(i)T} C^{(i)}\right)^+ + \left(k_k^{(i)} - k_k^{(i)T} C^{(i)}\right) h_k^{(i)} \delta_k \\
\hat{P}_k^+ &= \text{diag} \left( \exp \left( \hat{\beta}_{k+1}^{(i)} \right) \right)
\end{align*}
$$

(6.1)

The form of $\hat{P}^+$ guarantees that it remains positive-definite, which is required to preserve convergence of the Kalman filter.

6.2.2 Nesterov-Accelerated Methods

Nesterov acceleration is a method used to increase the convergence rate of gradient descent [(50)]. Although Nesterov-accelerated gradient descent converges in fewer timesteps, it does so by sacrificing monotonicity: the gradient descent trajectory will tend to oscillate as it converges towards the estimate.
In order to implement Nesterov acceleration, one must see that the quantity $h_k^{(i)}$ is a function of $K_{k-1}$, which in turn is a function of $\beta_k^{(i)}$:

\[
\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \beta_k^{(i)} \right) = \beta_k^{(i)} + \mu \delta_k^T C[i] h_k^{(i)} \left( K_{k-1} \left( \beta_k^{(i)} \right) \right).
\]

Hence, we can get the form of the Nesterov-accelerated gradient descent by introducing the quantity $\alpha_k^{(i)}$:

\[
\beta_{k+1}^{(i)} = \alpha_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \alpha_k^{(i)} \right) = \alpha_k^{(i)} + \mu \delta_k^T C[i] h_k^{(i)} \left( K_{k-1} \left( \alpha_k^{(i)} \right) \right)
\]

where $\alpha_k^{(i)}$ is given by

\[
\alpha_k^{(i)} = \beta_k^{(i)} + \frac{k - 1}{k + 2} \left( \beta_k^{(i)} - \beta_{k-1}^{(i)} \right).
\]

Therefore, the Nesterov-accelerated estimate $\hat{P}$ replaces the covariance update with the following equations:

\[
\hat{P} = \text{diag} \left( e^{\beta_{k+1}^{(i)}} \right), \quad k_k^{(i)} = \exp \left( \alpha_k^{(i)} \right) D_k^{-1} C[i]
\]

\[
\beta_{k+1}^{(i)} = \alpha_k^{(i)} + \mu \delta_k^T C[i] h_k^{(i)} \left( K_{k-1} \left( \alpha_k^{(i)} \right) \right), \quad (6.2)
\]

where $h_k$ is as in Equation (6.1), but using this new expression for $k_k^{(i)}$ in Equation (6.2).

### 6.2.3 Adaptive Learning Rate Methods

Using a constant learning rate $\mu$ can lead to suboptimal performance, and therefore it is prudent to adapt $\mu$ on-the-fly as the gradient descent is run. A well-known method for adapting the learning rate is given by Barzilai and Borwein [51]. Suppose we have the gradient descent with learning rate $\mu$:

\[
\beta_{k+1}^{(i)} = \beta_k^{(i)} - \frac{1}{2} \mu \nabla f \left( \beta_k^{(i)} \right).
\]

Let $\Delta \beta = \beta_k - \beta_{k+1}$ and let $\Delta g(\beta) = \nabla f \left( \beta_{k+1} \right) - \nabla f \left( \beta_k \right)$. Then, the Barzilai and Borwein method selects the learning rate parameter using a secant line approximation:

\[
\frac{1}{2} \mu_k = \arg \min_{\lambda} \| \Delta \beta - \lambda \Delta g(\beta) \|.
\]
For accelerated gradient descent, the updated learning rate $\mu$ is given by

$$
\mu_k = 2 \frac{\Delta g(\beta)^T \Delta \beta}{\Delta g(\beta)^T \Delta g(\beta)}, \Delta \beta = \beta_k - \alpha_{k-1}.
$$

Combining the accelerated gradient descent estimate of the error covariance with the adaptive learning rate $\mu_k$ yields the final algorithm as outlined in Algorithm 2.

**Algorithm 2:** Discrete Kalman Filter with Accelerated Gradient Descent on $P$ and Adaptive Learning Rate $\mu$

**Initialize**

$\hat{x}(t_1) = \hat{x}_0$

$P_0 = \mathbb{E}\{\hat{x}(t_1)\hat{x}(t_1)^T\}$

**for** $k = 1$ **to** $k = t_f$ **do**

1. **Gain**
   
   $K_k = \hat{P}_k^{-1}C^T(\hat{P}_k^{-1}C^T + R_k)^{-1}$

2. **Accelerated Gradient Descent and Adaptive $\mu_k$**
   
   $\Delta \beta = \beta_k - \alpha_{k-1}$
   
   $\Delta g(\beta) = \nabla f(\beta_{k+1}) - \nabla f(\beta_k)$
   
   $D_k = R_k + C\hat{P}_kC^T$
   
   $\mu_k = \frac{2(\Delta g(\beta)^T \Delta \beta)}{\Delta g(\beta)^T \Delta g(\beta)}$

3. **$\alpha_k(i)$**
   
   $\alpha_k(i)^{(i)} = \beta_k(i) + \frac{k + 1}{k + 2}(\beta_k(i) - \beta_k(i-1))$

4. **$\kappa_k(i)$**
   
   $\kappa_k(i) = \exp\left(\alpha_k(i)^{(i)}\right)$

5. **$\eta_k(i)$**
   
   $\eta_k(i)^{(i)} = \eta_k(i)^{(i)}(1 - \kappa_k(i)^{(i)}C(i)] + (\kappa_k(i)^{(i)} - \kappa_k(i)^{(i)}C(i)]^T \Delta_k$

6. **Update**
   
   $\hat{x}_k = \hat{x}_k + K_k[\hat{y}_k - C\hat{x}_k]$  

7. **Propagate**
   
   $\hat{x}_{k+1} = A_k \hat{x}_k + B_k u_k$

8. **$P_{k+1}$**
   
   $\hat{P}_{k+1} = \text{diag}\left(\exp\left(\beta_k(i)^{(i)}\right)\right)$

**end**

6.2.4 Filter Properties

In the standard Kalman filter, $P_k$ is computed from an algebraic Riccati equation. In order for this equation to have a positive-definite solution, a requirement for stable error dynamics, one must assume that $(A^T, C^T)$ is stabilizable and $(A, \sqrt{Q})$ is detectable (see [52]). Therefore, since our estimate of $P$ is positive definite by construction and not computed by an algebraic Riccati equation, we do not need to have these assumptions. The next theorem shows that all one needs for stability of the error dynamics is for the system matrix $A$ to be stable.
Theorem 14. Suppose $A$ is an asymptotically stable matrix, and so the eigenvalues of $A$ satisfy $|\lambda(A)| \leq 1$. Then, the error dynamics

$$\tilde{x}_k = x_k - \hat{x}_k$$

are stable, where $\hat{x}_k$ is the estimate computed according to Algorithm 1.

Proof. We can write the error dynamics as

$$\tilde{x}_{k+1} = x_{k+1} - \hat{x}_{k+1} = (A - LC)\tilde{x}_k + (w_k - L\nu_k),$$

where $L$ is given by

$$L = A\hat{P}C^T(C\hat{P}^T + R)^{-1}.$$

Hence, it suffices to show that $(A - LC)$ has eigenvalues in the unit disc. We can write $(A - LC)$ as

$$A - LC = A - A\hat{P}C^T(C\hat{P}^T + R)^{-1}C = A(I - N),$$

where $N = \hat{P}C^T(C\hat{P}^T + R)^{-1}C$. By the form of $C$ and $\hat{P}$, if the system has $p$ outputs, then $N$ is a diagonal matrix with the form

$$N^{(ii)} = \begin{cases} 
\frac{\epsilon_k^{(ii)}(C^{(ii)})^2}{\epsilon_k^{(ii)}(C^{(ii)})^2 + R^{(ii)}} & \text{if } 1 \leq i \leq p \\
0 & \text{if } p < i \leq n.
\end{cases}$$

It is clear that $0 \leq N^{(ii)} < 1$ since $R$ is a positive definite diagonal matrix. Therefore, $0 < (I - N)^{(ii)} \leq 1$, and so it follows that

$$|\lambda(L)| = |\lambda(A(I - N))| \\
\leq |\lambda_{\max}(A)||\lambda_{\max}(I - N)| \\
\leq |\lambda_{\max}(A)|,$$

and so by the asymptotic stability of $A$, the asymptotic stability of $(A - LC)$ follows. □

6.2.5 Example

Here, we show a simple numerical implementation of Algorithm 1 to see how the algorithm performs compared to the standard Kalman filter for different strengths of noise. Consider the linear system with matrices

$$A = \begin{bmatrix} 1.001 & 0.011 \\ -0.0301 & 0.98 \end{bmatrix}, \quad B = \begin{bmatrix} 5E - 5 \\ 1E - 2 \end{bmatrix}, \quad C = I$$

propagating with process noise of covariance $Q = 0.001$ and measurement noise of covariance $R = \alpha I$ for various values of $\alpha$. The first and second plots of Figure 6.1 show the error and first state of the system for $R = 0.115I$ for
Figure 6.1: Numerical test of Algorithm 1. Top: Error from true state from data, accelerated filter and Kalman filter. Middle: $y_1 = x_1$ for data, accelerated filter and Kalman filter. Bottom: Average steady-state error.

the raw measurement data, and the filtered data using the accelerated gradient descent and the standard Kalman filter. The third plot shows the average steady state error computed on the time interval $t \in [6, 10]$ for various values of $\alpha$ indicating the covariance of the measurement error. Figure 6.1 shows that the filter in Algorithm 1 achieves similar performance as the standard Kalman filter.

6.3 Distributed Kalman Filtering

[39] describes several methods of Kalman filtering using a network of sensors, with each $n$th sensor using a unique sensor model $\hat{y}_n, k = C_n, kx_k + v_n, k$, to arrive at a global measurement of a system. The context of their system model is on a graph $\mathcal{G}$, which is an abstract representation of the connection structure of sensors, which are represented by nodes. An edge in the graph denotes a communication link between sensors by which they can exchange information about their current state estimate. One such information exchange algorithm is called consensus, where each node in the graph changes its estimated value by continuously averaging with the estimates of its neighbours:

$$\dot{x}_i = \sum_{j \in N_i} (\dot{x}_j - \dot{x}_i).$$

One particular algorithm in [39] utilizes a Kalman-consensus filter in which every node of the network computes an estimate of the state of the system and then performs consensus with its neighbours on this estimate. This al-
lows sensors that only see a portion of the system to collect a global state estimate over time.

For high-dimensional systems, implementing this filter is problematic again because each sensor has to invert a large matrix at each timestep in order to compute the Kalman filter gain. In Algorithm 2, we summarize the Kalman-consensus filter, where every node performs an accelerated gradient descent with adaptive learning rate, and shares its estimate with its neighbours. In the following section, we provide an example of such a high-dimensional system.

**Algorithm 3:** Discrete Distributed Kalman-Consensus Filter $\mu$

*Initialize*

\[
\hat{x}(t_1) = \hat{x}_0 \\
P_0 = E\{\hat{x}(t_1)\hat{x}(t_1)^T\} \\
\epsilon = t_2 - t_1
\]

*for* $k = 1$ to $k = t_f$ *do*

*for each node* $j \in V$ *do*

**Aggregate Data**

\[
S_j = |N_i|^{-1} \sum_{l \in N_j} C_i \\
y_j = |N_i|^{-1} \sum_{l \in N_j} C_i^T \tilde{y}_i
\]

**Accelerated Gradient Descent and Adaptive $\mu_k$**

\[
\Delta \beta = \beta_k - \alpha_{k-1} \\
\Delta g(\beta) = \nabla f (\beta_{k+1}) - \nabla f (\beta_k) \\
D_k = R_k + C_kC^T \\
\mu_k = 2(\Delta g(\beta)^T \Delta \beta) / (\Delta g(\beta)^T \Delta g(\beta)) \\
\alpha_k = \beta_k + k + 1 \left( \beta_k - \beta_{k-1} \right) \\
\kappa_k = \exp \left( \alpha_k D_k^{-1} C_i \right) \\
\beta_k = \beta_k (i) + \mu_k D_k^{-1} C_i \left[ (1 - \kappa_k(i)} \right] h_k
\]

**Update Kalman-Consensus Estimate**

\[
\hat{x}_{j,k}^+ = \hat{x}_{j,k}^- + D_k^{-1} \left[ y_j - S_j \hat{x}_{j,k}^- + \epsilon \sum_{l \in N_j} (\hat{x}_{l,k}^- - \hat{x}_{j,k}^-) \right] \\
\hat{x}_{j,k}^- = A_k \hat{x}_{j,k}^+ + B_k u_k \\
\hat{p}_{k+1}^- = \exp \left( \beta_{k+1} \right)
\]

6.3.1 Example

Consider a physical phenomenon, such as weather, propagating on a planar surface according to the partial differential equation

\[
\ddot{u} = \left( \alpha \frac{\partial^2 u}{\partial x^2} + \beta \frac{\partial^2 u}{\partial y} \right).
\]
As studied in [34], and [35], by discretizing the surface into an equally-spaced $n \times n$ grid in the $x$ and $y$ directions and the temporal dimension, thus obtaining a grid of values $u_{ij}^k$, one can construct the linear system representation

$$\hat{u} = \frac{2}{\Delta x^2} I_n \otimes (\alpha D) + (\beta D) \otimes I_n \tilde{u}, \quad \tilde{u} = \text{vec}(u_{ij}^k)$$

where $D$ is a tridiagonal matrix with 1’s on the sub and superdiagonal, and −2’s on the diagonal. The discrete-time state matrix $A$ can be found using a matrix exponential, or a finite-order Taylor expansion. Periodic boundary conditions may be added by setting $D_{1n} = D_{n1} = 1$. Suppose that this phenomenon is being measured according to this discretization. We use a sensor model such that each node collects data from an $n_l \times n_l$ square portion of the grid $u_{ij}^k$ so that each individual grid point is measured by at least one sensor. The sensor network by the graph in Figure 6.2 is used, with $n_l = 4$ for a 50×50 grid, and the sensor noise is given unit covariance. Five 2D Gaussian functions are used to set the initial conditions for the simulation, which was run over a time interval of 10 seconds. The true state of the system at 1.2s is shown in Figure 6.3, and an aggregate measurement of the entire system is shown in Figure 6.4. An estimate from a single sensor at this timestep is shown in Figure 6.5, and one can see that there is a substantial reduction of noise.

6.4 GRADIENT DESCENT FOR KALMAN FILTER COVARIANCE UPDATE

Sutton’s Gradient Descent for updating the error covariance matrix approximation $\hat{P} = \text{diag}(\hat{p}^{(ii)}_k) = \text{diag}(\exp \beta^{(i)}_k)$ uses gradient descent on $\beta^{(i)}_k$:

$$\beta^{(i)}_{k+1} = \beta^{(i)}_k - \frac{1}{2\mu} \frac{\partial \left( s_k^T \delta_k \right)}{\partial \beta^{(i)}_k},$$
6.4 Gradient Descent for Kalman Filter Covariance Update

Figure 6.3: True state of system at $t = 1.2s$

Figure 6.4: Sample measurement $\hat{y}$ over the entire system at $t = 1.2s$

Figure 6.5: Sample system estimate at $t = 1.2s$ from a single sensor
where $\delta_k = \hat{y}_k - C\hat{x}_k$ is the error. To derive the exact form of the gradient descent on $\beta_k^{(i)}$, consider the following terms:

$$
\frac{\partial}{\partial \hat{x}_k^{(i)}} \left( \delta_k^T \delta_k \right) = 2\delta_k^T \frac{\partial \delta_k}{\partial \hat{x}_k^{(i)}} = -2\delta_k^T C[i].
$$

Next, we have

$$
\frac{\partial \delta_k}{\partial \beta_k^{(i)}} = \frac{\partial}{\partial \beta_k^{(i)}} [\hat{y}_k - C\hat{x}_k] = - \sum_{j=1}^{n} C[j] \frac{\partial \hat{x}_k^{(j)}}{\partial \beta_k^{(i)}}.
$$

We assume that for $i \neq j$,

$$
C[j] \frac{\partial \hat{x}_k^{(j)}}{\partial \beta_k^{(i)}} \approx 0 \Rightarrow \frac{\partial \delta_k}{\partial \beta_k^{(i)}} \approx -C[i] \frac{\partial \hat{x}_k^{(i)}}{\partial \beta_k^{(i)}}.
$$

Next, define the following two quantities:

$$
D_k := R_k + CP_k C^T = R_k + \sum_{i=1}^{n} \exp \left( \beta_k^{(i)} \right) C[i] C[i]^T
$$

$$
k_k^{(i)} := \exp \left( \beta_k^{(i)} \right) D_k^{-1} C[i].
$$

Next, we compute the following derivatives:

$$
\frac{\partial D_k}{\partial \beta(j)} = \sum_{i=1}^{n} \frac{\partial \beta_k^{(i)}}{\partial \beta(j)} C[i] C[i]^T = \beta_k^{(j)} C[j] C[j]^T.
$$

$$
\frac{\partial D_k^{-1}}{\partial \beta(j)} = -D_k^{-1} \frac{\partial D_k}{\partial \beta(j)} D_k^{-1} = -k_k^{(i)} C[j] D_k^{-1}
$$

$$
\frac{\partial k_k^{(i)}}{\partial \beta(j)} = \left[ \frac{\partial \beta_k^{(i)}}{\partial \beta(j)} D_k^{-1} + e_k^{(i+1)} \frac{\partial D_k^{-1}}{\partial \beta(j)} \right] C[i]
$$

$$
= \beta_k^{(i)} D_k^{-1} C[i] - e_k^{(i+1)} k_k^{(i)} C[i] D_k^{-1} C[i]
$$

$$
= k_k^{(i)} - k_k^{(i)} \left( \beta_k^{(i)} D_k^{-1} C[i] \right) C[i]
$$

$$
= k_k^{(i)} - k_k^{(i)} \left( \beta_k^{(i)} D_k^{-1} C[i] \right)^T C[i].
$$
Now, we can write the gradient descent on $\beta^{(i)}$:

$$p_{k+1} = p_k - \frac{1}{2\mu} \frac{\partial (\delta_k^T \delta)}{\partial \beta^{(i)}}$$

$$= p_k - \frac{1}{2\mu} \sum_j \frac{\partial (\delta_k^T \delta)}{\partial \tilde{\beta}^{(j)}} \frac{\partial \tilde{\beta}^{(j)}}{\partial \beta^{(i)}}$$

$$= p_k - \frac{1}{2\mu} \frac{\partial (\delta_k^T \delta)}{\partial \tilde{\beta}^{(i)}}$$

$$= p_k + \mu \delta_h C^{(i)} \frac{\partial \tilde{\beta}^{(i)}}{\partial \beta^{(i)}}.$$

All that remains is to find an approximation to $\frac{\partial \tilde{\beta}^{(i)}}{\partial \beta^{(i)}}$. We denote this approximation with $h^{(i)}_{k+1}$:

$$h^{(i)}_{k+1} \approx \frac{\partial \tilde{\beta}^{(i)}}{\partial \beta^{(i)}} = \frac{\partial}{\partial \beta^{(i)}} \left[ \tilde{x}_k^{(i)} + k_k^{(i)T} \delta_k \right]$$

$$= \frac{\partial \tilde{x}_k^{(i)}}{\partial \beta^{(i)}} + \frac{k_k^{(i)T}}{\partial \beta^{(i)}} \frac{\partial k_k^{(i)T}}{\partial \beta^{(i)}} \delta_k$$

$$= \frac{\partial \tilde{x}_k^{(i)}}{\partial \beta^{(i)}} + \left( k_k^{(i)} - k_k^{(i)T} C^{(i)} \right)^T \delta_k$$

$$= \tilde{P}_k \left( 1 - k_k^{(i)T} C^{(i)} \right) \delta_k.$$

This yields the final gradient descent update:

$$p_{k+1} = p_k + \mu \delta_h C^{(i)} h_k^{(i)}.$$

In conclusion, the Kalman filter with gradient descent error covariance update is given by the equations

$$\hat{x}_{k+1} = \hat{x}_k + K_k \left[ \hat{y}_k - C \hat{x}_k \right]$$

$$K_k = \tilde{P}_k C^T \left[ \tilde{C} \tilde{P}_k C^T + R_k \right]^{-1}$$

$$\tilde{P} = \text{diag} \left( e^{\beta_{k+1}^{(i)}} \right)$$

$$p_{k+1} = p_k + \mu \delta_h C^{(i)} h_k^{(i)}$$

$$h_{k+1} = h_k \left( 1 - k_k^{(i)T} C^{(i)} \right) + \left( k_k^{(i)} - k_k^{(i)T} C^{(i)} \right)^T \delta_k.$$

### 6.5 Conclusions and Future Works

In this part, we characterized the controllability and stabilizability of signed consensus networks in the linear and nonlinear settings. We showed that the topological notion of structural balance is the significant condition that appears when making statements on uncontrollability. In particular, structural
balance induces a gauge transformation that permits the extension of the classical controllability and stabilizability analysis of consensus networks to be applied to the signed Laplacian case.

We examined the role of structural balance both in the SISO and MIMO cases of the leader-follower consensus dynamics, and then extended this to the output controllability and stabilizability of the influenced consensus dynamics. We then looked at the task of identifying the bipartite consensus of certain classes of nonlinear network flows. In particular, we showed that the sign structure of the first Koopman mode corresponds to this bipartite consensus, and then used Extended Dynamic Mode Decomposition to numerically approximate this sign structure. Lastly, we considered estimation problems on large-scale consensus networks for the sensor fusion of high-dimensional data. Future work will determine bounds on the amount of data required to have an accurate approximation of the Koopman mode with respect to sign structure.
Part III

DESIGN
INRODUCTION

Networked systems are pervasive in nature, and therefore in many fields of science and engineering. Such systems arise for example in robotic swarms [53], animal flocking [6], oscillator networks [54], infrastructure systems [55], and opinion dynamics [3]. The interplay between modelling and synthesizing such systems has led to the general area of networked dynamic systems—an active area of research in systems and control community. A great amount of effort has recently been focused on understanding how the connection structure, or topology, of a network affects the behaviour or performance of a dynamical process on that network. To that end, a natural question to ask is how one can systematically construct a network topology such that a certain performance metric defined over that behaviour is satisfied.

A well-known method for constructing networks is the method of preferential attachment, where nodes are added to the network and connected to already-existing nodes with a probability proportional to the degree of those nodes [56]. The advantage of this method is that it produces networks with power-law degree distributions\(^1\) that resemble networks found in nature [57]. Another method for growing networks uses Kolmogorov-Sinai entropy as a heuristic parameter for evolving networks into ones with regular, random, scale-free or star topologies [58].

An area of recent focus is the study of controlling distributed systems. Networks arise naturally in the context of dynamics defined over distributed systems, and therefore the question of how the network topology affects network behaviour can be naturally extended to the question of how the network topology affects the controllability of the dynamics over that network. In particular, the consensus dynamics where agents perform distributed averaging over time, has attracted vigorous research [5], [33], [59]. Consensus has diverse applications in control and estimation, such as multi-agent systems [12], [13], distributed Kalman filtering [38], [60], and swarm deployment [61]. System-theoretic properties of the consensus algorithm have also been extensively examined in the literature. The fundamental algebraic object facilitating these studies is the graph Laplacian [5]. Using this setup allows one, for example, to draw connections between systems aspects of distributed algorithms and the graph structure, e.g., highlighting the role of symmetries in inducing uncontrollable modes [17]. Such a point of view has been extended to nonlinear variants of the protocol [33], and graphs admitting both positive and negative interactions [24], [62], [63]. Necessary and sufficient conditions for controllability and observability of linear consensus protocol have been explored using the notion of fractional automorphisms [18], [62]. O’Rourke and Touri discussed the controllability of random graphs in the context of adjacency dynamics [64].

\(^1\) So-called scale-free networks
More research has focused on how one may systematically construct a network that has favorable characteristics for consensus. Ghosh and Boyd developed an algorithm to select connections between agents in a network to maximize the connectivity of the network, and therefore reduce the time it takes for the agents to achieve consensus [65]. Chapman and Mesbahi showed how to construct large networks from graph products of smaller, atomic networks and discussed conditions under which these products preserve controllability [21], [66]. Yazicioglu and Egerstedt developed a method for decomposing and reconstructing graphs for leader-follower selection [67].

When designing a network graph, one has several objects to play with in order to achieve various performance characteristics. In particular, a graph consists of nodes connected by edges with weights, and input vectors indicating where one injects a control signal into the network. As discussed earlier, one example is an algorithm to select edges in a graph to maximize connectivity [65]. There has also been recent work in using submodular optimization for picking input vectors [68], [69]. An excellent summary of submodular optimization applications to the control of networked systems is given in [70]. Measures by which one may gauge network performance and specify network topologies have been suggested in [71], [72]. These include information-theoretic measures such as entropy [71], [73] and other so-called spectral measures [5], [71], as well as disturbance rejection cost in the form of the $H_2$ norm and coherence [66], [74]–[78]. Since consensus is a distributed protocol over a graph, one can modify the topology of the network to adapt to disturbances and antagonistic influences. Algorithms for rewiring the network are proposed in [66], and those for reweighting the network for optimal noise rejection are discussed in [75]. Most of the existing literature on the control and performance analysis for consensus consider the case where each node in the network has a scalar state.  

Reasoning about the performance of agents operating over a network also requires consideration of the individual agent dynamics. The majority of literature on consensus systems focuses on the case where the agent dynamics are taken to be identical single or double integrators. The analysis of multi-scale problems has historically offered techniques for formal description and controller synthesis for complex system dynamics [79], which motivates us to consider agents operating on distinct, individual time scales as part of our consensus formulation. This extends the consensus formulation and the framework for distributed control into a specific class of heterogeneous agents, which has the potential to increase the applicability of the protocols previously developed, as well as the graph-theoretic performance metrics previously discussed.

The integration of multiple agent time scales into the consensus formulation is not seamless, and there is a growing body of literature that addresses various complications with the introduction of slow and fast agents. This started with the discussion of the consensus value for multi-rate integrators in [80]. Since then issues such as convergence [81], stability [82], [83], controller design [84], [85], as well as single-influenced consensus performance [86] for networks featuring multi-time scale behavior have been ad-
dressed. This existing literature has demonstrated that the inclusion of time scales can have a significant impact on the fundamental system-theoretic aspects of a networked system, but that in many cases it is still possible to analyze the system through a graph-theoretic lens.

Recently, several extensions to the case of vector-valued node states have been proposed; such consensus protocols are characterized by having graphs with matrix-valued weights. System-theoretic properties of matrix-valued weighted consensus networks, as well as applications to bearing-constrained formation control have been considered in [87]. A consensus-based model of opinion dynamics on matrix-valued weighted graphs was developed in [88], [89]. Further applications of matrix-valued weighted graphs are found in the control of coupled oscillators [90], spacecraft formation control [91], as well as in attitude estimation [92].

An interesting extension to the theory of electrical networks arises in the context of an estimation problem over a matrix-valued weighted graph. It is well-known that the graph Laplacian plays a central role in resistor networks; it is the fundamental object that encodes effective resistance across the network [93]. An extension of electrical networks to matrix-valued resistances, voltages and currents was examined in the aforementioned estimation problem with relative measurements between the agents [94]. In this line of work, the effective matrix-valued resistance was shown to be related to the covariance matrix of the estimation procedure; in a related work, the scalar effective resistance was shown to be related to the \( \mathcal{H}_2 \) performance of consensus [66], [75]. One of the contributions of this chapter is extending the resistor network formalism to the computation of \( \mathcal{H}_2 \) performance on matrix-valued weighted consensus-type networks.

The notions of performance and control of consensus rely on the underlying topology of the network. As such, it is desirable to devise algorithms for system theoretic analysis and synthesis that are scalable and modular. The combinatorics literature has developed several useful models for graph design, such as the scale-free network [56] and the Erdos-Renyi random graph model [95]. An axiomatic approach that can be adopted in this direction is to take smaller, atomic elements and build large-scale graphs from them; one can then use graph-growing operations that preserve properties such as controllability [22], [96]. In the similar vein, spectral properties of Cartesian product networks have proven useful for systems analysis, where certain properties of the atomic elements translate to the entire network [45], [55], [97]–[99].

In this chapter, one of our approaches to address the performance problem on leader-follower consensus utilizes the paradigm of series-parallel networks. This is a class of graphs characterized by the absence of a subgraph homeomorphic to \( K_4 \). An important property of series-parallel networks is that they can be sequentially decomposed into smaller networks via simple operations in linear [100] and sublinear [101] time. Many NP-hard problems on general classes of graphs become linear on series-parallel graphs, such as finding maximum matchings induced subgraphs and independent sets, and the maximum disjoint triangle problem [102]. The decomposition of series-parallel networks has been exploited in a number of other disciplines. For example, the Quadratic Assignment Problem is solvable in polynomial time on
series-parallel networks [103]. Efficient algorithms also exist for the discrete
time-cost tradeoff problem [104], routing games and selfish routing [105],
and resource allocation by dynamic programming [106]. As such, it is nat-
ural to conjecture that a number of difficult system-theoretic problems on
networks can be solved or scaled efficiently on series-parallel networks. We
do so in this chapter, focusing on the problem of efficiently computing the
$H_2$ performance of leader-follower consensus networks.

Whiskering is a process for growing graphs where at each iteration, a
vertex and edge is connected to every node in the graph. In this chapter, we
discuss using this process, and generalizations of this process, to construct
large graphs that are controllable.

The contributions of this chapter are the following. We extend the use of
submodular optimization in network science to problems involving adding
nodes to the network. We present a graph-growth method that preserves
controllability of consensus on the graph, and provide relevant bounds on
the network performance. Furthermore, we develop a graph-growth algo-
rithm, and formulate convex optimization problems which we then solve
for specific test cases.

Next, by exploiting the structure of series-parallel networks, we present
a way of computing the $H_2$ norm of a leader-follower consensus network
in best-case $O(k\omega|R|\log|N|)$ (worst case $O(k\omega|R||N|)$) complexity, where
$|R|,|N|$ are the number of leaders and followers, respectively, and $O(k\omega)$ is
the complexity of inverting a $k \times k$ symmetric positive-definite matrix; the
current best lower bound for $\omega$ is 2.3728639 [107]. We assume that the node
state vector size $k$ is small relative to the size of the network, and hence, the
complexity of this computation is dominated by the number of nodes $|N|$.

We also provide a gradient descent method for adaptively re-weighting
the network to optimize $H_2$ performance that utilizes computations of sim-
ilar complexity. Using the matrix-valued resistances and currents extension
of electrical networks, we show that the analogous notion of effective re-
sistance is related to the $H_2$ norm on a leader-follower consensus network
with vector states. Again, the decomposition of series-parallel networks will
be used to efficiently compute the parameters in the proposed gradient al-
gorithm.

Additionally, we address the network design problem in the case where
the agents have different time scales. Essentially, a time scale refers to how
‘fast’ an agent integrates information from its neighbours. We explore an
optimal distribution of time scales over agents in the setting of the $H_2$ per-
formance problem. The issue arising from the introduction of time scales is
that the effective system matrix (the Laplacian with time scales) is no longer
symmetric – this means we cannot derive an analytic form of the $H_2$ norm
with standard techniques.

Instead, we utilize the notion of edge consensus. Instead of considering the
‘performance’ of the dynamics on the node states, we derive the dynamical
system corresponding to information flow on edges between nodes. In
the case of systems with time scales, this formulation has a symmetric edge
Laplacian. We examine the design problem in this setting, and show that a
separation principle exists – one can optimize the edge weights and time
scales separately to achieve good $H_2$ performance.
Lastly, we examine notions of ‘entropy’ on networks. These are measures that related to the complexity of networks, and more ‘complex’ networks have higher entropy. We show that these measures have a system-theoretic interpretation, making them amenable to optimization.

The part is organized as follows. Mathematical preliminaries are discussed in §7.1. We examine whiskering and its generalizations in Chapter 8. In Chapter 9, we look at the $\mathcal{H}_2$ performance of matrix-weighted series-parallel networks, and then expand this to matrix-weighted networks with time scales in the edge consensus setting in Chapter 10. Network entropy is discussed in Chapter 12, and the part is concluded in §12.6.

7.1 MATHERNICAL PRELIMINARIES

A few of the main results of this chapter involve analyzing the controllability of the linear consensus dynamics. Theorem 15 provides a well-known and useful tool for testing controllability by looking at uncontrollable modes from the eigen-decomposition of the system.

**Theorem 15 (Popov-Belevitch-Hautus Test \([108]\)).** Consider the linear ordinary differential equation

\[ \dot{x} = Ax + Bu. \]

The system pair $(A, B)$ is uncontrollable if and only if there is a non-zero left eigenvector of $A$ orthogonal to the columns of $B$. In other words $(A, B)$ is uncontrollable if and only if there exists $w \neq 0$ such that

\[ w^T A = \lambda w^T, \quad w^T B = 0. \]

7.1.1 Submodular Optimization

In this subsection, we outline some basic definitions from submodular optimization. Let $m \in \mathbb{Z}_+$, and $[m] := \{1, \ldots, m\}$. We call a real-valued function $f : 2^{[m]} \to \mathbb{R}$ nondecreasing if for sets $J \subset K \subset [m],$

\[ f(J) \leq f(K). \]

The function $f$ is submodular if for subsets $J, K \subset [m]$, we have that

\[ f(K) + f(J) \geq f(K \cup J) + f(K \cap J). \]

Furthermore, $f$ is nonincreasing if $-f$ is nondecreasing, and $f$ is supermodular if $-f$ is submodular. $f$ is modular if it is both supermodular and submodular.

A matrix $H$ is Hermitian if $H = H^\dagger$, where $^\dagger$ denotes the conjugate-transpose operation. Let $\Lambda_E$ be a finite interval of $\mathbb{R}$, and thereby denote the set of $n \times n$ Hermitian matrices with eigenvalues contained in $\Lambda_E$ as
$H_n(\Lambda_E)$. We say that $f$ is operator monotone on $\Lambda_E$ if for all $n \geq 1$ and for all $A, B \in H_n(\Lambda_E)$,

$$A \succeq B \implies f(A) \succeq f(B).$$

Lastly, let $A[K]$ denote the principal submatrix of $A$ obtained by deleting the rows and columns of $A$ corresponding to the elements in the set $[m] \setminus K$.

Recent work in submodular optimization has examined matrix functions (of say, $A$), such as the trace or the trace of powers of matrices, in the context of submodularity over a set $K \subseteq [n]$ on the principal submatrices $A[K]$ [109], [110]. We summarize the main general result of these works, as well as a more specific result about submodularity of functions over principle submatrices of $M$-matrices.

**Theorem 16** ([109], [110]). Let $f$ be a real continuous function on an interval $\Lambda_E$ of $\mathbb{R}$. Furthermore, let $f'$ be operator monotone on the interior of $\Lambda_E$. Then, for all $A \in H_n(\Lambda_E)$, the map from $2^{[n]} \rightarrow \mathbb{R}$

$$K \rightarrow \text{Tr} f(A[K])$$

is supermodular.

**Theorem 17** ([109]). Let $A$ be an $M$-matrix of size $n \times n$. Then, for all subsets $J, K \subseteq [n]$ we have that for $0 \leq p \leq 1$:


and for $1 \leq p \leq 2$ and for $p < 0$,


In particular, the map $K \rightarrow \text{Tr} A[K]^{-1}$ is supermodular if $A$ is an $M$-matrix.
8.1 Graph Growing with Submodular Optimization

In this section, we introduce a method of growing graphs that preserves controllability, by adding a leaf to every node of the graph. We then generalize this process to adding more complicated structures.

Graph whiskering is a process for adding nodes to a graph, originally studied for the purpose of looking at monomial ideals [111], [112]. For each whiskering iteration, a single unique node is connected to an already existing node in the graph, for all nodes in the graph. This corresponds to concatenating the Laplacian in the following form:

\[
\begin{bmatrix}
L_G + I & -I \\
-I & I
\end{bmatrix} = L_{G'},
\]

where we denote the operation as \(L_{G'} = W_1(L_G)\). Figure 8.1 shows an example of whiskering a graph three times. This process is of great interest because of several properties that make it useful for control theory, namely it preserves controllability and provides guarantees on the performance of control exerted on the resulting network. The former property is captured in the following theorem.

**Theorem 18.** Let \(L_{G'} = W_1(L_G)\). The pairs 
\[(L_{G'}, [b^T, b^T]^T)\text{ and } (L_{G'}, [b^T, 0^T]^T)\]
are controllable if and only if the pair \((L_G, b)\) is controllable.

**Proof.** We prove the contrapositive using the Popov-Belevitch Hautus test (Theorem 15). Suppose that \((L_G, b)\) is uncontrollable. Then, there exists \(w\) such that \(w^T b = 0\) and \(w^T L_G = \lambda L_G\). We show by construction that there exists a left eigenvector of \(L_{G'}\) that is orthogonal to the columns of \([b^T, b^T]^T\) and of \([b^T, 0^T]^T\).

We claim that \([w^T, \alpha^T]^T\) is an eigenvector of \(L_{G'}\) with eigenvalue \(\Lambda\), where

\[
\alpha = \frac{1}{1 - \lambda} w
\]

\[
\Lambda = \frac{1}{2} \left(\sqrt{\lambda^2 + 4 + \lambda + 2}\right).
\]
The Laplacian is symmetric, and so its left eigenvectors are transposed right eigenvectors. Therefore, a computation yields

\[
\begin{bmatrix}
L_g + I & -I \\
-I & I
\end{bmatrix}
\begin{bmatrix}
w \\
\alpha
\end{bmatrix}
= \begin{bmatrix}
(L_g + I)w - I\alpha \\
I\alpha - Iw
\end{bmatrix}
= \begin{bmatrix}
(\lambda + 1)w - \alpha \\
\alpha - w
\end{bmatrix}
= \begin{bmatrix}
\frac{(1 - \Lambda)(1 + \lambda) - 1}{1 - \Lambda}w \\
\frac{1}{1 - \Lambda}w
\end{bmatrix}
= \begin{bmatrix}
\Lambda w \\
\alpha
\end{bmatrix}
= \begin{bmatrix}
\Lambda w \\
\Lambda \alpha
\end{bmatrix}.
\]

This is orthogonal to the columns of \([b^T, b^T]^T\):

\[
[w^T, a^T]
\begin{bmatrix}
b \\
\frac{1}{1 - \Lambda}w
\end{bmatrix}
= \begin{bmatrix}
[w^T, \frac{1}{1 - \Lambda}w^T] \\
\frac{1}{1 - \Lambda}
\end{bmatrix}
= \left(1 + \frac{1}{1 - \Lambda}\right) w^T b = 0.
\]

\([w^T, a^T]\) is also orthogonal to the columns of \([b^T, 0^T]^T\):

\[
[w^T, a^T]
\begin{bmatrix}
b \\
0
\end{bmatrix}
= w^T b = 0.
\]
For the reverse direction, assume that \((L_G', b = [b_1^T, b_2^T]^T)\) is uncontrollable. Then by Theorem 15 we have an eigenvector of \(L_G'\) orthogonal to the columns of \(b\):

\[
\begin{bmatrix}
L_G + I & -I \\
-I & I
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2
\end{bmatrix}
= \lambda
\begin{bmatrix}
w_1 \\
w_2
\end{bmatrix}
= \begin{bmatrix}
L_G w_1 + (w_1 - w_2) \\
w_2 - w_1
\end{bmatrix}.
\]

It follows that

\[
w_2 = \frac{1}{1 - \lambda} w_1,
\]

and so \(w_1\) is an eigenvector of \(L_G\) with eigenvalue \((1 - \lambda)(1 - (1 - \lambda)^2)\). It is clear that from Equation (8.2) that \(w_1^T b_1 = 0\), and so \((L_G, b_1)\) is uncontrollable. This result also holds when \(b_2 = 0\), and the theorem follows. \(\square\)

Theorem 18 therefore provides a useful way of ensuring that a large graph is controllable. Since rank controllability tests for very large graphs are often inaccurate due to machine precision or are computationally expensive, one can use this method by starting with a small graph on which a controllability test is easily performed and iterate this process several times until a sufficiently large network is obtained.

A natural question to ask is what other types of growth processes preserve controllability? For example, what if we attach a leaf to every node, but also attach a path of length 2 to every node (as shown in Figure 8.2)? This corresponds to concatenating the Laplacian as follows:

\[
L_G \rightarrow \begin{bmatrix}
L_G + 2I & -I & -I & 0 \\
-I & I & 0 & 0 \\
-I & 0 & 2I & -I \\
0 & 0 & -I & I
\end{bmatrix} = L_G'.
\]
Let \( L_G' = W_2(L_G) \). The pairs \((L_G', [b^T, b_1^T, b_2^T, b_3^T]^T)\) (where \( b_i \in \{b, 0\} \)) are controllable if and only if the pair \((L_G, b)\) is controllable.

**Proof.** We again prove the contrapositive using Theorem 15. Suppose that \((L_G, b)\) is uncontrollable. Then, there exists \( w \) such that \( w^T b = 0 \) and \( w^T L_G = \lambda L_G \). We show by construction that there exists a left eigenvector of \( L_G' \) that is orthogonal to the columns of \([b^T, b_1^T]^T\) and of \([b^T, 0]^T\). It can be verified in a similar calculation as in the proof of Theorem 18 that \([w^T, \alpha^T, \beta^T, \gamma^T]\) is a left eigenvector of \( L_G' \) with eigenvalue \( \Lambda \), where

\[
\gamma = \frac{1}{1 - \Lambda} \beta, \quad \beta = \left( \frac{1}{1 - \Lambda} - \Lambda - 2 \right)^{-1} w, \quad \alpha = \frac{1}{1 - \Lambda} w
\]

and where \( \Lambda \) satisfies the equation

\[
\lambda + 2 = \frac{\Lambda (\Lambda^3 - 4\Lambda + 2)}{\Lambda^3 - 2\Lambda + 1}.
\]

Note that \( \alpha \) and \( \beta \) are simply scalings of \( w \), and therefore \( \gamma \) is also a scaling of \( w \). Therefore, \( \alpha, \beta \) and \( \gamma \) are all orthogonal to the columns of \( b \):

\[
\alpha^T b = \beta^T b = \gamma^T b = w^T b = 0,
\]

and so it is clear that

\[
[w^T, \alpha^T, \beta^T, \gamma^T][b^T, b_1^T, b_2^T, b_3^T]^T = 0
\]

for \( b_i \in \{b, 0\} \). It follows that \((L_G', [b^T, b_1^T, b_2^T, b_3^T]^T)\) is uncontrollable.

For the reverse direction, assume that the pair \((L_G', b = [b_1^T, b_2^T, b_3^T, b_4^T]^T)\) is uncontrollable. Then by Theorem 15 there exists an eigenvector \( w = [w_1^T, w_2^T, w_3^T, w_4^T]^T \) of \( L_G' \) orthogonal to the columns of \( b \) with eigenvalue \( \lambda \) such that

\[
\begin{bmatrix}
L_G + 2I & -I & -I & 0 \\
-I & I & 0 & 0 \\
-I & 0 & 2I & -I \\
0 & 0 & -I & I
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3 \\
w_4
\end{bmatrix}
= \lambda
\begin{bmatrix}
w_1 \\
w_2 \\
w_3 \\
w_4
\end{bmatrix}
\]

\[
L_G w_1 + 2w_1 - w_2 - w_3 \\
w_2 - w_1 \\
2w_3 - w_1 - w_4 \\
w_4 - w_3
\]

It follows from a simple calculation that \( w_1 \) is an eigenvector of \( L_G \) with eigenvalue

\[
\lambda = \frac{\lambda^4 - 6\lambda^3 + 14\lambda^2 - 14\lambda + 4}{(\lambda - 1)(\lambda^2 - 3\lambda + 1)}.
\]

Since \( w \) is orthogonal to the columns of \( b \), it follows that \( w_1^T b_1 = 0 \), and the theorem follows. \( \square \)
As Theorem 19 suggests, there are many ways to grow graphs such that they remain controllable. In this example, we showed that adding a cluster of nodes, namely a leaf and a path of length 2, to each node preserves controllability. It is a natural question to ask what types of node clusters in general one can place on each node to preserve controllability. The following theorem places some conditions on these clusters.

**Theorem 20 (General Graph Growth).** Let $\mathbf{L}_G$ be an $n \times n$ graph Laplacian, and let $\mathbf{L}_{G\delta}$ be an $n \times n$ graph Laplacian, $C$ be an $r \times r$ matrix, and $B$ be an $n \times r$ matrix (where $r = kn$ for $k \in \mathbb{Z}_+$) such that the matrix $\mathbf{L}_{G'} = \begin{bmatrix} \mathbf{L}_G + \mathbf{L}_{G\delta} & B \\ B^T & C \end{bmatrix}$ is a graph Laplacian. Since we are interested in adding the same cluster to each node, we can write $\mathbf{L}_{G\delta} = sI$, where $s$ is the number of edges added to the node when attaching it to the cluster. We have the following results:

1. Let $b_n, w_1 \in \mathbb{R}^n$ and $b_r, \beta \in \mathbb{R}^r$. Suppose $(\mathbf{L}_{G'}, b = [b_1, b_r])$ is uncontrollable. Then, there exists an eigenvector $W_1 \neq 0$ such that $\mathbf{L}_{G'}W_1 = \Lambda w_1$, $W_1^T b = 0$ where, say $W_1 = [w_1^T, \beta^T]^T$. Then $(\mathbf{L}_G, b_1)$ is uncontrollable when:
   
   a) $(\Lambda I - C)$ is invertible
   b) $w_1$ is an eigenvector of $B(\Lambda I - C)^{-1}B^T$.

2. Suppose $(\mathbf{L}_G, b)$ is uncontrollable. Then, there exists $w \neq 0$ such that $\mathbf{L}_Gw = \lambda w$ with $w^T b = 0$. We thus have that $(\mathbf{L}_{G'}, [b, 0])$ is uncontrollable if
   
   a) There exists $\Lambda \geq 0$ such that:
      
      i. $(\Lambda I - C)^{-1}$ is invertible
      ii. $w$ is an eigenvector of $B(\Lambda I - C)^{-1}B^T$ such that $B(\Lambda I - C)^{-1}B^T w = f(\Lambda)$
      iii. $\Lambda$ satisfies $\Lambda - \lambda - s = f(\Lambda)$.

**Proof.** We prove the two results separately.

1) Assuming the notation in Theorem 20-(1), suppose that $(\mathbf{L}_{G'}, b = [b_1, b_r])$ is uncontrollable. Then, by Theorem 15 there exists an eigenvector $W_1 = [w_1^T, \beta^T]^T \neq 0$ such that $\mathbf{L}_{G'}W_1 = \Lambda w_1, W_1^T b = 0$ yielding:

$$\mathbf{L}_{G'}W_1 = \begin{bmatrix} \mathbf{L}_G + \mathbf{L}_{G\delta} & B \\ B^T & C \end{bmatrix} \begin{bmatrix} w_1 \\ \beta \end{bmatrix} = \begin{bmatrix} (\mathbf{L}_G + sI)w_1 + B\beta \\ B^T w_1 + C\beta \end{bmatrix} = \Lambda \begin{bmatrix} w_1 \\ \beta \end{bmatrix}.$$  \hspace{1cm} (8.4)

Therefore, if $(\Lambda I - C)$ is invertible, the lower entry of the vector in Equation (8.4) gives:

$$\beta = (\Lambda I - C)^{-1}B^T w_1.$$
Then, the first entry of the vector in Equation (8.4) gives
\[ L_G w_1 = \left( (\Lambda - s)I - B(\Lambda I - C)^{-1}B^T \right) w_1. \]
This equation admits \( w_1 \) as an eigenvector of \( L_G \) if the action of \( B(\Lambda I - C)^{-1}B^T \) on \( w_1 \) is to scale \( w_1 \) by a fixed amount. In other words, if \( w_1 \) is an eigenvector of \( B(\Lambda I - C)^{-1}B^T \), then it is an eigenvector of \( L_G \), and since \( w_1^T b_1 = 0 \), the result follows.

2) Assuming the notation in Theorem 20-(2), suppose that \((L_G, b)\) is uncontrollable. Then, there exists \( w \neq 0 \) such that \( L_G w = \lambda w \) with \( w^T b = 0 \).

We seek an admissible solution for the equation
\[
\begin{bmatrix}
L_G + L_G \delta & B \\
B^T & C
\end{bmatrix}
\begin{bmatrix}
w \\
\beta
\end{bmatrix} = \begin{bmatrix}
(L_G + sI)w + B\beta \\
B^T w + C\beta
\end{bmatrix}
= \Lambda
\begin{bmatrix}
w \\
\beta
\end{bmatrix}
\tag{8.5}
\]
in terms of the eigenvalue \( \Lambda \) of \( L_G' \), and the lower part of the eigenvector, \( \beta \). If \((\Lambda I - C)\) is invertible, we can write
\[ \beta = (\Lambda I - C)^{-1}B^T w. \]
From the upper entry of the vector in Equation (8.5), we get the relation
\[ (\Lambda - \lambda - s)I w = B(\Lambda I - C)^{-1}B^T w. \]
Then, if \( w \) is an eigenvector of \( B(\Lambda I - C)^{-1}B^T \), say \( B(\Lambda I - C)^{-1}B^T w = f(\Lambda)w \), then we get an equation for \( \Lambda \):
\[ \Lambda - \lambda - s = f(\Lambda). \]
We add the stipulation that \( \Lambda \) must be admissible: \( \Lambda \geq 0 \) for \( \Lambda \) to be a Laplacian eigenvalue. Finally, it is clear that since \( w^T b = 0 \), we have that \( [w^T, \beta^T][b^T, 0^T]^T = 0 \). \( \square \)

8.1.1 Single Leaf Addition

First, we will consider the addition of a single node to the graph. The new node is attached to an existing node with a single edge, see Figure 8.3 (a).

The operation for adding a single leaf to node \( i \) is algebraically represented by appending a row and column to the Laplacian in the following manner:
\[ L_G \rightarrow \begin{bmatrix}
L_G + e_i e_i^T & -e_i \\
e_i^T & 1
\end{bmatrix} := W_i(L_G). \]
where \( W_i \) denotes the Laplacian of the whiskered graph. In order to apply a PBH-style argument to analyze controllability, we need to look for eigen-
vectors \( V \) and eigenvalues \( \Lambda \) of the Laplacian. We separate the eigenvector into an \( n \)-dimensional vector \( v \) and a scalar part \( \alpha \):

\[
\begin{bmatrix}
\mathcal{L}_G + e_i e_i^T & -e_i \\
-e_i^T & 1
\end{bmatrix}
\begin{bmatrix}
v \\
\alpha
\end{bmatrix}
= \Lambda
\begin{bmatrix}
v \\
\alpha
\end{bmatrix}.
\]  

Equation (8.6) can be expanded into:

\[
(\mathcal{L}_G + e_i e_i^T)v - \alpha e_i = \Lambda v
\]
\[
\alpha - e_i^T v = \alpha \Lambda
\]

Equation (8.8) yields:

\[
\alpha = \frac{e_i^T v}{1 - \Lambda},
\]

and substituting this into Equation (8.7) yields:

\[
\mathcal{L}_G v = \left( \Lambda I + \left( \frac{1}{1 - \Lambda} - 1 \right) e_i e_i^T \right) v.
\]

In the original whiskering process, \( V \) was shown to be described in terms of the eigenvectors of \( \mathcal{L}_G \). We will now derive expressions that show this is not the case for single leaf addition, with the exception of the consensus eigenvector. Rewriting Equation (8.9):

\[
\mathcal{L}_G v = \left( \Lambda I + \left( \frac{1}{1 - \Lambda} - 1 \right) e_i e_i^T \right) v
\]

If \( \Lambda = 0 \), then we get exactly that \( \mathcal{L}_G v = \Lambda v = 0_n \). Since \( v = 1 \) is the eigenvector of \( \mathcal{L}_G \) corresponding to \( \Lambda = 0 \), from Equation (8.8) we get that \( \alpha = [v]_i = 1 \). It follows that \([v]^T, \, \alpha]^T = 1_{n+1}^T \).

Next, from Equation (8.10) we can write:

\[
\mathcal{L}_G v = \Lambda v + \frac{\Lambda}{1 - \Lambda} [v]_i e_i
\]

For this to be an eigenvector of \( \mathcal{L}_G \), we need \((2 - \Lambda)(1 - \Lambda)^{-1} = 1\), but this equality has no solutions for finite \( \Lambda \). It follows that \( v \) cannot be written in terms of an eigenvector of \( \mathcal{L}_G \), with the exception of the consensus eigenvector.

Without relying on the original eigenvectors, we have to find conditions on the new graph that guarantee controllability. Since the added node is always a leaf, we focus on finding conditions on the input matrix that leads to the desired guarantees.

Assuming that the original graph is controllable, by the PBH-test, there is no feasible solution to the linear equations given in Lemma 15; i.e. the
eigenvectors of the Laplacian satisfy the linear inequalities that follow from the infeasibility of the PBH Lemma.

It thus follows that to conserve controllability, the new eigenvectors and input matrix of the graph have to satisfy Lemma 15 and the linear inequalities described therein. The eigenvectors are directly given from the structure of the leaf addition. Therefore, we have only the new element of the extended input matrix $B = [b^T, \beta]^T$ to adjust the result and guarantee controllability of the grown graph.

**Theorem 21** (Summary of Single Leaf Whiskering). Consider the Laplacian-control vector pair $(L_G, b)$ and the corresponding single leaf whiskered pair $(W_i(L_G), [b^T, \beta]^T)$ given by the operation

$$L_G \rightarrow \begin{bmatrix} L_G + e_i e_i^T & -e_i \\ -e_i^T & 1 \end{bmatrix} := W_i(L_G).$$

If $(L_G, b)$ is controllable, then $(W_i(L_G), [b^T, \beta]^T)$ is controllable if and only if

$$\beta \neq |b|_i + \frac{v^T L_G b}{|v|_i - \alpha} \quad (8.11)$$

for all eigenvectors $V = [v^T \alpha]^T \neq 1_{n+1}$ of $W_i(L_G)$, and

$$\beta \neq -\sum_i |b|_i. \quad (8.12)$$

**Proof.** The conditions in Equation (8.11) and Equation (8.12) are derived using the PBH-test (Lemma 15). The pair $(W_i(L_G), B)$ is controllable if and only if $V^T B \neq 0$ for all eigenvectors $V$ of $W_i(L_G)$. Noting that the null space of $W_i(L_G)$ is span$(1_{n+1})$ and that $1_{n+1}$ is an eigenvector of $W_i(L_G)$, this is equivalent to the conditions that

$$(W_i(L_G)V)^T B \neq 0 \text{ and } 1_{n+1}^T B \neq 0.$$ 

The latter condition gives

$$0 \neq 1_{n+1}^T B = 1_{n+1}^T \begin{bmatrix} b \\ \beta \end{bmatrix} = \beta + \sum_i |b|_i,$$

yielding the latter part of the theorem. The former condition gives

$$0 \neq (W_i(L_G)V)^T B$$

$$= \begin{bmatrix} L_G + e_i e_i^T & -e_i \\ -e_i^T & 1 \end{bmatrix} \begin{bmatrix} v \\ \alpha \end{bmatrix}$$

$$= v^T L_G b + ([v]_i - \alpha)(|b|_i - \beta),$$

yielding the first condition of the theorem. \qed
8.1.2 Multi-Leaf Addition

Extending the results of single leaf whiskering, we will now examine conditions on the grown graph when adding multiple nodes to it; see Figure 8.3 (b).

The operation of adding leaves to $1 < k < n$ distinct nodes in the graph (say the first $k$ nodes, without loss of generality) is represented by:

$$L_G \rightarrow \begin{bmatrix} L_G + \sum_{i=1}^{k} e_i e_i^T & -e_1 & \cdots & -e_k \\ -e_1^T & \vdots & & -e_k^T \\ \vdots & I_k & & \vdots \\ -e_k^T & & I_k & \end{bmatrix} := W[k](L_G).$$

As in the preceding section, we search for eigenvectors $V = [v_1^T, v_2^T]^T$ of $W[k](L_G)$, where $v_1 \in \mathbb{R}^n$ and $v_2 \in \mathbb{R}^k$:

$$\begin{bmatrix} L_G + \sum_{i=1}^{k} e_i e_i^T & -e_1 & \cdots & -e_k \\ -e_1^T & \vdots & & -e_k^T \\ \vdots & I_k & & \vdots \\ -e_k^T & & I_k & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \Lambda \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

$$= \left( L_G + \sum_{i=1}^{k} e_i e_i^T \right) v_1 + \sum_{i=1}^{k} [v_2]_i e_i$$

$$v_2 - \begin{bmatrix} [v_1]_1 \\ \vdots \\ [v_1]_k \end{bmatrix}.$$
From the equation representing the $n + 1, \ldots, n + k$ entries of $V$, we can see that
\[
\begin{bmatrix}
v_2 \\
\vdots \\
[v_1]_k
\end{bmatrix} = \frac{1}{1-\Lambda} \begin{bmatrix} [v_1]_1 \\
\vdots \\
[v_1]_k
\end{bmatrix}.
\]

Plugging this into the equation representing the first $n$ entries of $V$, we obtain,
\[
\left(L_\mathcal{G} + \sum_{i=1}^{k} e_i e_i^T \right) v + \sum_{i=1}^{k} \frac{1}{1-\Lambda} [v_1]_i e_i = \Lambda v_1
\]
\[
L_\mathcal{G} v_1 = \Lambda v_1 + \frac{\Lambda}{1-\Lambda} \sum_{i=1}^{k} [v_1]_i e_i.
\]

**Theorem 22** (Summary of Multi-Leaf Whiskering). Consider the Laplacian-control vector pair $(L_\mathcal{G}, b)$ and the corresponding multi-leaf whiskered pair
\[
\left(\mathcal{W}_{[k]}(L_\mathcal{G}), [b^T \beta^T]^T\right)
\]
given by the operation
\[
L_\mathcal{G} \rightarrow \begin{bmatrix}
L_\mathcal{G} + \sum_{i=1}^{k} e_i e_i^T & -e_i & \cdots & -e_k \\
-e_i^T & I_k \\
\vdots \\
-e_k^T
\end{bmatrix} := \mathcal{W}_{[k]}(L_\mathcal{G}).
\]

If $(L_\mathcal{G}, b)$ is controllable, then $(\mathcal{W}_{[k]}(L_\mathcal{G}), [b^T \beta^T]^T)$ is controllable if and only if
\[
v_1^T L_\mathcal{G} b \neq \sum_{i=1}^{k} ([v_1]_i - [v_2]_i) ([b]_i - [\beta]_i)
\]
for all eigenvectors $1_{n+k} \neq [v_1^T, v_2^T]^T := V$ of $\mathcal{W}_{[k]}(L_\mathcal{G})$, and
\[
\sum_{i=1}^{n} [\beta]_i \neq -\sum_{j=1}^{k} [b]_j.
\]

**Proof.** For all eigenvectors $V$ of the multi-leaf whiskered graph $\mathcal{W}_{[k]}(L_\mathcal{G})$, it has to be true that:
\[
0 \neq V^T B,
\]
Since the grown graph is fully connected, the vector of all ones $1_{n+k}$ is still the consensus eigenvector related to the eigenvalue of 0. Thus all other eigen-
vector can be expressed by Laplacian multiplication without loss of generality. Then, the PBH-test yields:

$$(W_{[k]}(L_G)V)^T B \neq 0 \text{ and } 1^n_{n+k}B \neq 0$$

for all eigenvectors $V \neq 1^n_{n+k}$ of $W_{[k]}(L_G)$.

The first condition of the theorem is given by the inequality for all eigenvectors that are not the consensus eigenvector:

$$0 \neq (W_{[k]}(L_G)V)^T B$$

$$= \left( (L_G + \sum_{i=1}^k e_i e_i^T) v_1 - \sum_{i=1}^k [v_1]_i e_i \right)^T b + \left( v_2 - \begin{bmatrix} [v_1]_1 \\ \vdots \\ [v_1]_k \end{bmatrix} \right)^T \beta$$

$$= v_1^T L_G b_1 + \sum_{i=1}^n ([v_1]_i - [v_2]_i) [b]_i - \sum_{i=1}^k ([v_1]_i - [v_2]_i) [\beta]_i$$

$$= v_1^T L_G b + \sum_{i=1}^k ([v_1]_i - [v_2]_i) ([b]_i - [\beta]_i).$$

The second condition of the theorem follows directly from the second inequality given by the PBH Lemma:

$$0 \neq 1^n_{n+k}B = \sum_{i=1}^n [b]_i + \sum_{j=1}^k [\beta]_j.$$

This second inequality nicely describes an important effect in graph growing. Due to the fact that every new node is attached to a single original node, an equal but opposite input flow into these new nodes causes uncontrollability. This result is not constrained to an exactly opposite input in the nodes (i.e., $\beta = -b$), but already appears when the total sum of the new input coefficients is equal to the original.

8.1.3 Fully Whiskered Graph

Examining the result from multi-leaf addition we will briefly discuss the case when adding $k = n$ nodes to the graph, see Figure 8.3 (c). This case has been examined in [22] with consideration of the eigenvectors of the whiskered graph depending on the eigenvectors of the original graph. We will show that the results presented in the previous paragraph are generalizations to the approach in [22] and that the choice of input matrix coefficients can thereby be relaxed.
Given the graph Laplacian $L_G$ and input matrix $b$, for $k = n$ we are looking for the following whiskered Laplacian:

$$W_n(L_G) = \begin{bmatrix} L_G + \sum_{i=1}^{n} e_i e_i^T & -e_1 & \cdots & -e_n \\ -e_1^T & I_n \\ \vdots \\ -e_n^T & -1 \\ -1 & 1 \end{bmatrix} = W(L_G)$$

which is identical to the fully whiskered graph given in Theorem 18. The extended input matrix is $B = [b^T, \beta^T]^T$.

Given that the first half of the eigenvector of the new graph is identical to the eigenvector $v$ of the original graph, the first condition of Theorem 22 becomes:

$$0 \neq \lambda v^T b + \sum_{i=1}^{k} ([v_1]_i - [v_2]_i) ([b]_i - [\beta]_i)$$

When choosing $\beta = b$, this inequality reduces to:

$$0 \neq \lambda v^T b$$

for all eigenvectors not equal to the consensus vector. Therefore, since the graph is fully connected, this inequality is always satisfied. In the second case, $\beta = 0_n$, the condition is:

$$0 \neq \lambda v^T b + \sum_{i=1}^{k} ([v_1]_i - [v_2]_i) [b]_i = \lambda v^T b + (v_1 - v_2)^T b$$

From the original graph we know that $0 \neq \lambda v^T b$ and in [22] it is shown that $v = v_1$, the first element of the new eigenvector is equal to the original eigenvector. Furthermore, it is also shown that:

$$v_2 = \frac{1}{1 - \lambda} v, \quad l = \frac{1}{2} \left( \sqrt{\lambda^2 + 4} + \lambda + 2 \right).$$

Thus, the following inequality has to be satisfied:

$$0 \neq \left( \lambda + 1 - \frac{1}{1 - l} \right) v^T b$$

$$= \frac{1}{2} \left( \sqrt{\lambda^2 + 4} + \lambda + 2 \right) v^T b = lv^T b.$$

Since the coefficient $l$ does not have a root with respect to $\lambda$, this inequality is always satisfied.
For either case of $\beta$, the second condition is obviously always satisfied. When $\beta = 0_n$, then:

$$0 \neq \sum_{i=1}^{n} [b]_i$$

is true if the original graph is controllable. And when $\beta = b$, we similarly get the condition:

$$0 \neq 2 \sum_{i=1}^{n} [b]_i.$$

This shows that both results can be used interchangeably. Furthermore, the new result proven in this section can be used to get more varied input matrices. So long as $\beta$ does not violate the conditions given in Theorem 22, the whiskered graph will be controllable.

### 8.2 Examples

In this section we will give some examples of graph growing that rely on the presented results; as such, the graph growth process preserves network controllability. We will use the given inequalities to determine detrimental input matrices. Then, we will show that controllability is lost using the rank of the respective controllability matrix when selecting these undesirable input matrices.

Consider the Laplacian corresponding to the graph given in Figure 8.4 with control vector $b = [1 \ 0]$. The controllability matrix is given by

$$L_{G} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \ b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ C = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix},$$

which has two linearly independent columns, indicating that the system is controllable. This Laplacian has two eigenvectors with respective eigenvalue:

$$v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \ v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \rightarrow \lambda_i = \{0, 2\},$$

where $v_1$ is the consensus eigenvector. In the next step we grow the graph by attaching a single leaf to the first node. It is obvious that this graph has inherent controllability issues due to the symmetry of the nodes with respect
to the original input node. We will use the results of Theorem 21 to give well stated constraints on the grown input matrix \( B \).

The Laplacian, the input matrix, and the eigenvectors of the Laplacian of the single leaf whiskered graph are given by:

\[
W_1(L_G) = \begin{bmatrix}
2 & -1 & -1 \\
-1 & 1 & 0 \\
-1 & 0 & 1
\end{bmatrix}, \quad B = \begin{bmatrix}
1 \\
0 \\
\beta
\end{bmatrix},
\]

\[
v_1 = \begin{bmatrix}
-2 \\
1 \\
1
\end{bmatrix}, \quad v_2 = \begin{bmatrix}
0 \\
-1 \\
1
\end{bmatrix}, \quad v_3 = \begin{bmatrix}
1 \\
1 \\
1
\end{bmatrix}.
\]

Due to the consensus eigenvector we get the following condition given in equation (8.12) on \( \beta \):

\[
\beta|_{j=3} \neq -\sum_i [b]_i = -1
\]

and from equation (8.11) for the eigenvectors \( v_1 \) and \( v_2 \) we get that:

\[
\beta|_{j=1} \neq [b]_i + \frac{v^T L_G b}{|v|_i - \alpha} = 1 + \frac{1}{3} \begin{bmatrix}
-2 \\
1 \\
1
\end{bmatrix} \begin{bmatrix}
1 & -1 \\
-1 & 1 \\
1 & 0
\end{bmatrix} = 0
\]

\[
\beta|_{j=2} \neq 1 + \begin{bmatrix}
0 & -1 \\
-1 & 1
\end{bmatrix} \begin{bmatrix}
1 \\
1
\end{bmatrix} = 2
\]

So long as \( \beta \neq \{-1, 0, 2\} \), the grown graph will be controllable.

The controllability matrix of \( (W_1(L_G), B) \) with respect to \( \beta \) is given by

\[
C(\beta) = \begin{bmatrix}
1 & 2 - \beta & 6 - 3\beta \\
0 & -1 & \beta - 3 \\
\beta & \beta - 1 & 2\beta - 3
\end{bmatrix}.
\]

When \( \beta = -1 \), we get that

\[
C(-1) = \begin{bmatrix}
1 & 3 & 9 \\
0 & -1 & -4 \\
-1 & -2 & -5
\end{bmatrix}.
\]

The columns are not linearly independent: \(-3c_1 + 2c_2 = c_3\), and so it is uncontrollable. For the other cases the rank deficiency of \( C \) is immediately obvious:

\[
C(0) = \begin{bmatrix}
1 & 2 & 6 \\
0 & -1 & -3 \\
0 & -1 & -3
\end{bmatrix}, \quad C(2) = \begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & -1 \\
2 & 1 & 1
\end{bmatrix}.
\]
In all cases the whiskered graph is uncontrollable.

Given the graph depicted in Figure 8.5 the Laplacian and the input matrix, where the only input is at the first node, are:

\[
L_G = \begin{bmatrix}
1 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 1
\end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.
\]

This graph is controllable with the full rank controllability matrix:

\[
C = \begin{bmatrix}
1 & 1 & 2 \\
0 & -1 & -2 \\
0 & 0 & 1
\end{bmatrix}.
\]

Adding two leaves as shown in Figure 8.5 the whiskered Laplacian and input matrix are as follows:

\[
W_2(L_G) = \begin{bmatrix}
1 & -1 & 0 & 0 & 0 \\
-1 & 3 & -1 & -1 & 0 \\
0 & -1 & 2 & 0 & -1 \\
0 & -1 & 0 & 1 & 0 \\
0 & 0 & -1 & 0 & 1
\end{bmatrix}, \quad B = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}.
\]

The second condition of Theorem 22 gives the following inequality:

\[
\sum_{i=1}^{n} \beta_i \neq -\sum_{j=1}^{k} [b]_j \Rightarrow \beta_1 + \beta_2 \neq -1
\]

As a counter-example, let \( \beta_1 = 0 \) and \( \beta_2 = -1 \). Then, the controllability matrix becomes:

\[
C(0, -1) = \begin{bmatrix}
1 & 1 & 2 & 7 & 29 \\
0 & -1 & -5 & -22 & -94 \\
0 & 1 & 4 & 15 & 58 \\
0 & 0 & 1 & 6 & 28 \\
-1 & -1 & -2 & -6 & -21
\end{bmatrix},
\]

which does not have full rank. Thus, this choice of \( \beta \) will yield an uncontrollable graph.
8.2.1 Optimization Algorithms: Adding Leaves

In this section, we discuss optimization problems that are related to growing graphs. In particular, we consider when one wants to add node clusters to only a specific set of nodes in the graph.

Consider a connected graph $G$ and its Laplacian matrix $L_G$. The second-smallest eigenvalue $\lambda_2(L_G)$ is a measure of how interconnected the graph is. It is also an inverse measure of how long it takes for agents connected with graph $G$ to achieve consensus by convergence to the agreement subspace. A well-known algorithm by Ghosh [65] adds edges between unconnected nodes in $G$ to maximize $\lambda_2(L_G)$.

The algorithm considers a set of candidate edges between unconnected nodes in $G$, and selects the $k$ candidate edges that maximize $\lambda_2(L_G)$. For a set of $m$ candidate edges $l = \{ij\}$, let $a_l$ be the vector with all-zero entries except $a_l(i) = 1$ and $a_l(j) = -1$ when $\{ij\}$ is a candidate edge. The selection of $k$ candidate edges from this set can be encoded with a $\{0,1\}^m$-vector $x$, where $x_l = 1$ if the algorithm selects candidate edge $l$, and zero otherwise. The optimization problem is then written in terms of the individual Laplacians $a_la_l^T$ for each edge $l$ as follows:

\[
\begin{align*}
\text{maximize} & \quad \lambda_2 \left( L_G + \sum_{l=1}^m x_la_la_l^T \right) \\
\text{subject to} & \quad 1^T x = k \\
& \quad x \in \{0,1\}^m.
\end{align*}
\]

The standard relaxation of this problem into a semidefinite program (SDP) is of the form

\[
\begin{align*}
\text{maximize} & \quad s \\
\text{subject to} & \quad s(I - 11^T/n) \preceq L(x) \\
& \quad 1^T x = k \\
& \quad 0 \leq x \leq 1 \\
& \quad L(x) = L_G + \sum_{l=1}^m x_la_la_l^T. \nonumber
\end{align*}
\]

We present a modification of this algorithm whereby one wants to add nodes to the graph $G$ in such a way that the graph grows in order to maximize $\lambda_2(G)$. Suppose $G$ has $n$ nodes. We want to choose one of these $n$ nodes to attach leaves to in order to ‘grow’ the graph to maximize $\lambda_2(G)$. Recall that $L_G[I]$ is the principal submatrix of $L_G$ obtained by deleting the rows and columns of $L_G$ corresponding to the elements in the set $I \setminus [m]$. Let $L_{tot}$ denote the graph that has every node whiskered:

\[
L_{tot} = \begin{bmatrix} L_G + I & -I \\ -I & I \end{bmatrix}.
\]
We can write this problem as
\[
\begin{aligned}
\text{maximize} & \quad \lambda_2(L_{G'}) \\
\text{subject to} & \quad L_{G'} \in \{ L_{\text{tot}}[n] \cup \{ i \}, \ i \in \{ n+1, \ldots, 2n \} \} \\
& \quad L_{\text{tot}} = \begin{bmatrix} L_G + I & -I \\ -I & I \end{bmatrix}.
\end{aligned}
\] (8.13)

This can be solved via exhaustive search over all possible whiskerings, however this becomes computationally intractable for large \( n \). We can relax this problem to a modified Ghosh-Boyd Max-\( \lambda_2(G) \) SDP as follows. Let \( e_i \) denote the \( i \)th standard basis vector in \( \mathbb{R}^n \). Then, we introduce a single node into the system and create a set of \( n \) candidate edges potentially connecting the new node to any pre-existing node in the graph. The individual Laplacian for each candidate edge is \( a_i a_i^T \), where \( a_i \in \mathbb{R}^{n+1} \) is of the form \( a_i = [e_i, -1]^T \). The SDP relaxation is then
\[
\begin{aligned}
\text{maximize} & \quad s \\
\text{subject to} & \quad s \left( I_{n+1} - \frac{(11^T)_{n+1}}{n+1} \right) \preceq L(x) \\
& \quad 1^T x = 1 \\
& \quad 0 \leq x \leq 1 \\
& \quad L(x) = L_{G'}' + \sum_{i=1}^n x_i a_i a_i^T \\
& \quad L_{G'}' = \begin{bmatrix} L_G & 0_n \\ 0_n^T & 0 \end{bmatrix}, \ 0_n \in \mathbb{R}^n.
\end{aligned}
\] (8.14)

Recall that \( L[K] \) is the principal submatrix of \( L \) constructed by removing the rows and columns indexed by the set \( [n] \setminus K \). For example, \( L([n] \setminus \{ 1 \}) = L[2 \ldots n] \) is the principal submatrix of \( L \) with the first row and column removed. This matrix is positive-definite. We can also relate the inverse of this matrix to the controllability Gramian \([71], [75] \) \( P \), which is a measure of the steady-state covariance of the agent states. \( P \) is the positive-definite solution of the Lyapunov equation
\[
\]
and is given by \( P = \frac{1}{2} L[2 : n]^{-1} \). Certain submodular functions of \( P \) (with respect to edge-addition) were studied in \([69] \).

The trace of \( P \) can be interpreted as an average amount of energy expended to move the agent states around the controllable subspace, and therefore it is of interest to be able to bound the value of \( \text{Tr} P \) on the results of our algorithms. We do this using the supermodularity properties of \( M \)-matrices from Theorem 17.
Theorem 23. Let $L'$ denote the whiskering process in Equation (8.1), where $L$ is $n \times n$, and so $L'$ is $2n \times 2n$. Let the controllability Gramian $P'$ of $L'$ be the solution to


Then,

$$\text{Tr}P' \geq n + C,$$

where $C$ is a constant depending on $L$.

Theorem 24. Let $L'$ denote the whiskering process in Equation (8.3), where $L$ is $n \times n$, and so $L'$ is $4n \times 4n$. Let the controllability Gramian $P'$ of $L'$ be the solution to


Then,

$$\text{Tr}P' \geq 4n + C,$$

where $C$ is a constant depending on $L$.

We can prove Theorem 23 and 24 together.

Proof. First, we prove Theorem 23. We know that the solution to


is given by $P' = \frac{1}{2}L'[2:2n]^{-1}$. From Theorem 17, using the fact that $L'$ is an M-matrix, we have that

$$\text{Tr}(P) = \text{Tr}(L'[2:2n]^{-1}) \geq \text{Tr}(L'[2:n]^{-1}) + \text{Tr}(L'[n+1:2n]^{-1}) = \text{Tr}(L[2:n]+I)^{-1} + \text{Tr}(I) = C + n,$$

where $C = \text{Tr}(L[2:n]+I)^{-1}$ depends only on $L$. Theorem 24 follows from an identical calculation, noting that

$$\text{Tr} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2I & -I \\ 0 & -I & I \end{bmatrix}^{-1} = 2\text{Tr}(I) + \text{Tr}(2I) = 4n.$$

□

We can use this result to bound the controllability Gramian when adding a single node to the system.

Theorem 25. Consider the task of attaching a single node to the system with $n \times n$ Laplacian $L$ to maximize $\lambda_2$, as denoted in Problem (8.13). Let $L'$ be the subsequent Laplacian, and so $P = \frac{1}{2}L'[2:n+1]^{-1}$. Then, $\text{Tr}P \geq C + 1$, where $C$ is a constant depending only on $L$.
Proof. Using Theorem 17 we can compute
\[
\text{Tr}(P) = \text{Tr}(L'[2 : n + 1]^{-1})
\geq \text{Tr}(L'[2 : n]^{-1}) + \text{Tr}(L'[n + 1 : n + 1]^{-1})
= \text{Tr}([L[2 : n] + e_i e_i^T]^{-1}) + 1 \geq C + 1,
\]
where \(i\) is the index of the attachment node chosen, and \(C = \min_i (\text{Tr}([L[2 : n] + e_i e_i^T]^{-1}))\) is a constant depending only on \(L\). □

In the next section, we will consider adding a cluster, and provide a similar result on the performance of \(\text{Tr}P\).

8.2.2 Optimization Algorithms: Adding Clusters

In the previous section, we considered the problem of optimally adding leaves to some nodes to optimize the algebraic connectivity of the graph. We will now consider the problem of adding a cluster of a node and a length-2 path, as depicted in Figure 8.2.

Let \(0_n \in \mathbb{R}^n\) and define \(a_{3\rightarrow 4} = [0_n^T, 0, 1, -1]^T\), \(a_{i,1} = [e_i^T, -1, 0, 0]^T\) and \(a_{i,2} = [e_i^T, 0, -1, 0]^T\). Then, choosing an attachment node to maximize \(\lambda_2\) can be written as
\[
\begin{align*}
\text{maximize} & \quad \lambda_2(L_G') \\
\text{subject to} & \quad L_G' \in \{L_{\text{tot}}[n] \cup \{i, i + n, i + 2n\}, \\
& \quad i \in \{n + 1, \ldots, 2n\}
\end{align*}
\]
\[
L_{\text{tot}} = \begin{bmatrix}
L + 2I & -I & -I & 0 \\
-I & I & 0 & 0 \\
-I & 0 & 2I & -I \\
0 & 0 & -I & I
\end{bmatrix}.
\]

(8.15)

We can write the SDP relaxation as:
\[
\begin{align*}
\text{maximize} & \quad s \\
\text{subject to} & \quad s \left(I_{n+3} - \frac{(11^T)n+3}{n+3} \right) \preceq L(x) \\
& \quad 1^T x = 1 \\
& \quad 0 \leq x \leq 1 \\
& \quad L(x) = L_G' + a_{3\rightarrow 4}a_{3\rightarrow 4}^T \\
& \quad \quad + \sum_{l=1}^n x_l (a_{i,1}a_{i,1}^T + a_{i,2}a_{i,2}^T) \\
& \quad L_G' = \begin{bmatrix}
L_G & 0_{n \times 3} \\
0_{3 \times n} & 0_{3 \times 3}
\end{bmatrix}.
\end{align*}
\]

(8.16)

Lastly, we give a performance bound on the Gramian analogous to Theorem 25.
8.3 Algorithm Implementation

In this section, we show examples of the optimization problems discussed in the previous section.

The optimization problems (8.13) and (8.15) were implemented using CVX [113], [114]. An additional relaxation method used to solve problems (8.14) and (8.15) discussed in [65], known as the perturbation heuristic, was also implemented for the purposes of comparison. The perturbation heuristic works by considering the eigenvector \( \mathbf{v} \) of \( L' \) corresponding to \( \lambda_2 \). At each iteration of the algorithm, the node cluster is attached to a single node in the graph. The attachment point is chosen by selecting the node with the largest value of

\[
\left( \mathbf{v}_i \mathbf{v}_{n+1} \right)^2
\]

over all \( i \in [n] \). Here, \( \mathbf{v}_{n+1} \) is the entry of \( \mathbf{v} \) corresponding to the node in the cluster attaching the cluster to the node \( i \).

If there is more than one node attaching the cluster to the node in the graph (for example, the leaf and path addition displayed in Figure 8.2), then without loss of generality, denote these (say, \( l \)) nodes as \( n+1, \ldots, n+l \). Then, the perturbation heuristic is to find the node \( i \in [n] \) maximizing

\[
\sum_{j=1}^{l} (\mathbf{v}_i - \mathbf{v}_{n+j})^2
\]

at each iteration.
Figure 8.7: Seed graph, and final graph after 9 iterations of the leaf-adding problem using the SDP relaxation (8.14), exhaustive search over problem (8.13) and the perturbation heuristic.

Figure 8.8: Seed graph, and final graph after 9 iterations of the path-cluster-adding problem using the SDP relaxation (8.16), exhaustive search over problem (8.15) and the perturbation heuristic.
The results of running these algorithms for 9 iterations are shown in Figures 8.6, 8.7, and 8.8. The seed graphs, and final graphs after 9 iterations for each of the three techniques (exhaustive search, convex relaxation and perturbation heuristic) are shown in Figure 8.7 for adding a single leaf, and in Figure 8.8 for adding the path cluster. For both cases, the convex relaxations (problems (8.14) and (8.16)) perform reasonably well and pick out slightly suboptimal solutions, as seen in Figure 8.6. The perturbation heuristic performs worse than the convex relaxation for adding the leaves, but performs just as well for adding the path cluster despite yielding a different final graph as seen in Figure 8.8.
Next, we discuss consensus over vector-valued states, as well as consensus with different time scales. This requires the notion of matrix-weighted graphs.

This section considers dynamics governed by the interconnections of multirate, single integrator agents connected over connected, weighted communication graphs. Every agent has a vector-valued state \( x \in \mathbb{R}^{k} \). In this formulation, we can consider a graph object defined by \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, W, T) \), where \( \mathcal{V} \) is the set of agents (nodes), \( \mathcal{E} \) is the set of edges, \( W \) is a set of edge weights, and \( T \) a set of time scaling factors for each agent’s state. This setup is expanded below.

Individual agents will be indexed by subscripts, e.g. \( v_i \in \mathcal{V} \) to represent the \( i \)th agent where \( 1 \leq i \leq |\mathcal{V}| \). If \( (i, j) \in \mathcal{E} \), the \( i \)th and \( j \)th agents are connected by an edge \((i \sim j)\), and they are referred to as adjacent agents. For a given agent, \( i \), \( N(i) = \{ j \mid i \sim j \forall j \in \mathcal{V} \} \) denotes the neighbors of \( i \), and \( \text{deg}(v_i) = |N(i)| \) denotes the unweighted degree of \( i \).

Since we are considering vector-valued agent states, edges between agents are matrix-valued. This allows for a generalized notion of dynamical coupling between neighboring agent states. Such matrix-valued weights will be denoted \( W_e \in \mathcal{S}_{++}^{k} \), and so \( W = \{ W_e \mid e \in \mathcal{E} \} \). The edge set can be ordered by a mapping, \( \kappa(\cdot) \), such that \( l = \kappa(ij) \) if and only if \( (i, j) \in \mathcal{E} \). By this mapping, we can denote the weight on edge \( \kappa(ij) \) by \( W_l \) or \( W_{ij} \), interchangeably. Furthermore, we also assume that each individual state of each node can run on a separate timescale. Thus, for each node \( i \) and set of corresponding time scales \( T_i = \{ \epsilon_{i,1}, \ldots, \epsilon_{i,k} \} \in T \), we associate the time scale matrix \( E_i = \text{diag}(\epsilon_{i,1}, \ldots, \epsilon_{i,k}) \in \mathcal{S}_{++}^{k} \). Note that the positive-definiteness of \( E_i \) is equivalent to requiring that each \( \epsilon_{i,k} > 0 \).

Next, we define some algebraic graph-theoretic objects.

The **incidence matrix** \( D(\mathcal{G}) \) is a \( |\mathcal{N}| \times |\mathcal{E}| \) matrix where the rows are indexed by the nodes, and the columns are indexed by the edges of \( \mathcal{G} \). Denote each column of \( D(\mathcal{G}) \) corresponding to edge \( \{i, j\} \) by \( a_{ij} \). For each edge \( l := \{i, j\} \), where \( i \) is the tail and \( j \) is the head, \( D(\mathcal{G})_{li} = 1 \) and \( D(\mathcal{G})_{lj} = -1 \). If \( \mathcal{G} \) is undirected, by convention we write that \( D(\mathcal{G})_{ij} = 1 \) and \( D(\mathcal{G})_{ji} = -1 \) for \( i > j \). Since we are dealing with matrix-valued weights, for defining the graph Laplacian below, we need the matrices \( D(\mathcal{G}) \doteq D(\mathcal{G}) \otimes I_k \) and \( a_{ij} \doteq a_{ij} \otimes I_k \), where \( I_k \) is the \( k \times k \) identity matrix, and \( \otimes \) denotes the Kronecker product. The **weight matrix** \( W \) is a \( |\mathcal{E}| \times |\mathcal{E}| \) blockwise diagonal matrix containing the weights \( W_{ij} \) of each edge \( e \).

The **graph Laplacian** \( L_{\mathcal{G}} \) of an undirected graph \( \mathcal{G} \) can equivalently be defined by the incidence and weight matrix as \( L_{\mathcal{G}} \doteq D(\mathcal{G}) W D(\mathcal{G})^T \) = \( \sum_{ij \in \mathcal{E}} a_{ij} W_{ij} a_{ij}^T \), and blockwise with the \( k \times k \) block whose rows are associ-
ated with the \(i\)th node and whose columns are associated with the \(j\)th node given by

\[
\begin{align*}
\sum_{j \in N_i} W_{ij} & \quad i = j \\
-W_{ij} & \quad ij \in E \\
0_{k \times k} & \quad ij \notin E.
\end{align*}
\]

Finally, a short aside on the terminology in this chapter, as the combination of scaling and weightings can become cumbersome at times. A weighted graph refers to a non-negative weighting on the edges of the graph. Matrices (such as the Laplacian) corresponding to weighted graphs will be denoted with \(w\) subscripts to differentiate them from an unweighted graph on the same topology. A scaled graph refers to a positive scaling on the nodes of the graph. The terminology is derived from the interpretation of the scaling parameters as indicators of the relative time scaling on the individual node dynamics, but they can equivalently be interpreted as a nodal weighting. Matrices associated with scaled graphs will be denoted with a tilde, and graphs with unity scaling on all states of nodes will be referred to as “mono-scale” graphs. In the course of our formulation, these descriptors may “stack” in certain scenarios, that is, we may consider scaled, weighted graphs in the most general cases.

We will also use some additional terminology from combinatorics and graph theory. A tree is a connected graph with no cycles, and a leaf is designated as a node of degree 1. A binary tree is a tree where one node is designated as the root, and all nodes of \(T\) are either leaves or parents. Each parent in a binary tree can have at most two children and one parent, except the root which has no parents. The height \(h\) of a binary tree is the length of the longest path from the root to a leaf. A complete (sometimes called full) binary tree is one where each node has either zero or two children.

Finally, the parallel addition of two symmetric matrices \(A, B\) is defined as \(A : B \triangleq A(A + B)^T B\); when \(A, B\) are nonzero scalars this reduces to the familiar formula of adding two resistances in parallel.

9.1 \textbf{LEADER-FOOLLOWER CONSENSUS NETWORK MODELS}

In this chapter, we examine the leader-follower consensus problem on matrix-valued weighted series-parallel networks. Consider a connected weighted graph \(G = (\mathcal{N}, \mathcal{E}, \mathcal{W})\) with the Laplacian \(L_G\). Each node \(i\) has a state \(x_i \in \mathbb{R}^k\). Denote a set of leaders \(\mathcal{R} \subset \mathcal{N}\) and a set of followers \(\mathcal{N} \setminus \mathcal{R}\). Suppose that one is able to take over the state \(x_i \in \mathbb{R}^k\) of a leader, and thereby exert control over the followers. Further, suppose that each leader is connected to a unique node in \(\mathcal{N} \setminus \mathcal{R}\), that collectively will be called the source nodes and designated as the set \(R\) (see Figure 9.1 for a schematic of the setup). Then, using \(B := B \otimes I_k\), the graph Laplacian of \(G\) can be partitioned as,

\[
L_G = \begin{bmatrix}
    L_R & -W_R B \\
    -B^T W_R & L_G(\mathcal{N} \setminus \mathcal{R}) + \sum_{i \in \mathcal{R}} e_i e_i^T
\end{bmatrix},
\] (9.1)
Figure 9.1: **Top:** leader-follower network setup. **Right:** electrical “grounding” of the leader set \( \mathcal{R} \), and current vector \( e_i = e_i \otimes I_k \) injected into the network via node \( v_i \); \( y_i^* \) is the voltage dropped from \( v_i \) to \( \mathcal{R} \). **Bottom:** Identification of grounded leader set into a single node.

where \( W_R \) is a block-diagonal matrix containing the matrix-valued weights of the edges connecting \( \mathcal{R} \) to \( R \) and \( e_i \equiv e_i \otimes I_k \). Suppose that we can observe the full state of individual follower nodes. Thereby, the control and observation matrices are given by \( B^T = [e_i^1 : : : e_i^m] \) and \( C = [e_j^1 \ldots e_j^o] \), where \( R = \{i_1, \ldots, i_m\} \) are the nodes attached to leaders, and \( \{j_1, \ldots, j_o\} \) are the nodes under observation.\(^1\) Next, recall that one can write the graph Laplacian in terms of the incidence and weight matrices as

\[ L_{\mathcal{G}(N \setminus \mathcal{R})} = E_{N \setminus \mathcal{R}} W E_{N \setminus \mathcal{R}}^T = \sum_{(i,j) \in \mathcal{E}(N \setminus \mathcal{R})} a_{ij} W_{ij} a_{ij}^T, \]

where \( E_{N \setminus \mathcal{R}} = E_{N \setminus \mathcal{R}} \otimes I_k \) and \( E_{N \setminus \mathcal{R}} \) is the result of removing the rows from \( E \) corresponding to the nodes in the leader set \( \mathcal{R} \). The matrix \( W \) denotes the matrix consisting of \( W_e \) on the block-diagonal, denoted \( \text{Blkdiag}[W_e] \). The corresponding leader-follower consensus dynamics are now given by,

\[ \dot{x} = - \left( L_{\mathcal{G}(N \setminus \mathcal{R})} + \sum_{i \in R} e_i W_i e_i^T \right) x + B^T u \]

\[ y = Cx, \]

where \( C = C \otimes I_k \), \( x \) is the vector containing the stacked states of the nodes, and \( u \) is the stacked vector of the leader node states—the control inputs to the followers. We note that the matrix \( A \equiv (L_{\mathcal{G}(N \setminus \mathcal{R})} + \sum_{i \in R} e_i W_i e_i^T) \) is positive definite if the underlying graph is connected.

We now introduce the paradigm of **series-parallel networks**, and how they can be utilized for the leader-follower consensus model. Building on connections to electrical networks and tools from the literature on series-parallel networks, we then proceed to show how one can—rather efficiently—compute

---

1 Note that the form of these matrices assumes that each leader is attached to a single, unique follower.
the $\mathcal{H}_2$ norm of leader-follower consensus, as well as devising efficient network-reweighting algorithms for this class of networks.

### 9.1.1 Electrical Networks

An insightful perspective on consensus comes from viewing the underlying graph as a resistive network [66], [115]. In the leader-follower setup, consider the graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{W})$ with scalar weights. For each edge $(i, j) \in \mathcal{E}$, consider placing a resistor between nodes $i$ and $j$ with conductance $w_{ij}$ (resistance $w_{ij}^{-1}$). Then, the effective resistance between arbitrary nodes $k, l \in \mathcal{N}$ is given by

$$R_{kl} = (e_k - e_l) L_{\mathcal{G}}^{-1}(e_k - e_l),$$

where $L_{\mathcal{G}}^{-1}$ denotes the pseudoinverse of $L_{\mathcal{G}}$.

Another perspective on effective resistance is the following. Consider ‘grounding’ all the leader nodes $r \in \mathcal{R}$, i.e., identifying them as one node, or electrically connecting them together by wires. Then, the diagonal entries of the matrix

$$A^{-1} = L_{\mathcal{G}(\mathcal{N} \setminus \mathcal{R})} + \sum_{i \in \mathcal{R}} w_{ie_i} e_i^T$$

yield the effective resistances between the $i$th node in $\mathcal{N} \setminus \mathcal{R}$ to the leader node set, i.e., $[A^{-1}]_{ii}$ is the effective resistance from $i$ to $\mathcal{R}$. If $x \in \mathbb{R}^{[\mathcal{N}]}$ denotes a vector of currents injected into nodes $\mathcal{N} \setminus \mathcal{R}$ from $\mathcal{R}$, then the quantity $[A^{-1}x]_i$ is the voltage drop from node $v_i$ to the grounded leader node set. If $x = e_s$, then this corresponds to a 1 Amp current injected into the node $v_s \in \mathcal{R}$; see Figure 9.1 for a schematic of this setup. In this case, we write $y^T_s = [A^{-1}e_s]_i$ as the voltage drop of node $i$; this quantity is used subsequently in the chapter.

One can generalize this electrical network interpretation to matrix-value weights [115]. Consider placing a matrix-value resistor on each edge $ij$ with matrix-valued conductance $W_{ij}$. Such a construct is similar to that of an $n$-port network [116]. Then, the $i$th $k \times k$ block on the diagonal of the matrix,

$$A^{-1} = L_{\mathcal{G}(\mathcal{N} \setminus \mathcal{R})} + \sum_{i \in \mathcal{R}} e_i W_i e_i^T$$

form the matrix-valued effective resistances from node $i \in \mathcal{N} \setminus \mathcal{R}$ to $\mathcal{R}$, denoted $R_{\mathcal{R}}^{\text{eff}}(\mathcal{R})$. This generalization extends to other electrical quantities, such as power and current; one can obtain the following formalism from [115].
A generalized current from node \( u \) to \( v \) with intensity \( i \in \mathbb{R}^{k \times k} \) is an edge function \( I : E \rightarrow \mathbb{R}^{k \times k} \) such that

\[
\sum_{(k,l) \in E, k=p} I_{(k,l)} - \sum_{(l,k) \in E, k=p} I_{(l,k)} = \begin{cases} i & p = u \\ -i & p = v \\ 0_{k \times k} & \text{else} \end{cases} \forall p \in \mathcal{N}
\]

and there exists a node function \( V : \mathcal{N} \rightarrow \mathbb{R}^{k \times k} \) satisfying

\[
R_{\text{eff}}^{(u,v)} I_{(u,v)} = V_u - V_v, \quad \forall \{u,v\} \in \mathcal{E}.
\]

In this setting, the power dissipated across an edge with a matrix weight \( R_{\text{eff}} \) and current \( I_e \) is given by the inner product,

\[
P_e(i_e) = \text{Tr}\left( I_e^T R_{\text{eff}} I_e \right).
\]

In §9.2, we show that this notion of electrical power is consistent with adding matrix-valued resistances in series and parallel. We also note that when \( x \in \mathbb{R}^{nk \times k} \) is a matrix of stacked \( k \times k \) current matrices injected into \( n \) nodes of \( \mathcal{G} \), then the matrix \( A^{-1}x \) is the stacked matrix of the voltage drops from each node to the grounded leader node set. In particular, the \( st \) \( k \times k \) block of \( A^{-1}x \) is denoted \( \text{Blk}^{k \times k}_s[A^{-1}x] \), and if \( x = (e_s \otimes I_k) = e_s \), then this corresponds to a current of identity intensity injected into node \( s \); again, this setup is shown in Figure 9.1.

Finally, we denote the quantity

\[
Y^s_i = \text{Blk}^{k \times k}_i[A^{-1}e_s],
\]

as the voltage drop from node \( i \) to \( \mathcal{R} \) under identity current injected into node \( s \); this reduces to the corresponding scalar definition when \( k = 1 \).

As a final remark, there is also an equivalent formulation of (9.2) in the case of matrix-valued resistances.

**Proposition 1.** Let \( \mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{W}) \) be a graph with matrix-valued weights. Consider the leader-follower setup in §9.1, with graph Laplacian \( \mathcal{L}_G = EW^T \). Then, the matrix-valued effective resistance between nodes \( i \) and \( j \) is given by

\[
R_{\text{eff}}^i(j) = (e_i \otimes I_k - e_j \otimes I_k)^T \mathcal{L}_G^+ (e_i \otimes I_k - e_j \otimes I_k)
= (e_i - e_j)^T \mathcal{L}_G^+ (e_i - e_j).
\]

In subsequent sections, we introduce series-parallel graphs. In particular, we show how \( y^s_i \), and its matrix-valued generalization \( Y^s_i \), can be computed efficiently over series-parallel graphs, and how it is related to the \( \mathcal{H}_2 \) norm. We will then proceed to show how this setup can be utilized to provide a reweighting algorithm that results in optimal \( \mathcal{H}_2 \) performance of the leader-follower consensus network.
9.1.2 Series-Parallel Networks

In this chapter, we consider the class of graphs known as series-parallel graphs. This class has a number of interesting properties that can be utilized for their system-theoretic analysis. Given a series-parallel graph there exist rather efficient ($O(|N| + |E|)$, $O(\log |N|)$ and $O(\log^2 |N|)$) algorithms that decompose the graph into atomic structures and simple composition operations on them [100], [101], [117]. This decomposition in turn facilitates efficient solutions to problems that are otherwise NP-hard on general classes of graphs, such as finding the minimum dominating set of a graph, maximum matchings and independent sets, and the maximum disjoint triangle problem [102], [118], [119].

**Definition 3 (Two-Terminal Series-Parallel Graphs).** A directed acyclic graph is called two-terminal series-parallel TTSP if it can be defined recursively as follows:

1. The graph defined by two vertices connected by an edge (a 1-path) is a TTSP graph.

2. If $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ are TTSP where $S_i = \{s_i\}$, $T_i = \{t_i\}$ are the unique source and sink of $G_i$, then the following operations produce TTSP graphs:
   a) Parallel Addition: $G_p \leftarrow s_1 \sim s_2, t_1 \sim t_2$.
   b) Series Addition: $G_s \leftarrow t_1 \sim s_2$.

Denote the parallel join of $G_1$ and $G_2$ as $G_1 \otimes G_2$, and the corresponding series join as $G_1 \varpi G_2$.

The two recursive operations defining the TTSP graph model allow for a simple constructive approach for defining graphs from atomic elements. Indeed, efficient algorithms exist that decompose TTSP graphs into a decomposition tree with the following structure [100], [101], [117].

![Decomposition trees of two graphs.](image)
Definition 4 (TTSP Decomposition Tree). A TTSP decomposition tree of a TTSP graph $G$ is a binary tree $T(G)$ with the following properties:

1. $T$ is a complete\(^2\) binary tree, in that every node has either 2 or 0 children.

2. Every leaf of $T$ corresponds to a 1-path.

3. Every parent of $T$ corresponds to either a series or parallel addition operation from Definition 3 on its children.

Remark 6. In general, one can ignore the assumption of directed edges in Definition 3 and consider undirected TTSP graphs, see [101]. We consider undirected graphs (with symmetric Laplacians) in this chapter, as the notions of effective resistance require undirected graphs.

Examples of TTSP graphs and their decomposition trees are shown in Figure 9.2. The graph is constructed by a reverse breadth-first-search: the deepest leaves combine with each other by the series or parallel join designated by their parent. At each layer of the decomposition tree, each operation can be performed independently of the other nodes in the layer. This means that the complexity of the reconstruction operation is limited by the height of the decomposition tree; it is this key insight that allows the efficient computation of system-theoretic measures in the remainder of the chapter.

In the following proposition, we quantify the height of the tree in terms of the size of the resulting graph.

Proposition 2 (Properties of $T(G)$). Let $G$ be a TTSP graph with $N$ nodes constructed from $l$ 1-paths with $p$ parallel joins and $s$ series joins. Then,

1. $G$ has $N = 2l - 2p - s$ nodes and $E = l$ edges.

2. The decomposition tree $T(G)$ of $G$ has $n = 2l - 1$ nodes.

\(^2\) Some works refer to such a tree as full.
3. The height $h$ of $T(\mathcal{G})$ is bounded by

\[
\log_2 (N + 2p - s) \leq h \leq \frac{N + 2p + s}{2} - 2 \quad (9.3)
\]

\[
\log_2 (E) \leq h \leq E - 1. \quad (9.4)
\]

**Proof.** By definition, $T(\mathcal{G})$ is a complete binary tree, meaning that each parent has either 2 or 0 children. If it has zero children, it is a leaf and therefore corresponds to a 1-path added to $\mathcal{G}$. Each leaf thus adds one edge to $\mathcal{G}$, and since series and parallel joins do not delete edges, it follows that $E \equiv |E| = l$.

Each leaf also adds two nodes to $\mathcal{G}$, but each series join identifies a pair of nodes, and each parallel join identifies two pairs of nodes, thus removing one and two nodes from $\mathcal{G}$ respectively. Hence,

\[
N \equiv |N| = 2l - 2p - s. \quad (9.5)
\]

It is well-known that a complete binary tree has $n = 2l - 1$ nodes. Using (9.5), it follows that

\[
n = N + 2p + s - 1. \quad (9.6)
\]

Furthermore, the number of nodes $n$ of $T(\mathcal{G})$ is bounded by its height by the relation $2h + 1 \leq n \leq 2^{h+1} - 1$, leading to

\[
\log_2 (n + 1) - 1 \leq h \leq \frac{n - 1}{2}. \quad (9.7)
\]

Substituting (9.6) into the inequality (9.7) yields (9.3), and substituting $l = 2^{-1}(n + 1)$ into the inequality (9.7) yields the bounds in (9.4).

\[\square\]

### 9.2 System-theoretic analysis on series-parallel graphs

In this section, we examine how series-parallel graphs can be used to simplify ‘hard’ computational problems on networks by exploiting the structure of the decomposition tree. The basic ideas is that although some quantities are difficult to compute over the entire graph, they may be easy to compute over small “atomic” graphs (i.e., 1-paths or other atomic elements), and the quantities may propagate across simple graph compositions (i.e., series or parallel connections) by a recursive relation.\(^3\) Given a decomposition tree, these quantities may in fact be computable in parallel in $O(h)$ time. We give an intuitive example here that directly relates to the $H_2$ norm computation.

A natural use for series-parallel graphs is in electrical networks, as resistances add over a series connection of resistors, and conductances add over a parallel connection. Naïve methods for computing the effective resistances over general graphs can be expensive, such as the pseudoinverse-based formula in (9.2) which has complexity $O(|N|^{\omega})$\(^{120}\).

Recall that one can construct the tree by moving up the tree in a reverse breadth-first-search order and joining the graphs at each leaf via the opera-

\(^3\) Which is essentially a spatial dynamic programming.
tion prescribed by their parent. Similarly, given a TTSP network of resistors (edges with scalar weights corresponding to conductances) and its decomposition tree, one can start at the leaves (corresponding to 1-paths, i.e., individual resistors), and either add resistances together (if the join is series), or add their reciprocals (if the join is parallel). Hence, by performing this computation up the tree to the root, one yields the effective resistance across the TTSP graph from source to sink.

This is not the only useful electrical computation one can do on a TTSP graph. Suppose that a current is applied across the TTSP graph, from source to sink. Given measurements of currents across the resistors, it is possible to compute the total power dissipated across the graph without computing the effective resistance. Over a series connection of resistors $R_1, R_2$, the power dissipated is additive: $P_s = P_1 + P_2 = i^2(R_1 + R_2)$. Over a parallel connection of resistors, the total power dissipated is minimized over the resistors, subject to the conservation of current:

$$P_p(i) = \min_{\text{s.t. } i_1 + i_2 = i} \frac{i_1^2 R_1 + i_2^2 R_2}{P_1 \boxplus P_2};$$

the operation $P_1 \boxplus P_2$ denotes the infimal convolution; an example of the power computation over a TTSP graph is shown in Figure 9.4.

In the case of matrix-valued edge weights, there is a natural generalization of the infimal convolution of powers for both vector-valued and matrix-valued currents. The key insight is in realizing that power is essentially an inner product-induced norm – for a vector-valued current $i \in \mathbb{R}^k$ across
a matrix-valued resistor \( R, P = \langle i, R i \rangle \), which reduces to \( P = i^2 R \) when \( k = 1 \). Similarly, when we have a matrix-valued current \( I \in \mathbb{R}^{k \times k} \), as in \( \S 9.1.1 \), \( P(I) = \text{Tr}(I^T R I) \).

**Proposition 3.** Let \( R_1 \) and \( R_2 \) denote two-matrix valued resistances in \( S^k_{++} \), and consider current \( I \in \mathbb{R}^{k \times k} \) across \( R_1 \) and \( R_2 \) in parallel. Then, the effective resistance across the join is given by

\[
R_{\text{tot}} = R_1 : R_2.
\]

**Proof.** Recall that power with current \( i \) dissipates across a parallel join according to the infimal convolution

\[
P_p(I) = \min_{\text{s.t. } I_1 + I_2 = I} \text{Tr}(I_1^T R_1 I_1) + \text{Tr}(I_2^T R_2 I_2)
\]

\[
\approx (P_1 \Box P_2)(I).
\]

Applying the identity \((f \Box g)^* = f^* + g^*\), where \( f^* \) denotes the Fenchel conjugate, to Problem (9.8) yields the minimum \( P_p(I) = \text{Tr}(I^T (R_1 : R_2) I) \).

\( \Box \)

**9.2.1 Noise Rejection and Adaptive Weight Design**

Consider the leader-follower consensus dynamics setup from \( \S 9.1 \). In the scalar case (i.e., \( k = 1 \)), the \( \mathcal{H}_2 \) performance of the system is characterized by the norm of the transfer matrix, and when all nodes are observed (i.e., \( C = I \)), it is given by [75],

\[
(\mathcal{H}_2 G, B)^2 = \frac{1}{2} \text{Tr} \left( B^T A^{-1} B \right).
\]

In the case of matrix-valued edge weights, we have the following proposition.

**Proposition 4.** Consider the leader-follower consensus dynamics setup for arbitrary \( k \in \mathbb{Z}_{++} \). Suppose that all nodes are observed, i.e., \( C = C \otimes I_k = I \otimes I_k \). Then the \( \mathcal{H}_2 \) norm is given by

\[
(\mathcal{H}_2 G, B) = \frac{1}{2} \text{Tr} \left( B^T A^{-1} (W) B \right).
\]

**Proof.** The \( \mathcal{H}_2 \) norm is given by \( \text{Tr}(B^T P B) \), where \( P \) satisfies the Lyapunov equation

\[
-A(W)P - PA(W)^T + C^T C = 0.
\]

The ansatz \( P = \frac{1}{2} A^{-1} (W) \) yields the solution. \( \Box \)
Consider the task of re-assigning positive weights to the edges of the network to minimize the $\mathcal{H}_2$ norm. This can be done via a gradient-descent method on the optimization problem,

$$\min \left( \mathcal{H}_2^G \right)^2 = \frac{1}{2} \text{Tr} \left( B^T A (W)^{-1} B \right)$$

subject to

$$W := \text{Blkdiag}(W_e)$$
$$W_e \in S_{++}^{k \times k}, \forall e \in E$$
$$U_e \succeq W_e \succeq L_e, \forall e \in E, L_e, P_e \in S_{++}^{k \times k}.$$  

(9.9)

**Remark 7.** In the unconstrained case of Problem (9.9) (i.e., the edge weights $W_e$ are still positive-definite, but have no upper and lower bounds), the weights in the graph become can become unbounded in that an arbitrarily large weight $W_e$ (in the sense of the Loewner order on $W_e$) close to an input node can make the $\mathcal{H}_2$ norm arbitrarily small. Hence, it is necessary to include bounds on the matrix weights; this also has the benefit of not allowing the edges to become effectively disconnected. Furthermore, this also motivates the modification of the cost function to include a penalty term $\frac{1}{2} h \sum_{e \in E} \|W_e\|^2$ for some matrix norm $\| \cdot \|$ in order to reduce the large-weight blowup.

The optimization problem (9.9) has several key features, highlighted in the following proposition.

**Proposition 5.** Consider the setting of Problem (9.9). Then,

1. The objective function $\frac{1}{2} \text{Tr} \left( B^T A (W)^{-1} B \right)$ is strongly convex on the cone of positive-definite matrices $W_e \in S_{++}^k$.

2. The gradient of the objective function with respect to a single edge weight $W_e$ at a point $H \in S_{++}^k$ is given by $\nabla f_{W_e}[H] = -\frac{1}{2} Q^T Q$, where $Q = A_j A^{-1} e_s$, and so

$$\nabla f_{W_e}[H] = -\frac{1}{2} \sum_{s \in R} \left( Y_i - Y_j \right)^T \left( Y_i - Y_j \right),$$

(9.10)

where $Y_i = \text{Blk}_{i}^{k \times k} \left[ A^{-1} (e_s \otimes I_k) \right]$, and $A^{-1}$ is evaluated with $H$ in place of $W_e$.

The proof is left to the appendix.

We can solve Problem (9.9) using a projected gradient descent algorithm:

$$x_{k+1} = \text{Proj}_C \left( x_k + \frac{1}{h} \nabla f(x_k) \right),$$

where $C$ is the cone generated by the constraints of Problem (9.9).

Following Remark 7, let us include a Frobenius norm penalty on the edge weights in the cost:

$$f(W) = \frac{1}{2} \text{Tr} \left[ B^T A^{-1} B \right] + \frac{h}{2} \sum_{e \in E} \text{Tr} \left( W_e^T W_e \right)^2.$$
The gradient of the penalty term with respect to $W_e$ is simply $hW_e$, so each edge in the network updates its weight according to the dynamics given by the gradient update

$$W_{(i,j)}^{t+1} = \text{Proj}_C \left[ \left( 1 - \frac{1}{\sqrt{t}} \right) W_{(i,j)}^t + \frac{1}{\sqrt{t}} \nabla f_{(i,j)}[W_{(i,j)}^t] \right]. \quad (9.11)$$

A centralized algorithm and a decentralized conjugate gradient algorithm for solving Problem (9.9) in the scalar edge weight case were presented in [75]. For scalar edge weights, the natural efficient projection onto the constraint set is via the $\| \cdot \|_\infty$ norm, as this consists of at most two comparisons and a copy operation per edge weight in each step of the dynamics (9.11) at time $t$. In the case of matrix edge weights, there is no such natural efficient projection. Instead, at each step one solves the problem

$$\text{Proj}_C(X) = \min_{Y \in C \cap S^n_+} \| Y - X \|$$

for some matrix norm $\| \cdot \|$. However, we assume that $k$ is relatively small compared to the size of the overall network $\|E\|$, and so this operation is not the most computationally expensive part of the setup.\(^4\)

The main computational hurdle in solving Problem (9.9) is the computation of the gradient in (9.10), as it requires computing the voltage drops $Y_{s_i} Y_{s_j}$ for each edge $\{i, j\} \in E$. For the scalar case $k = 1$, the conjugate gradient algorithm is able to compute $y_{s_i} = [A(W)^{-1}e_{s_i}]_i$ in linear time (in $|N|$ steps) for each edge [75]; the case of matrix-valued resistances has not been analyzed in previous work, and we do so here.

We now present a decentralized algorithm for computing this quantity on a certain class of two-terminal series-parallel graphs, and a characterization of its complexity. In this setup, for every input node $s \in R$, the graph is a TTSP with $s$ as the sink, and a node representing the grounded leader set as the source.

**Definition 5 (All-Input TTSP Graphs).** Consider a graph $G$ in the setup of the leader-follower consensus dynamics. Identify each leader node $i_1, i_2, \ldots, i_r \in R$ as one node $l$ connected to all source nodes $s \in R$. The graph $G$ is called an all-input two-terminal series-parallel graph if, for all source nodes $s \in R$, $G$ is TTSP with $s$ as the source and $l$ as the sink.

See Figure 9.5 for an example of an all-input TTSP graph.

Informally, the efficient algorithm that we will present subsequently proceeds as follows. For each source node $s$, the algorithm utilizes the decomposition tree of $G$ with $s$ as the source and the grounded leader set as the sink; this is why $G$ needs to be TTSP with respect to all source nodes. The voltage drop can then be computed from resistances and currents across each join, and the currents can be extracted from the power dissipated across each edge.

\(^4\) For example, in formation control one may desire to use consensus on position and velocity, so $k \leq 6$ depending on whether the formation is planar or 3D.
join. Hence, the effective resistances are computed first, as in Algorithm 4:

**Algorithm 4:** Effective Resistance over TTSP Graph

**Input:** Decomposition tree $T(G)$
- Edge weights $W(G)$

**Result:** Effective resistances $\rho(G)$ over $T(G)$

for each leaf of $T(G)$ do
  - Output $R_{\text{eff}} = W_e^{-1}$ to parent;
end

for each parent $j$ of $T(G)$ do
  - if received $R_{\text{eff}}$ from both children $i = 1, 2$ then
    - if $j$ is a series join then
      - Output $R_{\text{eff}} = R_{\text{eff}_1} + R_{\text{eff}_2}$ to parent;
    - else
      - Output $R_{\text{eff}} = R_{\text{eff}_1} : R_{\text{eff}_2}$ to parent;
    - end
  - else
    - wait;
  - end
end

Once the effective resistance of each branch at each join is known, starting from the root of the decomposition tree, the currents at each join can be extracted from the power computations. This computation is summarized in Algorithm 5, and depicted in Figure 9.4:

**Algorithm 5:** Branch Currents over TTSP Graph

**Input:** Decomposition tree $T(G)$
- Effective resistances $\rho(G)$

**Result:** Currents $I(G)$ over $T(G)$

for each parent $j$ of $T(G)$ do
  - if $j$ is root then
    - if $j$ is a series join then
      - Output $I_{\text{out}} = I_k$ to children;
    - else
      - Output $(I_1, I_2) = \arg P_1 \square P_2$ to children;
    - end
  - else if received $I_{\text{in}}$ from parent then
    - if $j$ is a series join then
      - Output $I_{\text{out}} = I_{\text{in}}$ to children;
    - else
      - Output $(I_1, I_2) = \arg P_1 \square P_2$ to children;
    - end
  - else
    - wait;
  - end
end
Lastly, once the currents through each join are known, the voltage drops over each branch are computed starting from the leaves of the decomposition tree, as seen in Algorithm 6.

Algorithm 6: Voltage Drops over TTSP Graph

Input: Decomposition tree $T(G)$
Effective resistances $\rho(G)$ over $T(G)$
Currents $I(G)$ over $T(G)$

Result: Voltage drops $Y_s$ over $T(G)$

for each leaf of $T(G)$ do
    Output $V_{e_i} = W_{e_i}^{-1} I_e$ to parent;
end

for each parent $j$ of $T(G)$ do
    if received $V_{e_j}$ from both children $i = 1, 2$ then
        if $j$ is a series join then
            Output $V_{out} = V_{in_1} + V_{in_2}$ to parent;
        else
            Output $V_{out} = V_{in_1} = V_{in_2}$ to parent;
        end
    else
        wait;
    end
end

The computation of the voltage drops required for the gradient update scheme in \((9.11)\) simply requires calling Algorithms 1, 2 and 3 in succession for each $s \in \mathcal{R}$. We have the following result.

Theorem 27. Consider the computation of the voltage drops $Y_s$ in the gradient update scheme \((9.11)\). The sub-algorithms have the following complexities:

- Algorithm 4: Best-case complexity of $O(|\mathcal{R}| \log |\mathcal{N}|)$ (worst-case complexity of $O(|\mathcal{R}||\mathcal{N}|)$) calls of a $O(k^\omega)$ matrix inversion.
- Algorithm 5: Best-case complexity of $O(|\mathcal{R}| \log |\mathcal{N}|)$ (worst-case complexity of $O(|\mathcal{R}||\mathcal{N}|)$) calls of a $O(k^\omega)$ matrix inversion.
- Algorithm 6: Best-case complexity of $O(|\mathcal{R}| \log |\mathcal{N}|)$ (worst-case complexity of $O(|\mathcal{R}||\mathcal{N}|)$).
Proof. Algorithms 1, 2 and 3 are computations done on a binary tree of height $h$, with each layer’s computation done in parallel. Hence, the complexity is determined by $h$, which by Proposition 2 is $O(\log |N|)$, and $O(|N|)$ for best and worst-case, respectively. Each call of the three algorithms must be done $|\mathcal{R}|$ times. Algorithm 1 may perform an inversion of a $k \times k$ matrix, which has complexity $O(k^\omega)$. Algorithm 2 requires computing

$$\begin{align*}
(I_1, I_2) &= \text{arg } P_1 \cap P_2 \\
&= (R_{\text{eff}_1}^{-1} (R_{\text{eff}_1}^{-1} : R_{\text{eff}_2}^{-1}) \mathbf{I}_m, R_{\text{eff}_2}^{-1} (R_{\text{eff}_1}^{-1} : R_{\text{eff}_2}^{-1}) \mathbf{I}_n),
\end{align*}$$

over every parallel join, which also requires inverting $2k \times k$ matrices (the parallel addition of resistances is computed in Algorithm 1). The current best lower bound for $\omega$ is 2.3728639 $\times 10^7$. □

9.2.2 Synthesis of Optimal $\mathcal{H}_2$ Controllers

The algorithm described above can be used to compute not only the terms required for the gradient update scheme for the edge weights, but also for an a priori synthesis calculation of the $\mathcal{H}_2$ norm.

First, let us examine the structure of the $\mathcal{H}_2$ norm in the context of the leader-follower consensus setup.

Lemma 13. Consider the leader-follower consensus setup with $C = I \otimes I_k$. Then,

$$\begin{align*}
(\mathcal{H}_2^{G,B})^2 &= \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ \text{Blk}_{s}^{k \times k} \left[ A^{-1}_s e_s \right] \right] \\
&= \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ Y_s \right] = \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ \rho_{s,R}^{\text{eff}} \right].
\end{align*}$$

Proof. We note that,

$$\begin{align*}
(\mathcal{H}_2^{G,B})^2 &= \frac{1}{2} \text{Tr} \left( B^T A^{-1} B \right) \\
&= \frac{1}{2} \text{Tr} \left( \sum_{i,s \in \mathcal{R}} (e_i \otimes I_k) A^{-1}_s (e_s \otimes I_k)^T \right) \\
&= \frac{1}{2} \text{Tr} \left( \sum_{s \in \mathcal{R}} (e_s \otimes I_k) A^{-1}_s (e_s \otimes I_k)^T \right) \\
&= \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ \text{Blk}_{s}^{k \times k} \left[ A^{-1}_s e_s \right] \right] \\
&= \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ Y_s \right] = \frac{1}{2} \sum_{s \in \mathcal{R}} \text{Tr} \left[ \rho_{s,R}^{\text{eff}} \right]. \quad (9.12)
\end{align*}$$

Each quantity $Y_s$ on the left side of the sum in (9.12) is the voltage drop from the source node $s \in \mathcal{R}$ to the grounded leader node set $\mathcal{R}$. This is precisely the voltage dropped over the last parallel join of the series-parallel decomposition of an all-input TTSP graph; the join in question is exactly
the one depicted in Figure 9.5. We can utilize this observation to efficiently compute the \( H_2 \) norm \textit{a priori} knowing only the weights of the edges and the decomposition tree of \( G \).

We use the following setup. Consider a TTSP graph \( G \) with source node \( s \) and sink node \( t \). Ground the source node \( s \), and consider the grounded Laplacian \( A \) with respect to the grounded source \( s \). This is a leader-follower system with a single leader.

Note that the parallel join depicted in Figure 9.5 effectively makes one of the terminals of the resulting graph an element of \( R \), and the other terminal (the sink) an element of \( R \). The control matrix of the leader-follower consensus problem corresponds to exactly those elements in \( R \), which are the ‘sinks’ of the TTSP graph used in that computation. Therefore, our choice of \( B \) must always select the state of the sink vector \( t \); hence \( B = e_t = e_t \otimes I_k \).

We now proceed in two steps. First, we need a lemma that essentially states that for an arbitrary TTSP graph, there exists an equivalent 1-path TTSP graph with the same effective resistance. Then, any composition rule on arbitrary TTSP graphs can be reduced to a composition on the equivalent 1-paths, simplifying the analysis. Afterward, we show that the series-parallel composition of graphs produces a similar series-parallel computation of the \( H_2 \) performance.

**Lemma 14.** Consider two graphs: an arbitrary TTSP graph \( G_1 \) with source \( s_1 \) and sink \( t_1 \) with effective resistance \( \rho_{s_1,t_1}^{\text{eff}} \), and a 1-path TTSP graph \( G_2 \) with source \( s_2 \) and sink \( t_2 \) with effective resistance \( \rho_{s_2,t_2}^{\text{eff}} \). Let their respective control matrices be \( B_1 = e_{t_1} \) and \( B_2 = e_{t_2} \). Further suppose that \( \rho_{s_1,t_1}^{\text{eff}} = \rho_{s_2,t_2}^{\text{eff}} \). Then, \( (H_2^1)^2 = (H_2^2)^2 \).

**Proof.** The setup of the graphs in the statement of the lemma is a leader-follower consensus with grounded (leader) nodes \( s_1, s_2 \). Therefore, we can invoke Lemma 13. Using Lemma 13, denoting the graphs’ respective Dirichlet Laplacians as \( A_1, A_2 \) we can compute:

\[
(H_2^1)^2 = \frac{1}{2} \text{Tr} \left[ B_1^T \left[ A_1^{-1} \right] B_1 \right]
= \frac{1}{2} \text{Tr} \left[ (e_{t_1}^T \otimes I_k) \left[ A_1^{-1} \right] (e_{t_1} \otimes I_k) \right]
= \frac{1}{2} \rho_{s_1,t_1}^{\text{eff}} = \frac{1}{2} \rho_{s_2,t_2}^{\text{eff}}
= \frac{1}{2} \text{Tr} \left[ (e_{t_2}^T \otimes I_k) \left[ A_2^{-1} \right] (e_{t_2} \otimes I_k) \right]
= \frac{1}{2} \text{Tr} \left[ B_2^T \left[ A_2^{-1} \right] B_2 \right]
= (H_2^2)^2.
\]

\( \square \)

Lemma 14 will allow us to reduce the computation of the \( H_2 \) norm of a composite TTSP graph to the computation of \( H_2 \) norms of an equivalent 1-path. This allows us to prove the following theorem.
Theorem 28. Consider two graphs: an arbitrary TTSP graph \( G_1 \) with source \( s_1 \) and sink \( t_1 \), and a second arbitrary TTSP graph \( G_2 \) with source \( s_2 \) and sink \( t_2 \). Let the Dirichlet Laplacian of \( G_i \) grounded with respect to its source \( s_i \) be \( A_i \), and its control matrix be \( B_i = e_{t_i} \otimes I_k \). Hence its \( H_2 \) norm is given by \((H_2^{G_i})^2 = \frac{1}{2} \text{Tr}[B_i^T A_i^{-1} B_i] \). Then,

\[
(H_2^{G_1 \circ G_2})^2 = (H_2^{G_1})^2 + (H_2^{G_2})^2,
\]

\[
(H_2^{G_1 \otimes G_2})^2 \leq (H_2^{G_1})^2 : (H_2^{G_2})^2,
\]

with equality in the last display if and only if \( \rho_{s_1,t_1} = c \rho_{s_2,t_2} \) for \( c \in \mathbb{R}^+ \).

Proof. By Lemma 13, the \( H_2 \) norm of an arbitrary graph can be computed from an equivalent 1-path. Hence, we need to show (9.13) and (9.14) for series and parallel joins of 1-paths.

Consider the 1-paths in Figure 9.6, with weights \( W_1, W_2 \in S_{++}^k \) between the sources (square nodes) and sinks (circular nodes). The Dirichlet Laplacians of both with respect to the grounded source nodes are simply \( \mathcal{L}_{G_{G_1}} = [W_1], \mathcal{L}_{G_{G_2}} = [W_2] \), and their respective control matrices are \( B_{G_{G_1}} = B_{G_{G_2}} = I_k \) and \( H_2 \) norms are \((H_2^{G_1})^2 = \frac{1}{2} \text{Tr}[W_1^{-1}], (H_2^{G_2})^2 = \frac{1}{2} \text{Tr}[W_2^{-1}] \).

Similarly, the Laplacians of the series and parallel joins in Figure 9.6 are,

\[
\mathcal{L}_{G_{G_1 \circ G_2}} = [W_1 + W_2], \quad \mathcal{L}_{G_{G_1 \otimes G_2}} = \begin{bmatrix} W_1 + W_2 & -W_2 \\ -W_2 & W_2 \end{bmatrix}.
\]

The control matrices are \( B_{G_{G_1 \circ G_2}} = I_k, B_{G_{G_1 \otimes G_2}} = [0_{k \times k} \ I_k] \). Therefore, the \( H_2 \) norm in the series join case is

\[
(H_2^{G_1 \circ G_2})^2 = \frac{1}{2} \text{Tr}\left[ \begin{bmatrix} 0_{k \times k} & I_k \end{bmatrix} \begin{bmatrix} W_1 + W_2 & -W_2 \\ -W_2 & W_2 \end{bmatrix}^{-1} \begin{bmatrix} 0_{k \times k} \\ I_k \end{bmatrix} \right]
= \frac{1}{2} \left( W_2 - W_2 (W_1 + W_2)^{-1} W_2 \right)^{-1},
\]

(9.15)
where the formula in the last display follows from standard block matrix inversion formulae. Note that we can write
\[ W_1(W_1 + W_2)^{-1}W_2 = W_2(W_1 + W_2)^{-1}W_2 - W_2(W_1 + W_2)^{-1}W_2 + W_1(W_1 + W_2)^{-1}W_2 = (W_1 + W_2)(W_1 + W_2)^{-1}W_2 - W_2(W_1 + W_2)^{-1}W_2 = W_2 - W_2(W_1 + W_2)^{-1}W_2. \]

Inverting this expression and substituting into (9.15) yields,
\[ (\mathcal{H}_2 G_1 \circ G_2)^2 = \frac{1}{2} \left( W_1(W_1 + W_2)^{-1}W_2 \right)^{-1} = \frac{1}{2} W_1^{-1} + \frac{1}{2} W_2^{-1} = \left( \mathcal{H}_2 G_1 \right)^2 + \left( \mathcal{H}_2 G_2 \right)^2, \]
as desired. We can compute the \( \mathcal{H}_2 \) norm in the parallel join case by invoking Theorem 13 from [121], which states that for \( A, B \in S_{++}^k \),
\[ \text{Tr}(A : B) \leq \text{Tr}(A) : \text{Tr}(B), \]
with equality if and only if \( A = cB \) for some \( c \in \mathbb{R}_{++} \). By this result, and by Lemma 13, we can compute:
\[ (\mathcal{H}_2 G_1 \circ G_2)^2 = \frac{1}{2} \text{Tr} \left[ \rho_{1 \text{eff}} : \rho_{2 \text{eff}} \right] \leq \frac{1}{2} \left( \text{Tr} \left[ \rho_{1 \text{eff}} \right] : \text{Tr} \left[ \rho_{2 \text{eff}} \right] \right) = \left( \mathcal{H}_2 G_1 \right)^2 : \left( \mathcal{H}_2 G_2 \right)^2, \]
with equality if and only if \( \rho_{1 \text{eff}} = c\rho_{2 \text{eff}} \) for \( c \in \mathbb{R}_{++} \). 

We now propose the following Algorithm 7 for computing the \( \mathcal{H}_2 \) norm of a TTSP graph with control input \( e_s \). If the matrix weights across the graph
are scalar multiples of each other, then Algorithm 7 computes precisely the \( H_2 \) norm; otherwise it computes an upper bound.

**Algorithm 7**: \( H_2 \) norm of TTSP Graph with \( \mathbf{B} = \mathbf{e}_i \)

**Input**: Decomposition tree \( \mathcal{T}(\mathcal{G}) \), weights \( \mathbf{W}(\mathcal{G}) \), \( \mathbf{e}_i \)

**Result**: \( (H_2^{\mathcal{G}})^2 \)

**for each leaf \( \mathcal{L} \) of \( \mathcal{T}(\mathcal{G}) \) do**

- Output \( (H_2^{\mathcal{L}})^2 = \frac{1}{2}W^{-1}_L \) to parent;

**end**

**for each parent \( j \) of \( \mathcal{T}(\mathcal{G}) \) do**

- if received \( H_2^{\mathcal{G}_i} \) from both children then
  - if \( j \) is a series join then
    - Output \( (H_2)^2 \leftarrow \left( H_2^{\mathcal{G}_1} \right)^2 + \left( H_2^{\mathcal{G}_2} \right)^2 \) to parent;
  - else
    - Output \( (H_2)^2 \leftarrow \left( H_2^{\mathcal{G}_1} \right)^2 : \left( H_2^{\mathcal{G}_2} \right)^2 \) to parent;
  - end
- else
  - wait
- end

**end**

**return** \( (H_2)^2 \) at root node of \( \mathcal{T}(\mathcal{G}) \).

**Remark 8.** Note that in the setting of Theorem 28, if we have \( k = 1 \) then \( \text{Tr}(\mathcal{A} : \mathcal{B}) = \text{Tr}(\mathcal{A}) \cdot \text{Tr}(\mathcal{B}) \), which is why Algorithm 7 computes the exact \( H_2 \) norm.

We now proceed to the final result of this section.

**Theorem 29.** Consider a leader-follower consensus network on an all-input TTSP graph \( \mathcal{G} \). Then, the \( H_2 \) norm is given by

\[
(H_2^{\mathcal{G}, \mathcal{B}})^2 = -\frac{1}{2}\text{Tr}[\mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}] = \sum_{s \in \mathcal{R}} (H_2^{\mathcal{G}, \mathbf{e}_s})^2 ,
\]

and the best-case complexity of computing this \( H_2 \) norm is \( O(k|\mathcal{R}|\log|\mathcal{N}|) \), and the worst-case complexity is \( O(k|\mathcal{R}||\mathcal{N}|) \).

**Proof.** We can compute:

\[
(H_2^{\mathcal{G}, \mathcal{B}})^2 = -\frac{1}{2}\sum_{s \in \mathcal{R}} \text{Tr}[\mathbf{e}_s^T\mathbf{A}^{-1}\mathbf{e}_s] = \sum_{s \in \mathcal{R}} [H_2^{\mathcal{G}_s, \mathbf{e}_s}]^2 .
\]

This is a sum of \( |\mathcal{R}| H_2 \) norms of leader-follower consensus networks with control input \( \mathbf{e}_s \). Since at each layer of the decomposition tree \( \mathcal{T}(\mathcal{G}) \) each computation happens independently, the complexity depends on the height of the decomposition tree. The remainder of the proof is identical to that of Theorem 27.

**Remark 9.** The standard Lyapunov equation solution to the \( H_2 \) norm is of complexity \( O(k^3|\mathcal{N}|^3) \), or \( O(k^3|\mathcal{N}|^2) \) if the system matrix is symmetric. Hence, the series-parallel method provides significant computational savings.
In this section, we provide a computational example for the adaptive reweighting algorithm. We perform the gradient descent on the TTSP graph in Figures 9.7-9.10 with 3 leader nodes, identified to a single node depicted as an orange square. The weights are assumed to be elements of $\mathcal{S}^2_+$, which are $2 \times 2$ symmetric matrices, and thus have 3 independent elements. The 3 independent elements of each weight $W_{(i,j)}$ are represented as a multi-edge, with 3 edges between each node $i$ and its neighbour $j$. The edges connecting the leader nodes to the source nodes are identity weights, and the remaining weights were randomly initialized using the generator

$$W_{(i,j)} = \frac{G_{(i,j)} + G^T_{(i,j)}}{2} + 2I_2,$$

(9.16)

where $G_{(i,j)}$ is a $2 \times 2$ matrix with entries distributed according to a zero-mean standard Gaussian.

The weight penalty was set as $h = 0.05$, and upper/lower bounds for each weight were randomly generated according to Equation (9.16), with an additional scalar factor of 0.05 for the lower bound, and 10 for the upper bound.

The algorithm converges in a few steps, as shown in Figure 9.11.

9.4 PROOF OF PROPOSITION 5:

Recall that

$$\mathcal{H}_2^2 = \frac{1}{2} \sum_{s \in R} \text{Tr} \left[ e_s^T A^{-1} e_s \right]$$

(9.17)

$$A = A_c W_c A_c^T + \sum_{e \in \mathcal{E} \setminus c} A_e W_e A_e^T.$$

We want to find the gradient of the $\mathcal{H}_2$ norm with respect to $W_c$ for each $c \in \mathcal{E}$, and we do this by computing the gradients of each individual term $\text{Tr} \left[ e_s^T A^{-1} e_s \right]$ in (9.17), for each $s \in R$. 
Figure 9.8: Edges weights for example iteration 3

Figure 9.9: Edges weights for example iteration 5

Figure 9.10: Edges weights for example iteration 7
To this end, consider the functions

\[ f(X) = \text{Tr}(e_s^T X e_s) \]
\[ g(X) = X^{-1} \]
\[ h(X) = A_c X A_c^T + \sum_{e \in \mathcal{E} \setminus c} A_e W_e A_e^T. \]

These functions have differentials,

\[ d_f(X) [H] = \text{Tr}(e_s^T H e_s) \]
\[ d_g(X) [H] = -X^{-1} H X^{-1} \]
\[ d_h(X) [H] = A_c H A_c^T. \]

A single term of interest is given by

\[ \text{Tr} \left( e_s^T A^{-1} e_s \right) = f(g(h(W_c))) = (f \circ g \circ h)(W_c), \]

and the differentials are thus similarly composed. Note that \( g \) is strongly convex on the cone of PD matrices, and both \( f \) and \( h \) are linear functions in \( X \). It follows that the objective function is strongly convex, proving the first part of the proposition.

Next, we have

\[ d(g \circ h)(X) [H] = -X^{-1} \left( A_c H A_c^T \right) X^{-1} \]
\[ X = \left( A_c H A_c^T + \sum_{e \in \mathcal{E} \setminus c} A_e W_e A_e^T \right) = A, \]

and we also have that

\[ d(f \circ g \circ h)(X) [H] = -\text{Tr} \left( e_s^T A^{-1} A_c H A_c^T A^{-1} e_s \right). \]

Let \( Q = A_c^T A^{-1} e_s \). Then, we can identify the gradient at a point \( H \) as,

\[ d(f \circ g \circ h)(X) [H] = -\text{Tr} \left( Q^T H Q \right) \]
\[ = \langle -Q^T Q, H \rangle, \]
and so the gradient at $H$ is identified as $-Q^T Q = -(A_i^T A^{-1} e_s)^T (A_i^T A^{-1} e_s)$.

Let us examine the form of $Q$ by letting $g = \{i, j\}$; in particular,

$$Q = A_i^T A^{-1} e_s = \left(a_g^T \otimes I_k\right) A^{-1} (e_s \otimes I_k)$$

$$= \left(e_i^T \otimes I_k - e_j^T \otimes I_k\right) A^{-1} (e_s \otimes I_k).$$

This is precisely $\text{Blk}_{k \times k} [A^{-1} e_s] - \text{Blk}_{k \times k} [A^{-1} e_s] := Y_i - Y_j$, or the difference in matrix-valued voltage drops to the grounded leader node from node $i$ to node $j$. Hence, the gradient with respect to weight $W_c$ is given by

$$\nabla_{W_c} \mathcal{H}_2[H] = -\frac{1}{2} \sum_{s \in R} (Y_i - Y_j)^T \left(Y_i - Y_j\right),$$

where $A^{-1}$ is computed with $H$ in place of $W_c$.

**Remark 10.** *It is precisely this matrix inverse that we are able to avoid computing using the methods for series-parallel networks in §9.2.*
TIMESCALED CONSENSUS

10.1 LEADER-FOLLOWER CONSENSUS MODELS WITH TIMESCALES

In this section, we will describe a general formulation for consensus over a communication network with non-negative edge weighting, positive node time scaling, while accounting for possible measurement and process noise. The scaled consensus problem is derived from considering a group of $n$ multi-rate integrators \[80\], with zero-mean Gaussian process noise, $\omega_i(t)$ such that $\mathbb{E}[\omega_i(t)\omega_i(t)^T] = \Gamma_i$, $\forall i \in \mathcal{N}$,

$$
\begin{bmatrix}
\epsilon_i,1 \hat{x}_i,1 \\
\vdots \\
\epsilon_i,n \hat{x}_i,n
\end{bmatrix} = E_i \hat{x}_i(t) = u_i(t) + \omega(t),
$$

(10.1)

where $\omega(t)$ is the stacked vector of process noises $\omega_i$. Accounting for the communication between agents can be accomplished by representing the system as a graph with node set given by the agents, and edge set defined by the interconnections. Suppose that communication between agents $i$ and $j$ is corrupted by zero-mean Gaussian noise $v_{ij}$, and let $v_i$ denote the sum of all noise inputted into agent $i$ from the connections to its neighbours in $N(i)$. Without loss of generality, we can assume that the covariance of $v_i$ is given by $\mathbb{E}[v_i(t)v_i(t)^T] = \Omega_i$. A weighted, decentralized feedback controller (corrupted by the above noise model) that seeks to bring agents into consensus is given by,

$$
u_i(t) = v_i(t) + \sum_{j \in N(i)} (W_{ij}(x_j(t) - x_i(t)))
$$

(10.2)

where $W$ is the block-diagonal matrix of edge weights with properties detailed in §13.1. The vector $u$ is the stacked vector of control vectors $u_i$, and $v$ is the stacked vector of all noises. The matrix version of applying (10.2) to (10.1) gives the general, time scaled and weighted consensus problem with process and measurement noise,

$$
\dot{x}(t) = -E^{-1} \mathcal{L}_{\mathcal{G}_w}(\mathcal{G}) x(t) + \begin{bmatrix} E^{-1} & -E^{-1} \mathcal{D}(\mathcal{G}) \end{bmatrix} \begin{bmatrix} \omega(t) \\ v(t) \end{bmatrix}
$$

(10.3)

where $\mathcal{L}_{\mathcal{G}_w} = \mathcal{D}(\mathcal{G}) \mathcal{W}(\mathcal{G})^T$ is the weighted Laplacian matrix, and $E = \text{Blkdiag}(E_1, \ldots, E_n)$ is the time scale matrix of $\mathcal{G}$. The output, $z(t)$ in (10.3), will be used to monitor the performance of the network and assess how the available system output impacts the network performance.
As noted by [122] (in the scalar-valued node state case), the zero eigenvalue of the Laplacian matrix (assuming \( G \) is connected) precludes reasoning about the \( H_2 \) performance of (10.3). This property of the Laplacian matrix persists in the scaled, weighted case [86] (as well as the matrix-valued edge weight case), so as in [122] we will appeal to a similarity transform that separates out the zero eigenvalues, which is described in the following theorem.

**Theorem 30.** The scaled and edge-weighted graph Laplacian for a connected graph with time scale matrix \( E \) and weight matrix \( W \), given by

\[
\mathcal{L}_{G,w} = E^{-1}D(G)WD(G)^T,
\]

is similar to

\[
\begin{bmatrix}
\tilde{\mathcal{L}}_{G,e,R} & \mathbf{R} \\
\mathbf{0} & \mathbf{0}
\end{bmatrix},
\]

where \( \tilde{\mathcal{L}}_{G,e,s} = D(G)\mathbf{R}^{-1}D(G) \) is the edge Laplacian of \( G \) with time scales, and

\[
\mathbf{R}(G) = \begin{bmatrix} I & T_c^e \end{bmatrix} = R \otimes I_k
\]

with,

\[
T_c^e = (D(G)\mathbf{R})^{-1}D(G) \mathbf{R}
\]

and \( \mathbf{R} \) is the basis of the cut space of \( G \) as defined as in [122]. Here, the \( \tau \) and \( c \) subscripts on \( G \) denote the incidence matrices for a spanning tree and the complementary edges in \( G \), respectively.

**Proof.** First, we prove a lemma.

**Lemma 15 (Forms of \( T_c^e \) and \( D \)).** The following hold:

\[
T_c^e = T_c^\tau \otimes I_k
\]

\[
D = D_c^\tau R.
\]

**Proof.** We can compute:

\[
T_c^e = (D_c^T D_c)^{-1} D_c^T D_c^\tau
\]

\[
= \left( (D_c \otimes I_k)^T (D_c \otimes I_k) \right)^{-1} (D_c \otimes I_k)^T (D_c \otimes I_k)
\]

\[
= \left( (D_c^T D_c) \otimes I_k \right)^{-1} (D_c^T D_c) \otimes I_k
\]

\[
= \left( (D_c^T D_c)^{-1} D_c \right) \otimes I_k
\]

\[
= T_c^\tau \otimes I_k,
\]
and similarly,

\[ D = D \otimes I_k \]
\[ = \begin{bmatrix} D_r & D_c \end{bmatrix} \otimes I_k \]
\[ = D_r \begin{bmatrix} I & T^c_r \end{bmatrix} \otimes I_k \]
\[ = (D_rR) \otimes I_k \]
\[ = (D_r \otimes I_k) (R \otimes I_k) \]
\[ = D_rR. \]

Following [86], [122], we define the similarity transforms

\[ S_v(G) = \begin{bmatrix} E^{-1}D(G_r) (D(G_r)^T E^{-1}D(G_r))^{-1} 1_{n \otimes k} \end{bmatrix} \]
\[ S_v(G)^{-1} = \begin{bmatrix} D(G_r)^T \Xi^{-1}F \end{bmatrix}, \]

where

\[ 1_{n \otimes k} = 1_n \otimes I_k \]
\[ F = \begin{bmatrix} E_1 & \cdots & E_n \end{bmatrix} \]
\[ \Xi = \text{diag} \left( \{\epsilon_{s,i}\}_{i=1}^k \right) \]
\[ \epsilon_{s,i} = \sum_{j=1}^n \epsilon_{j,i}. \]

Note that \( \epsilon_{s,i} \) is the sum of all the time scale parameters of the \( i \)th state over all nodes.

**Lemma 16.** The similarity transforms are well-defined, in that \( S_v^{-1}S_v = I \).

**Proof.** Denote the product \( S_v^{-1}S_v \) in block form:

\[ S_v^{-1}S_v = \begin{bmatrix} A & B \\ C & D \end{bmatrix}. \]

First, note that

\[ FE^{-1} = \begin{bmatrix} E_1 & \cdots & E_n \end{bmatrix} E^{-1} = \begin{bmatrix} I & \cdots & I \end{bmatrix} = 1^T_n \otimes I_k. \]
Then, we can compute each term:

\[ \mathcal{A} = D_r^T E^{-1} D_r \left( D_r^T E^{-1} D_r \right)^{-1} = I \]

\[ \mathcal{B} = D_r^T 1_{n \otimes k} \]

\[ = (D_r \otimes I_k)^T (1_k \otimes I_k) \]

\[ = (D_r^T 1_n) \otimes I_k = 0 \otimes I_k = 0. \]

\[ \mathcal{C} = \Xi^{-1} F E^{-1} D_r \left( D_r^T E^{-1} D_r \right)^{-1} \]

\[ = \Xi^{-1} \left( 1_n^T \otimes I_k \right) (D_r \otimes I_k) \left( D_r^T E^{-1} D_r \right)^{-1} \]

\[ = \Xi^{-1} \left( 1_n^T D_r \otimes I_k \right) \left( D_r^T E^{-1} D_r \right)^{-1} \]

\[ = 0. \]

Lastly, we have

\[ \mathcal{D} = \Xi^{-1} F 1_{n \otimes k} \]

\[ = \begin{bmatrix}
\varepsilon_{s,1}^{-1} & \cdots \\
\varepsilon_{s,k}^{-1} & \varepsilon_{s,1}^{-1} & \cdots \\
& \ddots & \ddots & \ddots \\
& & \varepsilon_{s,k}^{-1} & \cdots \\
& & & \varepsilon_{s,1}^{-1} & \cdots \\
& & & & \varepsilon_{s,k}^{-1}
\end{bmatrix}
\begin{bmatrix}
E_1 & \cdots & E_n \\
\sum_{j=1}^n \varepsilon_{j,1} & \cdots & \sum_{j=1}^n \varepsilon_{j,k}
\end{bmatrix}(1_n \otimes I_k) \]

\[ = \begin{bmatrix}
\varepsilon_{s,1}^{-1} \\
\vdots \\
\varepsilon_{s,k}^{-1}
\end{bmatrix}
\begin{bmatrix}
\sum_{j=1}^n \varepsilon_{j,1} & \cdots & \sum_{j=1}^n \varepsilon_{j,k}
\end{bmatrix}
\]

\[ = I_k \]

\[ \square \]

Next, denoting for brevity \( D_r := D(\mathcal{G}_r), D := D(\mathcal{G}) \), etc, and noting that \( D_r^T 1_{n \otimes k} = (D^T 1) \otimes I_k = 0 \), we have

\[ S_v^{-1} L_{G,r} S_v = \]

\[ \begin{bmatrix}
D_r^T E^{-1} D_r & \Xi^{-1} F D D_r^T D_r (D_r^T \Xi^{-1} D_r)^{-1} & 0 \\
\Xi^{-1} F D D_r^T D_r (D_r^T \Xi^{-1} D_r)^{-1} & \Xi^{-1} F E D_r^T D_r (D_r^T \Xi^{-1} D_r)^{-1} & 0 \\
D_r^T E^{-1} D_r & \Xi^{-1} 1_{n \otimes k} & 0 \\
\Xi^{-1} 1_{n \otimes k} & \Xi^{-1} 1_{n \otimes k} & 0
\end{bmatrix} \]

\[ = \begin{bmatrix}
\tilde{L}_{G,r} & 0 \\
0 & 0
\end{bmatrix}, \]

as desired. \( \square \)
The model in (10.3) can now be transformed to edge variables with the scaled coordinate transformation introduced in Theorem 30, and noting that \( S_t x_e(t) = x(t) \). The scaled, weighted consensus model with noise is then equivalent to,

\[
\begin{align*}
\dot{x}_e(t) &= \begin{bmatrix}
-L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G}) \mathcal{W} \mathcal{R}(\mathcal{G})^T & 0 \\
0 & 0
\end{bmatrix} x_e(t) \\
+ & \begin{bmatrix}
\mathcal{D}^1_t E^{-1} & -L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G})
\end{bmatrix} \begin{bmatrix}
\omega(t) \\
v(t)
\end{bmatrix} \\
z(t) &= \begin{bmatrix}
\mathcal{R}(\mathcal{G}) & 0
\end{bmatrix} x_e(t).
\end{align*}
\] (10.4)

where \( L^\tau \phi_{e,s}(\mathcal{G}_t) = D^1_t E^{-1} D^\tau \) is the edge Laplacian for spanning tree \( \mathcal{G}_t \) which is symmetrically “weighted” by the time scaling parameters. We can note that the form of (10.4) naturally suggests a partitioning of the edge state variable into a set of states in the spanning tree and those in the consensus space \( \text{span}(1 \otimes I_k) \), \( x_e(t) = \begin{bmatrix} x_e(t) & x_f(t) \end{bmatrix} \). The resulting dynamics for the spanning tree states is taken from (10.4) as,

\[
\Sigma^\tau_e := \begin{cases}
\dot{x}_e(t) &= -L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G}) \mathcal{W} \mathcal{R}(\mathcal{G})^T x_e(t) \\
&+ \mathcal{D}^1_t E^{-1} \Omega \dot{\omega} - L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G}) \Gamma \dot{v} \\
z(t) &= \mathcal{R}(\mathcal{G})^T x_e(t),
\end{cases}
\] (10.5)

where \( \dot{v} \) and \( \dot{\omega} \) are normalized noise signals, \( \Omega = \mathbb{E} [\omega(t) \omega(t)^T] \), and \( \Gamma = \mathbb{E} [v(t) v(t)^T] \). An important note is that the output of (10.5) contains information of the cycle states due to the inclusion of \( \mathcal{R}(\mathcal{G}) \) and the fact that the cycle states are linear combinations of the tree states. We can also consider the same edge state model with output given solely by the spanning tree states,

\[
\Sigma^\tau_e := \begin{cases}
\dot{x}_e(t) &= -L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G}) \mathcal{W} \mathcal{R}(\mathcal{G})^T x_e(t) \\
&+ \mathcal{D}^1_t E^{-1} \Omega \dot{\omega} - L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R}(\mathcal{G}) \Gamma \dot{v} \\
z(t) &= x_e(t).
\end{cases}
\] (10.6)

The \( \mathcal{H}_2 \) performance of (10.5) and (10.6) are given by \( \text{tr}(R^T X^* R) \) and \( \text{tr}(X^*) \), respectively, where \( X^* \) is the positive-definite solution to the Lyapunov equation,

\[
-L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R} \mathcal{W} \mathcal{R}^T X - X \mathcal{R} \mathcal{W} \mathcal{R}^T L^\tau \phi_{e,s}(\mathcal{G}_t) + \mathcal{D}^1_t E^{-1} \Omega \mathcal{E}^{-1} E^{-1} D^\tau + L^\tau \phi_{e,s}(\mathcal{G}_t) \mathcal{R} \Gamma \mathcal{E}^{-1} E^{-1} D^\tau = 0.
\] (10.7)

In general, the addition of the weighting and scaling precludes a closed form solution to (10.7) (which is desirable to find \( X^* \)’s dependence on \( E, W \)), and numeric results yield a nonlinear mixing of weights and scaling param-
eters in the entries of $X$. However, in the following section we will outline a case when analytic solutions to (10.7) exist, their interpretations, and the insights they can give for design of edge weights and scaling parameters for optimal performance.

10.2 PERFORMANCE AND DESIGN PROBLEMS

10.2.1 $\mathcal{H}_2$ Performance

In this section, we discuss the $\mathcal{H}_2$ performance for the models of edge consensus in the cases of nodes with timescales, and weighted edges.

10.2.1.1 Analytic Solutions to the Lyapunov Equation

By appropriately selecting the covariance matrices $\Omega$ and $\Gamma$, we can find analytic solutions for (10.7) in line with those found in [122]. Specifically, if we take $\Omega = \sigma_w \mathbf{E}^{1/2}$ and $\Gamma = \sigma_v \mathbf{W}^{1/2}$, then by inspection (10.7) has the following solution,

$$X^* = \frac{1}{2} \left( \sigma_w^2 (\mathbf{RWR}^T)^{-1} + \sigma_v^2 \mathcal{L} \mathcal{G}_{\tau, s}^\tau \right). \quad (10.8)$$

This selection of $\Omega$ and $\Gamma$ may at first seem based in convenience, but there is actually strong justification for this choice. Consider a scaled node that is subject to process noise with a variance being controlled by an external influencer. As can be seen from the dynamics in (10.5) or (10.6), a input noise signal with unity variance will be scaled by the node’s time scaling factor; intuitively we can see that slow (higher scaling parameter) will attenuate the input, while faster nodes act as an amplifier for the input. So in the case of a scaled network, to achieve a consistent value of $\sigma_\omega$ variance into each node, the input signal must be scaled by the scaling parameters over the network.

The same reasoning applies to the measurement noise over the edges, which can be attenuated or amplified by the presence of non-unity weights. Thus, we can interpret the $\sigma_w$ and $\sigma_v$ parameters as the desired, effective variance to be applied to each node/edge, and the scaling/weighting matrices are then applied to ensure that all nodes/edges receive an appropriately adjusted input signal. Now that we have justified our choice of covariance matrices, let us consider the implications of this result.

Equation (10.8) is of particular interest because it shows by associating the covariances with the magnitudes of the edge weights and the time scale parameters, the edge and node weightings are completely separated in their effect on the $\mathcal{H}_2$ performance, save for the placement of the $\sigma_\omega$ and $\sigma_v$ parameters (that is, the effective covariance parameter of the process noise is a “node” parameter, but multiplies the term containing the edge weighting.
performance and design problems

10.2 PERFORMANCE AND DESIGN PROBLEMS

and vice versa). To aid in the consideration of these independent contributions later, we can then define,

\[
\mathcal{H}_2(\Sigma) = \frac{\sigma_w^2}{2} \text{tr}(R^T (RWR^T)^{-1} R) + \frac{\sigma_v^2}{2} \text{tr}(R^T \mathcal{L}_{\theta e,s}^r R)
\]

\[
:= \mathcal{H}_2(\Sigma, W) + \mathcal{H}_2(\Sigma, E),
\]

and similarly,

\[
\mathcal{H}_2(\hat{\Sigma}) = \frac{\sigma_w^2}{2} \text{tr}((RWR^T)^{-1}) + \frac{\sigma_v^2}{2} \text{tr}(\mathcal{L}_{\theta e,s}^r)
\]

\[
:= \mathcal{H}_2(\hat{\Sigma}, W) + \mathcal{H}_2(\hat{\Sigma}, E).
\]

We will see in later sections that the minimization of \(\mathcal{H}_2(\cdot, W)\) in comparison to \(\mathcal{H}_2(\cdot, E)\) gives rise to different strategies. To enable this, we can further refine (10.9), considering the weighting term and scaling term separately.

First, we start with the edge weight term,

\[
\mathcal{H}_2(\Sigma, W) = \frac{\sigma_w^2}{2} \text{tr}(R^T (RWR^T)^{-1} R)
\]

\[
= \frac{\sigma_w^2}{2} \text{tr} \left( \begin{bmatrix} I & T^e_r \end{bmatrix} (RWR^T)^{-1} \begin{bmatrix} I & T^e_r \end{bmatrix} \right)
\]

\[
= \frac{\sigma_w^2}{2} \text{tr} \left( \begin{bmatrix} (RWR^T)^{-1} & (RWR^T)^{-1} T^e_r \\ T^e_r (RWR^T)^{-1} & (T^e_r)^T (RWR^T)^{-1} T^e_r \end{bmatrix} \right)
\]

\[
= \frac{\sigma_w^2}{2} \left( \text{tr} ((RWR^T)^{-1}) + \text{tr} ((T^e_r)^T (RWR^T)^{-1} T^e_r) \right)
\]

\[
= \mathcal{H}_2(\Sigma, W) + \frac{\sigma_w^2}{2} \text{tr} \left( (T^e_r)^T (RWR^T)^{-1} T^e_r \right).
\]

(10.10)

A similar relation can be found for the time scale term,

\[
\mathcal{H}_2(\Sigma, E) = \frac{\sigma_v^2}{2} \text{tr}(R^T \mathcal{L}_{\theta e,s}^r R)
\]

\[
= \frac{\sigma_v^2}{2} \text{tr} \left( \begin{bmatrix} \mathcal{L}_{\theta e,s}^r & \mathcal{L}_{\theta e,s}^r T^c_r \\ (T^e_r)^T \mathcal{L}_{\theta e,s}^r & (T^e_r)^T \mathcal{L}_{\theta e,s}^r T^c_r \end{bmatrix} \right)
\]

\[
= \mathcal{H}_2(\Sigma, E) + \frac{\sigma_v^2}{2} \text{tr} \left( (T^e_r)^T \mathcal{L}_{\theta e,s}^r T^c_r \right).
\]

(10.11)

The simplifications to (10.9) show that the \(\mathcal{H}_2\) performance of the \(\Sigma\) system, which has output containing information from the cycle edge states, predictably contains the \(\mathcal{H}_2\) performance for the \(\hat{\Sigma}\) system as an isolated term. Taken together, (10.10) and (10.11) illustrate how the output information differences between (10.4) and (10.5) influence the overall \(\mathcal{H}_2\) performance for identical tree-edge-state dynamics.

As seen in [122], the cycle structure of arbitrary graphs makes closed-form solutions for terms containing \((RR^T)^{-1}\) difficult. In the case considered here, the consideration of edge weights allows us to consider terms containing \((RWR^T)^{-1}\) instead. Unfortunately, this does not offer a reprieve from the
issues discussed for the unweighted case. For unity edge weighting, we can appeal to the results presented there for $\mathcal{H}_2(\Sigma, W)$ and $\mathcal{H}_2(\Sigma, W)$ in the case of cycles and complete graphs. We reproduce those below for completeness and to bring them in line with our notation,

$$\mathcal{H}_{2,C_n}(\Sigma, W = I) = \frac{\sigma_2^2 (n-1)^2}{n}$$

$$\mathcal{H}_{2,K_n}(\Sigma, W = I) = \frac{\sigma_2^2 (n-1)}{n}.$$ 

In the case of tree graphs, however, the analysis simplifies even further as we will discuss in the next section.

10.2.1.2 Tree Graphs

When the underlying graph topology is a tree, $R = I$, and (10.8) simplifies to,

$$X^* = \frac{1}{2} \left( \sigma_2^2 (W)^{-1} + \sigma_2^2 L_{\epsilon, e,s}^T \right).$$

Furthermore, in this case $\mathcal{H}_2(\Sigma_e) = \mathcal{H}_2(\Sigma_e) = \text{tr}(X^*)$. A closed form solution for the performance in this case is given in the following lemma.

**Lemma 17.** For a tree graph, the $\mathcal{H}_2$ norm of the $\Sigma_e$ (10.6) system is given by,

$$\mathcal{H}_2(\Sigma_e) = \frac{1}{2} \text{tr} \left( \sigma_2^2 (W)^{-1} + \sigma_2^2 L_{\epsilon, e,s}^T \right) = \frac{1}{2} \sum_{l=1}^{k} \left( \sigma_2^2 \sum_{k=1}^{n-1} \frac{1}{w_{j,l}} + \sigma_2^2 \sum_{i=1}^{n} \frac{\text{deg}(v_i)}{\epsilon_{i,l}} \right),$$

(10.12)

where $\text{deg}(v_i)$ is the unweighted degree of agent $v_i$, and $j$ is the index over the edges.

**Proof.** Consider a single layer of substates, denoted by $l$. We have that $W_l$ is diagonal matrix of weights, so the trace of the inverse is simply the sum of the inverted weights. For the second term, consider one of the diagonal elements of $L_{\epsilon, e,s}^T$.

$$[L_{\epsilon, e,s}^T(ql)] = a_q^T \epsilon_{l}^{-1} a_q = \epsilon_{i,l}^{-1} + \epsilon_{j,l}^{-1},$$

where $a_q$ is the edge vector corresponding to the edge between nodes $i$ and $j$, that is, $q = \kappa(ij)$. Now consider a single node, $v_i$. In the sum over all edges of the graph, $\epsilon_{i,l}^{-1}$ will appear once for every edge that connects $v_i$ to its neighbors, which is the unweighted degree of $v_i$. Considering all other nodes gives the result for the second term.

Finally, the preceding argument holds for all the sub-state layers, which gives the sum over all sub-states. $\square$

From this result, we can see that there exists a trade-off between the time scale parameters and the topology (in this case, the degree distribution).
which determines the overall performance of the network. Also, we can con-
trast the influence of time scale parameters and edge weights in this case. For
a given distribution of scaling parameters and edge weights, changing the
assignment of edge weights does not affect the $\mathcal{H}_2$ performance. However,
the assignment of scaling parameters can have a significant effect on the
performance of the network, which is in line with the similar results in the
context of single-input influenced consensus [86]. We see this explicitly in
the following example.

10.2.1.3 Effect of Time Scale Distribution Example

Consider a tree graph on six nodes with topology given in Figure 10.1. As-
sume that we have some distribution of edge weights and node scaling pa-
rameters such that $\sum w_i^{-1} = 2 \alpha$, $\sum \epsilon_i = 1$, and further, that $\epsilon_i \in (0.1, 0.2, 0.4)$. Also, let $\sigma_v = \sigma_\omega = 1$.

![Tree graph for six agents, $T$. Each agent has an associated scaling pa-
parameter, $\epsilon_i$, with subscript denoting the agent number, and each edge
is similarly labeled $w_j$. Agent 3 has the highest degree.]

Now, for any assignment of weights, the first term in (10.12) will be $2 \alpha$. However, the second term depends on the assignment of the scaling parameters. Consider the two following assignments:

- $\epsilon_{1,4,5,6} = 0.1, \epsilon_2 = 0.2, \epsilon_3 = 0.4$
  In this case, the second term is equal to $\sum_{i=1}^6 \text{deg}(i)/\epsilon_i = 60.0$ resulting in a combined $\mathcal{H}_2 = 30 + \alpha$. Now, consider a different distribution of scaling parameters.

- $\epsilon_{1,3,4,5} = 0.1, \epsilon_2 = 0.2, \epsilon_6 = 0.4$
  We can see that with agent 3 on a faster time scale, the second term suffers, $\sum_{i=1}^6 \text{deg}(i)/\epsilon_i = 82.5$, resulting in a comparatively higher performance value of $\mathcal{H}_2 = 41.25 + \alpha$.

Thus, we can see that high-degree nodes operating on slower time scales
results in lower $\mathcal{H}_2$ performance, corresponding to better network resilience.

10.2.2 Timescale Design For $\mathcal{H}_2$ Resilience

The results of the previous section suggest a heuristic for minimizing the
$\mathcal{H}_2$ performance is to identify high degree agents and ensure that they are
operating on a slower time scale than the lower degree agents. However, it
is not clear from the analysis for tree graphs if this holds in the presence of
cycles. To investigate this, we can note that the minimization of $\mathcal{H}_2(\Sigma)$ can be formulated as a convex problem,

$$\min_{\epsilon_1^{-1}, \ldots, \epsilon_n^{-1}} \text{tr} \left( RX R^T \right)$$

subject to

$$\frac{1}{\epsilon_{\max}} \leq \frac{1}{\epsilon_{i,l}} \leq \frac{1}{\epsilon_{\min}} \quad \forall \, i \in \mathcal{N}, \, l \in \{1, \ldots, k\}$$

$$\mu_l \leq \sum_{i=1}^{n} \frac{1}{\epsilon_{i,l}} \quad \forall \, l \in \{1, \ldots, k\}$$

$$X = \frac{1}{2} \left( (RWR^T)^{-1} + \mathcal{L}_{\theta_{\epsilon,s}}^{\tau} \right),$$

where we have taken the effective variances, $\sigma_\omega$ and $\sigma_v$, to be unity. The objective is convex in the optimization variables $(1/\epsilon_{1,1}, \ldots, 1/\epsilon_{n,k})$, and the constraints are linear. The design parameter of $\mu_l$ serves to ensure that not all the agents can operate on the slowest time scale, that is, it can be tuned to ensure that each layer of sub-states in the network has a requisite responsiveness that must be balanced with the desired resilience.

To investigate the effects of cycles on the distribution of time scales that minimizes the $\mathcal{H}_2(\Sigma)$, we generated a random graph for 10 agents (with probability of an edge between any two nodes as 0.15) that features multiple independent cycles. For $\epsilon_{\min} = 0.01$, $\epsilon_{\max} = 2.0$ and $\mu = 510.5$ (which can be interpreted as the value allowing up to half the nodes to be slow), the results and graph topology are presented Figure 10.2. These results (which appear to hold over a wide range of randomly generated graphs) suggest that the presence of cycles do not detract from the heuristic developed for tree graphs - that high degree nodes should be assigned slow time scales (high scaling parameters) to minimize $\mathcal{H}_2(\Sigma)$.

![Figure 10.2: Random graph topology for 10 agents. The node number and time scale assigned by (P1) are printed in each node, which shows the slowest time scales are assigned to nodes with highest degree.](image)

What is of further interest is that despite the fact that the time scales present in (P1) manifest through the edge Laplacian of the spanning tree (which may not reflect the degree distribution of the full graph that the heuristic is driven by), the optimal distribution appears to be independent of
the spanning tree chosen. We can see this by generating spanning trees with a variety of degree distributions, such as those in Figures 10.3 and 10.4. Both of these spanning trees generate the distribution presented in Figure. This suggests that the multiplication by $R(\mathcal{G})$ and its transpose in the objective function of (P1) grant the problem information about the full degree distribution of the graph containing the spanning tree. However, we can recall from (10.11) and (10.10) that the $\mathcal{H}_2$ performance for the $\Sigma$ system can be viewed as the performance for the $\Sigma$ system with an additive term that encompasses the contribution of the cycle states. Consider then, the quantity

$$K = \frac{\mathcal{H}_2(\Sigma, E)}{\mathcal{H}_2(\Sigma, E)}$$

which is a measure of how well the performance as measured by the spanning tree states represents the graph performance including cycle information. For the spanning tree in Figure we have $K \approx 0.66$, and for the spanning tree in Figure we have $K \approx 0.24$. From these ratios we can make two observations pertaining to the performance of networks with cycles and their spanning trees. First, spanning trees that more accurately reflect the true degree distribution of the parent graph will have a higher $K$. Secondly, it is apparent that a significant portion of the $\mathcal{H}_2$ performance can come from the cycle contributions, so in general the performance of a given spanning tree may not be a good indicator of the full network performance. However, this result is suggestive that for graphs with few cycles, a spanning tree that reproduces the degree distribution of the full graph as closely as possible may be a good approximation for the full network performance.

The time scale assignment problem considered here demonstrates that while the heuristic developed from results on tree graphs appears to hold for more complex graph topologies, the performance of a given spanning tree does not necessarily reflect the performance of the full tree. In the next section, we will consider a reformulation of this problem that allows for a decentralized gradient update for achieving a similar time scale assignment scheme.

![Figure 10.3: Spanning tree of the graph in Figure 10.2 used to calculate the optimal time scale distribution.](image)
10.2.3 Decentralized Gradient Updates for Optimal $\mathcal{H}_2$ Performance

The main goal of this section is to motivate and discuss a decentralized update scheme on the network timescales to optimize the $\mathcal{H}_2$ norm. In cyber-physical systems, a chief concern is preventing system disruption by an antagonist - here, we have been considering this antagonist’s choice of weapon to be noise injected into the system. This can take the form of signal jammers, corruption of measurements, GPS spoofing, etc. In response to such a disturbance, it is of interest for the network to be able to quickly, and autonomously, adapt parts of the network to minimize the effect of this influence. Consider Problem (P2),

$$\min_{\epsilon_1, \ldots, \epsilon_n} \quad \text{tr} \left( R^T X R \right) + \frac{h}{2} \left\| E \right\|_2^2$$

subject to

$$\frac{1}{\epsilon_{\max}} \leq \frac{1}{\epsilon_i} \leq \frac{1}{\epsilon_{\min}} \quad \forall i \in N$$

$$X = \frac{1}{2} L_{e,s}^e.$$ 

(P2)

This is an $\mathcal{H}_2$-norm minimization on the time scale portion of the separated $\mathcal{H}_2$ performance, discussed in §10.2.2. In lieu of the sum constraint $\mu \leq \sum_i \epsilon_i^{-1}$, we introduce a regularization term $2^{-1} h \| E \|_2^2$ which achieves a similar goal of penalizing large timescales which lead to the trivial result of all nodes at the slowest allowable time scale. As we will show below, this regularizer has the added benefit that it will be amenable to a decentralized gradient update scheme to solve Problem (P2).
Proposition 6 (Gradient Update Scheme). Consider the setting of Problem (P2). Then, a decentralized algorithm to solve Problem (P2) is given by the projected gradient descent scheme on the solution variables \( \{\epsilon_1^{-1}, \ldots, \epsilon_n^{-1} \} \)

\[
\frac{1}{\epsilon_{i}^{k+1}} = \text{Proj}_C \left( \frac{1}{\epsilon_i^k} - \frac{1}{h\sqrt{\epsilon_i^k}} \frac{\partial f}{\partial (\epsilon_i^k)} \left( \frac{1}{\epsilon_i^k} \right) \right)
\]

\[
f \left( \frac{1}{\epsilon_i^k} \right) = \frac{1}{2} \left( \text{tr} \left( R^T L_{\epsilon_i^k}^T R \right) + h\|E\|_2^2 \right)
\]

\[
\frac{\partial f}{\partial (\epsilon_i^k)} \left( \frac{1}{\epsilon_i^k} \right) = \left( \frac{\deg(v_i)}{2} - h\left( \epsilon_i^k \right)^3 \right)
\]

where the projection on the constraints \( C \) is simply the projection onto the 'box constraints' \( \epsilon_{\max}^{-1} \leq \epsilon_i^{-1} \leq \epsilon_{\min}^{-1} \), and the exponents of \( \epsilon_i \)'s denote the update step.

Proof. Noting that \( E := E(\epsilon_i^k) \), we can write the objective function in Equation (10.13) as

\[
f \left( \frac{1}{\epsilon_i^k} \right) = \frac{1}{2} \left( \text{tr} \left( R^T D_r^T E^{-1}(\epsilon_i^k)^{-1} D_r R \right) + h \sum_{i=1}^{n} \left( \epsilon_i^k \right)^2 \right).
\]

A standard matrix calculus identity \([123]\) yields the gradient:

\[
\frac{\partial f}{\partial (\epsilon_i^k)} \left( \frac{1}{\epsilon_i^k} \right) = \text{tr} \left( DRR^T D_r^T E^{-1} \epsilon_i^k \right) - h \left( \epsilon_i^k \right)^3
\]

\[
= \left[ D_r RRT \right]_{ii} - h \left( \epsilon_i^k \right)^3.
\]

Next, since \( D = D_r R \), we have that

\( D_r RRT D_r^T = DD^T = L \),

and so \( \left[ D_r RRT D_r^T \right]_{ii} = L_{ii} = \deg(v_i) \). \( \Box \)

Remark 11. The gradient update scheme in Proposition 6 is truly decentralized, because the gradient in Equation (10.14) only depends on the local node time scale parameter \( \epsilon_i^k \), and the degree of the \( i \)th node. Both of these quantities are locally known to the \( i \)th node at time step \( k \).

10.2.4 Decentralized Gradient Updates on Weights for Optimal \( \mathcal{H}_2 \) Performance

Consider Problem (P3),

\[
\min_{\{W_i\}_{i=1}^{|E|}} \text{tr} \left( R^T (RWR^T)^{-1} R \right) + \frac{h}{2} \sum_{e \in E} \text{tr} \left[ W_e^T W_e \right]^2
\]

\[\text{s.t. } W_{\min} \leq W_e \leq W_{\max}, \forall e \in E \]

\[W = \text{blkdiag}(W_i).\]
Proposition 7 (Gradient Update for Edge Weights). The gradient of the cost function with respect to the edge weight $W_e$ in Problem ($P_3$) is given by

$$
\nabla_{W_e} f[H] = -\text{deblk}_e^k [Q^T Q] + h W_e,
$$

where $\text{deblk}_e^k [Q^T Q]$ is the $e$th $k \times k$ diagonal block of $Q^T Q$, and $Q$ is given by

$$
Q = R^T (RW^c R^T)^{-1} R
$$

$$
W^e_H = \text{blk}_e^k (H) + \sum_{l \in E \setminus c} \text{blk}_l^k (W_l),
$$

where

$$
\text{blk}_e^k (H) = e_e H e_e^T = (e_e \otimes I_k) H (e_e^T \otimes I_k)
$$

denotes the $kn \times kn$ matrix with $H$ on the $c$th $k \times k$ diagonal block, with zeros otherwise.

A gradient update scheme for solving Problem ($P_3$) is therefore

$$
W^{k+1}_e = W^k_e - \frac{1}{h \sqrt{k}} \nabla_{W_e} f[H]
$$

$$
= W^k_e - \frac{1}{h \sqrt{k}} \left( h W_e - \text{deblk}_e^k [Q^T Q] \right).
$$

Proof. We first identify the gradient of the cost function with respect to $W_c$, the weight on the $c$th edge in $E$. To this end, consider the functions

$$
f(X) = R \left[ e_c X e_e^T + \sum_{l \in E \setminus c} e_l W_l e_l^T \right] R^T
$$

$$
= R \left[ \text{blk}_e^k (X) + \sum_{l \in E \setminus c} \text{blk}_l^k (W_l) \right] R^T
$$

$$
g(X) = X^{-1}
$$

$$
h(X) = \text{tr} \left[ R^T X R \right].
$$

where

$$
\text{blk}_l^k (W_l) = e_l W_l e_l^T = (e_l \otimes I_k) W_l (e_l^T \otimes I_k)
$$

denotes the $kn \times kn$ matrix with $W_l$ on the $l$th $k \times k$ diagonal block, with zeros otherwise. Then, the cost function with $W_e$ as the argument, is given by

$$
\text{tr} \left[ R^T \left( R W R^T \right)^{-1} R \right] = (h \circ g \circ f) (W_e).
$$
These functions have differentials

\[
d f_X[H] = R \left[ \text{blk}_k^k(H) \right] R^T
\]
\[
d g_X[H] = -X^{-1}HX^{-1}
\]
\[
d h_X[H] = \text{tr} \left[ R^T HR \right].
\]

By the chain rule, we have that

\[
d (\circ f) X[H] = -Y^{-1} \text{Rblk}_k^k(H) R^T Y^{-1}
\]
\[
Y \triangleq f[H] = R \left[ \text{blk}_k^k(H) + \sum_{l \in \mathcal{E} \setminus c} \text{blk}_l^l(W_l) \right] R^T
\]
\[
\triangleq \text{RW}_H^c R^T,
\]

and so

\[
d (\circ f) X[H] = -\left( \text{RW}_H^c R^T \right)^{-1} \text{Rblk}_k^k(H) R^T \left( \text{RW}_H^c R^T \right)^{-1}.
\]

Similarly, we have

\[
d (h \circ g \circ f) X[H] = -\text{tr} \left[ Q \text{blk}_k^k(H) Q^T \right]
\]
\[
Q \triangleq R^T \left( \text{RW}_H^c R^T \right) R.
\]

Hence, we can write

\[
d (h \circ g \circ f) X[H] = -\text{tr} \left[ Q \text{blk}_k^k(H) Q^T \right]
\]
\[
= -\text{tr} \left[ Qe_c H e_c^T Q^T \right]
\]
\[
= -\text{tr} \left[ e_c^T Q^T Q e_c H \right]
\]
\[
= \langle -e_c^T Q^T Q e_c, H \rangle,
\]

and so the gradient of the cost function with respect to the cth weight \( W_c \) is identified as the cth \( k \times k \) diagonal block of \( -Q^T Q \).

\[\square\]
Flocking is a behaviour exhibited by certain multi-agent systems that are coordinating their motion into a cohesive formation, for example birds or stampeding buffalo. A consensus-type algorithm can be proposed that allows a system of $n$ agents to agree on their velocity vector while maintaining a separation from their neighbours [75].

11.0.1 Scalar State Case

Consider the protocol

$$\epsilon_i \dot{x}_i = - \sum_{j \in N(i)} w_{ij}(x_i - x_j - d) - \sum_{j \in N(i)} w_{ij}(\dot{x}_i - \dot{x}_j).$$

This is double-integrator consensus model that seeks to achieve a separation of distance $d$ between neighbours $i$ and $j$, as well as consensus on the velocity $\dot{x}_i$ and $\dot{x}_j$. In matrix form with disturbances and no constant separation input, this protocol is given by

$$\begin{bmatrix}
    \dot{\bar{x}} \\
    E \bar{x}
\end{bmatrix} = \begin{bmatrix} 0 & I \\ -L_{\theta_w} & -L_{\theta_w} \end{bmatrix} \begin{bmatrix} \bar{x} \\
    \dot{\bar{x}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ I & -D_{\theta} \end{bmatrix} \begin{bmatrix} \omega \\
    v \end{bmatrix}.$$ 

As before, we take the covariance of $\omega$ to be $\Omega = \sigma_\omega E^{1/2}$ and the covariance of $v$ to be $\Gamma = \sigma_v W^{1/2}$. We can examine the equivalent double-integrator edge consensus model.

**Theorem 31.** The double-integrator edge consensus model is given by

$$\begin{bmatrix}
    \dot{x}_e \\
    \dot{\bar{x}}_e
\end{bmatrix} = \begin{bmatrix} 0 & I \\ -L_{\theta_e R W R^T} & -L_{\theta_e R W R^T} \end{bmatrix} \begin{bmatrix} x_e \\
    \dot{x}_e \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ D_{\theta} E^{-1} & -L_{\theta_e R} \end{bmatrix} \begin{bmatrix} w \\
    v \end{bmatrix},$$
and so the double-integrator consensus on the edge states of the chosen spanning tree $\tau$ is given by

\[
\begin{bmatrix}
\dot{x}_\tau \\
\dot{\bar{x}}_{\tau}
\end{bmatrix} =
\begin{bmatrix}
0 & I \\
-L_{G_e} R W R^T & -L_{G_e} R W R^T
\end{bmatrix}
\begin{bmatrix}
x_{\tau} \\
\bar{x}_{\tau}
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 \\
D^T_{\tau} & -L_{G_e} R
\end{bmatrix}
\begin{bmatrix}
\omega \\
v
\end{bmatrix}.
\]

Proof. Applying the coordinate transform $x_e = S_v x$ and following a similar calculation as in Theorem 30 yields the result.

We can also explicitly compute the form of the $H_2$ norm for the double-integrator consensus.

**Theorem 32.** The controllability gramian for the time scaled double-integrator consensus is given by

\[
X^* = \frac{1}{2} \begin{bmatrix}
X_1 & 0 \\
0 & X_2
\end{bmatrix}
\]

\[
X_1 = \sigma_v^2 \left( RW R^T \right)^{-1} \mathcal{L}_{G_e}^{-1} \left( RW R^T \right)^{-1} + \sigma_v^2 \left( RW R^T \right)^{-1}
\]

\[
X_2 = \sigma_v^2 \left( RW R^T \right)^{-1} + \sigma_v^2 \mathcal{L}_{G_e}.
\]

Furthermore, the blocks $X_1, X_2$ of $X^*$ correspond to the position and velocity states, respectively. Hence, one can consider the $H_2$ performance of the position and velocity states separately or aggregately by examining the $H_2$ norms

\[
H_2^{(1)} = \frac{1}{2} \text{tr} \left[ R^T X_1 R \right], 
H_2^{(2)} = \frac{1}{2} \text{tr} \left[ R^T X_2 R \right]
\]

\[
H_2^{(1)} = \frac{1}{2} \text{tr} \left[ R^T (X_1 + X_2) R \right].
\]

Proof. From the dynamics (9.3), the controllablity gramian is given by the positive semi-definite solution to the Lyapunov equation

\[
A X^* + X^* A^T + BB^T = 0,
\]

where $X^*, A, BB^T$ are given by

\[
A = \begin{bmatrix}
0 & I \\
-L_{G_e} R W R^T & -L_{G_e} R W R^T
\end{bmatrix}
\]

\[
BB^T = \begin{bmatrix}
0 & 0 \\
0 & \sigma_v^2 \mathcal{L}_{G_e} + \sigma_v^2 \mathcal{L}_{G_e} R R^T \mathcal{L}_{G_e}
\end{bmatrix}
\]

\[
X^* = \begin{bmatrix}
X_1 & X_3 \\
X_3 & X_2
\end{bmatrix}.
\]
Solving Equation (9.1) yields

\[ X_3 = 0 \]
\[ X_2 = \sigma_w^2 \left( RWRT \right)^{-1} + \sigma_v^2 L_{\mathcal{G}_e} \]
\[ X_2 = X_1 RWRT L_{\mathcal{G}_e}. \]

and solving for \( X_1 \) in the last display yields

\[ X_1 = \sigma_w^2 \left[ \left( RWRT \right)^{-1} L_{\mathcal{G}_e}^{-1} \left( RWRT \right)^{-1} \right] + \sigma_v^2 \left( RWRT \right)^{-1}. \]

To measure the position, velocity, or both states for consideration in the \( \mathcal{H}_2 \) norm, one can choose the observation matrices

\[
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix},
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix},
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix}
\]

respectively.

### 11.0.2 Matrix-Valued Double Integrator Consensus

In the case of vector-valued states, we can write the dynamics as

\[
\begin{bmatrix}
\dot{x} \\
Ex
\end{bmatrix} = \begin{bmatrix}
0 & I \\
-L_{\mathcal{G}_w} & -L_{\mathcal{G}_w}
\end{bmatrix} \begin{bmatrix}
x \\
\dot{x}
\end{bmatrix} + \begin{bmatrix}
0 & 0 \\
I & -D_{\mathcal{G}}
\end{bmatrix} \begin{bmatrix}
\omega \\
v
\end{bmatrix}.
\]

**Theorem 33.** The double-integrator edge consensus model is given by

\[
\begin{bmatrix}
\dot{x}_e \\
\dot{\bar{x}}_e
\end{bmatrix} = \begin{bmatrix}
0 & 0 & I \\
-L_{\mathcal{G}_e}RWTR & 0 & -L_{\mathcal{G}_e}RWTR & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
x_e \\
\bar{x}_e
\end{bmatrix} + \begin{bmatrix}
0 & 0 \\
0 & 0 \\
D_T E^{-1} & -L_{\mathcal{G}_e}R \\
\Xi^{-1}_{\mathcal{G}_e} T & 0
\end{bmatrix} \begin{bmatrix}
\omega \\
v
\end{bmatrix},
\]

and so the double-integrator consensus on the edge states of the chosen spanning tree \( \tau \) is given by

\[
\begin{bmatrix}
\dot{x}_r \\
\dot{\bar{x}}_r
\end{bmatrix} = \begin{bmatrix}
0 & I \\
-L_{\mathcal{G}_e}RWTR & -L_{\mathcal{G}_e}RWTR
\end{bmatrix} \begin{bmatrix}
x_r \\
\dot{x}_r
\end{bmatrix} + \begin{bmatrix}
0 & 0 \\
0 & 0 \\
D_T & -L_{\mathcal{G}_e}R
\end{bmatrix} \begin{bmatrix}
\omega \\
v
\end{bmatrix}.
\]

**Proof.** Applying the coordinate transform \( x_e = S_c x \) and following a similar calculation as in Theorem 30 yields the result. □
We can also explicitly compute the form of the $\mathcal{H}_2$ norm for the matrix-weighted double-integrator consensus.

Theorem 34. The controllability gramian for the time scaled double-integrator consensus is given by

\[
X^* = \frac{1}{2} \begin{bmatrix} X_1 & 0 \\ 0 & X_2 \end{bmatrix}
\]

\[
X_1 = \sigma_w^2 \left( (RWR^T)^{-1} L_{\theta_e}^{-1} \left( RWR^T \right)^{-1} \right) + \sigma_v^2 \left( RWR^T \right)^{-1}
\]

\[
X_2 = \sigma_w^2 \left( RWR^T \right)^{-1} + \sigma_v^2 L_{\theta_e}.
\]

Furthermore, the blocks $X_1, X_2$ of $X^*$ correspond to the position and velocity states, respectively. Hence, one can consider the $\mathcal{H}_2$ performance of the position and velocity states separately or aggregately by examining the $\mathcal{H}_2$ norms

\[
\mathcal{H}_2^{(1)} = \frac{1}{2} \text{tr} \left[ R^T X_1 R \right], \quad \mathcal{H}_2^{(2)} = \frac{1}{2} \text{tr} \left[ R^T X_2 R \right]
\]

\[
\mathcal{H}_2 = \frac{1}{2} \text{tr} \left[ R^T (X_1 + X_2) R \right].
\]

Proof. From the dynamics (9.9), the controllability gramian is given by the positive semi-definite solution to the Lyapunov equation

\[
AX^* + X^* A^T + BB^T = 0,
\]

where $X^*, A, BB^T$ are given by

\[
A = \begin{bmatrix} 0 & I \\ -L_{\theta_e} RWR^T & -L_{\theta_e} RWR^T \end{bmatrix}
\]

\[
BB^T = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_w^2 L_{\theta_e} + \sigma_v^2 RR^T L_{\theta_e} \end{bmatrix}
\]

\[
X^* = \begin{bmatrix} X_1 & X_3 \\ X_3 & X_2 \end{bmatrix}
\]

Solving Equation (9.4) yields

\[
X_3 = 0
\]

\[
X_2 = \sigma_v^2 \left( RWR^T \right)^{-1} + \sigma_v^2 L_{\theta_e}
\]

\[
X_2 = X_1 RWR^T L_{\theta_e},
\]

and solving for $X_1$ in the last display yields

\[
X_1 = \sigma_w^2 \left[ (RWR^T)^{-1} L_{\theta_e}^{-1} \left( RWR^T \right)^{-1} \right] + \sigma_v^2 \left( RWR^T \right)^{-1}.
\]
To measure the position, velocity, or both states for consideration in the $\mathcal{H}_2$ norm, one can choose the observation matrices

$$
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix}, \quad
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix}, \quad
\begin{bmatrix}
R^T & 0 \\
0 & R^T
\end{bmatrix}
$$

respectively.

\[\square\]

11.0.3 Numerical Example

The weight update scheme applied to the second-order consensus problem was implemented numerically. The task assigned to the agents was to use the second-order consensus protocol to achieve the formation shown in Figure 11.1—a formation assigned by sampling discrete points on a 2D spiral.

Consensus on the formation is achieved by the second-order protocol with a constant signal $d$ specifying the position in the formation:

$$
\dot{x}_i = - \sum_{j \in N(i)} W_{ij}(x_i - x_j - d) - \sum_{j \in N(i)} W_{ij} (\dot{x}_i - \dot{x}_j).
$$

Since this is a constant signal, the $\mathcal{H}_2$ performance of the edge states remains the same as in the previous section.

The initial selection of weights was chosen at random using the generator

$$
W_{(i,j)} = \alpha \left[ \frac{G_{(i,j)} + G_{(i,j)}^T}{2} + 2I_2 \right]
$$

where $G_{(i,j)}$ is a $2 \times 2$ matrix with entries distributed according to a zero-mean standard Gaussian, and $\alpha = 0.3$ was chosen arbitrarily to yield a suboptimal initial selection of weights. The upper and lower bounds on the weights were chosen with the same generator in Equation (9.7), but with $\alpha_l = 0.05$ and $\alpha_u = 10$. The penalty parameter was chosen as $h = 0.01$. 

Figure 11.1: Relative formation of the agents, defined by discrete points on a spiral
Figure 11.2: Cost of weight selection versus iteration

Figure 11.3: Visualization of edge weights at $k = 1$

The gradient descent algorithm from Proposition 7 converges in a few iterations, as shown in Figure 11.2, and intermediate graph weights are visualized in Figure 11.7. The 3 independent parameters of each weight are here visualized as 3 edges in a multigraph. None of the optimal edge weights saturated the upper and lower bounds in this setup.

These weights were then used in a simulation of the dynamics in Equation (9.11) over a time span of 30 seconds. At $10 \leq t \leq 20$, the formation is subject to a 'gust' of noise on the edges and self-dynamics of covariance $\sigma_v^2 I$ and $\sigma_w^2 W$, with $\sigma_v = \sigma_w = 5$. In one simulation, no weight update was performed, and in the other, the updated weights were used during the wind gust. The edge states are shown in Figure 11.8, and the variance away from the consensus value $x_e(t_f)$

$$\text{Var}[x_e(t)] := [x_e(t) - x_e(t_f)]^2$$

is shown in Figure 11.9. The updated weights clearly outperform the initial, suboptimal choice of weights.
Figure 11.4: Visualization of edge weights at $k = 2$

Figure 11.5: Visualization of edge weights at $k = 3$

Figure 11.6: Visualization of edge weights at $k = 4$
Figure 11.7: Visualization of edge weights over iterations $k = 1$ (top left), 2 (top right), 3 (bottom left) & 4 (bottom right). Each of the 3 independent parameters of the $2 \times 2$ matrix-valued weight is visualized in a multigraph.
Figure 11.8: Edge states over time subjected to gust at $10 \leq t \leq 20$; top two plots are the edge states in the $x, y$ directions respectively with the weight update, and the bottom two plots are the edge states in the $x, y$ directions respectively with no weight update.
Figure 11.9: Variance from consensus for a representative edge \([x_e(t) - x_e(t_f)]^2\) for the case with the weight update and without.
Networked dynamic systems are ubiquitous in many areas of engineering, and one can argue that they are of paramount importance for understanding complex phenomena in a wide range of disciplines including sociology (social networks [124]), physics (quantum networks [125], [126]), and biology (gene networks [127]).

As such, there has been a surge of research activities in distinct scientific areas with two fundamental questions at their core. Firstly, what is the role of the network and its structure in characterizing the behavior that these networked systems exhibit? Secondly, to what extent can this structure be used for the purpose of control?

Furthermore, it is desired to provide such a characterization with an eye towards large networks. Some of the such open problems are: to what extent can these characterizations be done efficiently as the number of nodes and edges in the network grow? Do the computational requirements corresponding to this characterization depend on the local network structure? Does the study of system measures lead to a conceptual framework to examine dynamic networks and their system theoretic properties [66]?

In this chapter, we highlight the importance of two measures associated with the network that we believe have not been examined more thoroughly in the context of their system theoretic significance. These measures quantify a notion of “information content” in the network— and as such, they are referred to as the loop entropy and the Kolmogorov-Sinai (KS) entropy. Both measures have been extensively employed in the physics literature—however, we believe they should be more systematically studied in the networked dynamic systems due to their inherent connections with basic system theory of networks; a related work in this direction include that of Siami and Motte [72].

The contribution of the chapter is twofold. First, we examine the connection between loop entropy and noise propagation properties of networks. The loop entropy is defined separately for both the structure of the network, as well as for how the dynamics respond to a noise input. We show in Section 12.3 that these entropies are the same for the consensus dynamics. Secondly, we consider the KS entropy in Section 12.4 in the context of networks evolving via their adjacency matrix. Some examples of entropies are given in Section 12.5.

12.2 Network Model

In this section, we provide the necessary background on the underlying model that will be considered in this chapter.
12.2.1 Mathematical Notation and Graph Theory

The identity matrix is defined as $I$ and $e_i$ is the column vector with all zeros except a 1 in the $i$th position. A symmetric matrix $A$ is positive semi definite if for all $v \in \mathbb{R}^n$, $v^T A v \geq 0$. The positive semi-definite ordering is defined on two symmetric matrices $A, B$ such that $A \succeq B$ if $A - B$ is positive semi-definite. The matrix $A^{(n-1)}$ is a matrix constructed from $A$ by removing the first row and column.

The fundamental algebraic structure that represents a network is the graph $G(V, E, W)$ with a vertex set $V$ of cardinality $|V| = n$, an edge set $E$ of cardinality $|E| = m$, and a set of weights $W$ on the edges. The edge set consists of pairs of vertices $i$ and $j$ that are called adjacent if $\{i, j\} \in E \subseteq V^2$, and this edge has weight $w_{ij} \in W_G$. The number of adjacent nodes to node $i$ is called the degree, and is denoted by $d_i$. The degree matrix is denoted as $\Delta = \text{Diag}(d_i)$. The weights define an $m \times m$ weight matrix $W_G$ with the $l$th edge $\{i, j\}$ having weight $w_{ij}$ on the diagonal entry $(W_G)_{ll}$. The adjacency matrix $A$ is defined such that $a_{ij}$ has weight $w_{ij}$ if there is an edge $\{i, j\}$ and 0 otherwise. The graph Laplacian $L_G$ is given by $\Delta - A$.

The graph is said to be connected if there exists a path of edges from every node to every other node. The graph Laplacian is a positive semi-definite matrix, and the multiplicity of the zero eigenvalue denotes the number of connected components of the graph. The eigenvalues of the Laplacian are ordered such that $0 = \lambda_1 \leq \cdots \leq \lambda_n$. Similarly, we order the eigenvalues of the adjacency matrix such that $\mu_1 \leq \cdots \leq \mu_n$.

12.2.2 Network Models

In this chapter, we consider four models. The first is the controlled consensus dynamics [5]:

$$\dot{x} = -L_G x(t) + B u(t) + G \omega(t),$$

(12.1)

where $\omega(t)$ is a zero-mean Gaussian random vector depicting a random disturbance to the network, $x(t)$ is the states of the nodes of the network, and $u(t)$ is the control input. For compactness of presentation, the time $t$ indication on the states will be henceforth removed.

The single integrator leader-follower dynamics can be interpreted as a leader agent $x_l$ which provides an input signal to the set of follower agents $x_f$ via their adjacent edges. This allows the graph Laplacian to be partitioned into two sub-Laplacians for the leaders and followers:

$$L_G = \begin{bmatrix} a & -B_G^T \\ -B_G & A_G \end{bmatrix}$$

(12.2)

with a perturbation matrix $B_G$ encoding the connections between the leader and the followers:

$$\dot{x}_f = -A_G x_f + B_G u + G \omega.$$  

(12.3)
Here we take $G = I$. The matrix $A_G$ is known as the Dirichlet matrix, or grounded Laplacian [128], [129]. It has the property of being strictly positive definite when the graph is connected, and hence is invertible unlike the original Laplacian $L_G$.

Similarly, we can define the double integrator dynamics as,

$$
\begin{bmatrix}
\dot{x}_f \\
\dot{x}_f
\end{bmatrix} =
\begin{bmatrix}
0 & I \\
-A_G & -A_G
\end{bmatrix}
\begin{bmatrix}
x_f \\
x_f
\end{bmatrix} +
\begin{bmatrix}
0 \\
B_G
\end{bmatrix} u +
\begin{bmatrix}
0 \\
G
\end{bmatrix} \omega 
$$

$$
: = -A_G
\begin{bmatrix}
x_f \\
x_f
\end{bmatrix} + B_G u + G \omega 
$$

The controlled adjacency dynamics are given by

$$
\dot{x} = Ax + Bu. 
$$

There has been a large body of work centered around finding connections between network structure and the system properties of the corresponding networked system. For example, it has been shown that symmetry structures in the graph topology undermine the controllability of the network [17]. Various notions of robustness measures have been introduced in an effort to describe the performance of networks, such as the convergence time $\tau = 1/\lambda_2$, and the utility of these measures often depends on the network topology [5], [66].

One quantity that captures system behavior with relation to system topology in other fields of science is that of entropy. Fundamentally, entropy is a measure that captures the randomness or disorder of a system. For example, a polymer can be modeled as a vector random walk where each iteration of the walk adds a monomer of some small length from tip-to-tail of the previous iteration [130]. If the polymer is extremely straight, its ‘disorder’ is very low and hence the entropy is small. On the other hand, a very disordered or loopy polymer has high entropy. The entropy of a polymer has a direct analytical effect on the dynamics of how the polymer behaves and evolves in a fluid at some temperature. In this chapter, we explore two notions of entropy defined for a network, and show that for the leader-follower dynamics we can indeed link the topological disorder to control-theoretic properties of the network.

12.3 Loop Entropy and Spanning Trees

In this section, we define the standard notion for the entropy of a network. Using a well-known result by Kirchoff, we show that this notion of entropy is related to the number of spanning trees in the graph representing the Laplacian. Although a similar line of inquiry has been pursued in [72] (for a different notion of entropy) and [5], there are several interpretations of this result with respect to leader-follower dynamics that have not been explored in the literature.

---

1 This can be done without loss of generality since if $G$ is non-identity, the effect of multiplying it with $\omega$ can be propagated into the covariance of the noise of $\omega$. 
Definition 6. The entropy of a network $\mathcal{G}$ is given by the sum $S_\mathcal{G} = \sum_{i=2}^{n} \log \lambda_i$.

The non-zero eigenvalues of $L_\mathcal{G}$ are related to those of a submatrix $L_\mathcal{G}^{(n-1)}$ of $L_\mathcal{G}$. For clarity, we will refrain from connecting $L_\mathcal{G}^{(n-1)}$ to $A_\mathcal{G}$ as seen in the partition in (12.2) until we complete the exposition of the properties of $S_\mathcal{G}$. We will make this connection at the end of the section. The matrix $L_\mathcal{G}^{(n-1)}$ is constructed by removing an arbitrary $i$th row and column of $L_\mathcal{G}$.

The eigenvalues of $L_\mathcal{G}^{(n-1)}$ are strictly positive, and satisfy:

$$\frac{1}{n} \lambda_2(L_\mathcal{G}) \cdots \lambda_n(L_\mathcal{G}) = \det L_\mathcal{G}^{(n-1)} = \lambda_1(L_\mathcal{G}^{(n-1)}) \cdots \lambda_{n-1}(L_\mathcal{G}^{(n-1)}).$$

It is important to note that the eigenvalues of $L_\mathcal{G}^{(n-1)}$ are all positive because it is a Dirichlet matrix (or grounded Laplacian) of $L_\mathcal{G}$. It should now be clear that

$$S_\mathcal{G} = \sum_{i=2}^{n} \log \lambda_i(L_\mathcal{G}) = \sum_{i=1}^{n-1} \log \lambda_i(L_\mathcal{G}^{(n-1)}) + \log n.$$ We can connect this definition to a well-known result by Kirchoff. First, recall that a tree is a connected undirected graph with no cycles. Furthermore, a spanning tree of a graph $\mathcal{G}$ is a tree containing every vertex of $\mathcal{G}$. Let $\tau(\mathcal{G})$ denote the number of unique spanning trees of $\mathcal{G}$. A well-known result by Kirchoff states that the $\det L_\mathcal{G}^{(n-1)} = \tau(\mathcal{G})$. The following observation is thus immediate.

**Theorem 35.** Let $L_\mathcal{G}$ be the unweighted network Laplacian for the consensus dynamics. The network entropy is given by $S_\mathcal{G} = \log (n \tau(\mathcal{G}))$.

Theorem 35 motivates a new paradigm in terms of interpreting the entropy measure of a networked system. We can now begin to view the entropy as being a fundamentally topological quantity. This formulation of entropy also has a very elegant analogy to Boltzmann entropy in statistical physics. Boltzmann defined the entropy of a physical system, say a gas, to be $S = k_b \log W_s$ where $W_s$ is the number of microstates of the physical system and $k_b$ is a constant [131]. The formula given in Theorem 35 is the number of possible paths a signal can take to propagate throughout the network. In this sense, we have a notion of systemic ‘disorder’ that captures the idea of enumerating possible states of the system, where in our case the quantity evolving the system is the control signal.

The system-theoretic interpretation motivating the examination of the determinant of the submatrix $L_\mathcal{G}^{(n-1)}$ in the entropy formula is the following. We take our system and designate a single node as the leader, and the rest of the nodes as followers. If we enumerate the leader node to correspond to the first vertex, then our Laplacian matrix is partitioned exactly according to:

$$L_\mathcal{G} = \begin{bmatrix} a & \delta^T \\ \delta & L_\mathcal{G}^{(n-1)} \end{bmatrix}$$
where \( \mathcal{A}_G = L_G^{(n-1)} \) in the dynamics (12.3), and the formula of entropy is \( S_G = \log \det \mathcal{A}_G \). In the next section, we will motivate another graph-centric notion of this entropy, and then discuss how these notions relate to graph controllability.

12.3.1 Counting Loops

In this section, we describe a method of associating the notion of entropy to a notion of loops in the network. We do this indirectly: we identify the Dirichlet matrix of a network Laplacian to a non-unique flow graph. For the purposes in this chapter, we can consider a flow graph to be a directed graph with self-loops on each node. From this flow graph, the usual notion of the Coates Determinant is then examined; we will see that this will depend on the number of loops in the flow graph. This will shed light on a relation between the number of loops in the flow graph and the number of spanning trees.

Definition 7. A connection \( C \) of a flow graph \( G \) is a subgraph of \( G \) with the following properties:

1. Each node of \( C \) is in \( G \)
2. Each node of \( C \) has a single edge originating from it and terminating at it.

Definition 8. A directed loop of \( G \) is a subgraph of \( G \) whose edges \( b_1, \ldots, b_l \) can be ordered such that

1. The tip of \( b_k \) is the origin of \( b_{k+1} \), for \( k = 1, \ldots, l - 1 \)
2. The origin of \( b_1 \) is the tip of \( b_l \).
3. Each node tip along the loop is encountered only once.

A fundamental result by Coates allows us to compute the determinant of \( L_G^{(n-1)} \) using a digraphic interpretation of this Dirichlet matrix. We can construct the flow graph corresponding to \( L_G^{(n-1)} \) as follows: Take each diagonal element \((L_G^{(n-1)})_{ii}\) and identify it to a self-loop on the \( i \)th node with weight \((L_G^{(n-1)})_{ii}\). Then, each node \( i \) is connected to node \( j \) in one direction with weight \((L_G^{(n-1)})_{ij}\) and in the other direction with weight \((L_G^{(n-1)})_{ji}\).

Lemma 18 (Coates Determinant). Let \( G^{(n-1)} \) be the flow graph associated with the Dirichlet matrix \( L_G^{(n-1)} \). Let the subscript \( \rho \) denote a connection of the flow graph \( G \). Then, let the total gain \( C(G)_{\rho} \) of the connection \( \rho \) be the product of all the edge weights \((L_G^{(n-1)})_{ij}\) in the connection \( \rho \). Finally, let \( L_{\rho} \) be the number of directed loops in the connection \( \rho \). Then, the determinant of \( L_G^{(n-1)} \) is given by:

\[
\det L_G^{(n-1)} = \Delta_c = (-1)^{n-1} \left( \sum_{\rho} (-1)^{L_{\rho}} C(G^{(n-1)})_{\rho} \right),
\]

where the sum is taken over all connections \( \rho \).
12.3 Loop Entropy and Spanning Trees

Figure 12.1: The two flow graphs of the transfer function of a 4-cycle controlled from a single node.

**Proof.** A proof of this lemma is given in [132].

This determinant formula is interesting because it depends on the number of directed loops in the flow graph - a topological feature.

**Theorem 36.** Let $L_G$ be the Laplacian for the consensus dynamics; let $G^{(n-1)}$ be the flow graph associated with the Dirichlet matrix $L^{(n-1)}_G$. Then,

$$S_G = -\log \left( (-1)^n \sum_{\rho} (-1)^\rho C(G^{(n-1)})_{\rho} \right).$$

An application of the Coates determinant formula is for computing the transfer function of the controlled consensus dynamics (12.1). Assuming the system output is given as $y = B_T^T x$, the transfer function for the network assumes the form,

$$T(s) = \frac{sI + L_G}{sI + B_G}.$$

(12.6)

To use the Coates determinant formula, we must create two flow graphs corresponding to the numerator and denominator of $H(s)$. The flow graph of the denominator will have self-loops at each $i$th node with gain $s + L_G_{ii}$, and each edge $\{i, j\}$ of $L_G$ will be replaced by a digon (pair of directed edges pointing in opposite directions) with gains $L_G_{ij}$. The numerator will have an additional node for each column of $B$ (corresponding to separate input channels) connected with a digon to each leader node with gains $\pm 1$. An example is shown in Figure 12.1, where a 4-cycle is controlled, with $B = [1, 0, 0, 0]^T$. The center and right flow graphs correspond to the denominator and numerator in (12.6), respectively. The Coates formula provides the coefficients of the polynomials in the transfer function, which in this case is

$$T(s) = \frac{4 + 10s + 6s^2 + s^3}{16s + 20s^2 + 8s^3 + s^4} = \frac{(2 + s)(2 + 4s + s^2)}{s(2 + s)^2(4 + s)}.$$

12.3.2 Loop Entropy and Network Gramians

In this section, we highlight a result that connects the loop entropy to controllability properties of the consensus dynamics. Recalling the previous discussion highlighting the two interpretations of the loop entropy submatrix $L^{(n-1)}_G$, without loss of generality let us assume that the loop entropy takes the form $S_G = \log \det A_G = \sum_{i=1}^n \log \lambda_i(A_G)$.

---

2 This system is uncontrollable to illustrate the pole-zero cancellation.
Definition 9 (Controllability Gramian). Consider the full single integrator consensus dynamics driven by white noise. Then, the controllability Gramian is the unique positive semi-definite matrix $\Sigma$ satisfying

$$-\mathcal{A}_G \Sigma - \Sigma \mathcal{A}_G^T = -GG^T. \quad (12.7)$$

The controllability Gramian describes the mapping from inputs to steady state behavior of the internal states. In the meantime, it turns out that there is an intimate relationship between the controllability Gramian and the loop entropy of the system.

Theorem 37 (Controllability Gramian and Loop Entropy). Let $\mathcal{A}_G$ be the Dirichlet matrix of $L_G$ in the full single integrator dynamics (12.3), let $\Sigma$ be its controllability Gramian, and let $G = I$. Then, the loop entropy is given by

$$S_G = \log \left( 2^n \det \Sigma \right).$$

Proof. Since $\mathcal{A}_G$ is positive definite, the controllability Gramian satisfies (12.7), and hence is given by $\Sigma = \frac{1}{2} \mathcal{A}_G^{-1}$ and so $\left( 2^n \det \Sigma \right)^{-1} = \det \mathcal{A}_G$. Taking the logarithm of both sides yields the result. $\Box$

Theorem 38 (Loop Entropy for the Double Integrator). Let $\mathcal{A}_G$ be the dynamics matrix from the full double integrator dynamics (12.4):

$$\mathcal{A}_G = \begin{bmatrix} 0 & -I \\ -\mathcal{A}_G & \mathcal{A}_G \end{bmatrix}. \quad (12.4)$$

Next, let $\Sigma$ be the controllability Gramian satisfying

$$-\mathcal{A}_G \Sigma - \Sigma \mathcal{A}_G^T = -G \bar{G}^T,$$

where $\bar{G} = [0, I]^T$. Then, the loop entropy satisfies

$$S_G = \log \left( 2^{-2n} \det \Sigma^{-1} \right).$$

Proof. The controllability Gramian for the double integrator dynamics is

$$\Sigma = \frac{1}{2} \begin{bmatrix} I & 0 \\ 0 & \mathcal{A}_G^{-1} \end{bmatrix},$$

and a similar computation as from the previous proof yields the result that $S_G = \left( 2^{-2n} \det \Sigma^{-1} \right)$. $\Box$

The log determinant of the covariance matrix is proportional to the volume of the controllability ellipsoid. Recall that one interpretation of the controllability Gramian $\Sigma$ is that it is the steady state covariance of $x(t)$ when the consensus dynamics is driven by white noise. Hence, when the steady state covariance of the state is small, the entropy is large, and vice versa. In this sense, a more disordered network is more robust to noise. Recalling our result linking the entropy to the number of spanning trees, a high entropy also corresponds to a large number of spanning trees, or a large number of possible paths noise can take as it propagates throughout the network.
Intuitively, one can think of the leader-node symmetries, described by Rahmani et al [17], as being noise amplifiers, while disorder and large numbers of spanning trees act to dampen noise. Hence, when designing network topologies, it is beneficial to add edges between nodes such that entropy is maximized. In Section 12.5, we illustrate this with some examples.

Next, we will discuss solutions to the Lyapunov equation when $G \neq I$. Choose $G$ to be $\sum_{i \in S} e_i e_i^T$ for some set of indices $S$ corresponding to follower nodes. To do this, we define the covariance entropy.

**Definition 10 (Covariance Entropy).** Let $(-L^{(n-1)}_G, G)$ be as before, and a controllable pair. Denote $\Sigma_S$ to be the solution of the Lyapunov equation

$$-L^{(n-1)}_G \Sigma_S - \Sigma_S L^{(n-1)}_G = -GG^T.$$  

Define the covariance entropy to be

$$M_S = \log_2 n \det \Sigma_S^{-1}.$$  

Clearly, when $G = I$, the covariance entropy is equal to the loop entropy. For a certain class of networks, the covariance entropy has a natural ordering property.

**Theorem 39 (Covariance Entropy Ordering).** Construct $L^{(n-1)}_G$ as before, but with the stipulation that $(-L^{(n-1)}_G, G)$ be controllable for any $G = \sum_{i \in S} e_i e_i^T$ with indices $S$ corresponding to follower nodes. Let $M_S$ and $M_{S \cup \{k\}}$ denote the covariance entropies with input edge sets $S$ and $S \cup \{k\}$, respectively. Then, $M_S \geq M_{S \cup \{k\}}$. In particular, the loop entropy minimizes the covariance entropy.

**Proof.** As before, choose $G$ to be $\sum_{i \in S} e_i e_i^T$ for some set of indices $S$. Since $-L^{(n-1)}_G$ is stable, it follows that the solution to the Lyapunov equation

$$-L^{(n-1)}_G \Sigma_{\{i\}} - \Sigma_{\{i\}} L^{(n-1)}_G = -GG^T$$

is given by

$$\Sigma_{\{i\}} = \int_0^\infty e^{-tL^{(n-1)}_G} e_i e_i^T e^{-tL^{(n-1)}_G} dt$$

and is positive semi-definite and so $\Sigma_{S \cup \{k\}} - \Sigma_S = \Sigma_{\{k\}}$ is positive semi-definite. It follows that $\Sigma_{S \cup \{k\}} \geq \Sigma_S$. Lastly, since the system is controllable, $\log_2 n \Sigma_S^{-1} \leq \Sigma_{S \cup \{k\}}$. \qed

### 12.4 Kolmogorov-Sinai Entropy

In this section, we examine the Kolmogorov-Sinai entropy that is related to the adjacency matrix spectrum. We will also compute this entropy for a variety of simple graphs, which will elucidate the topological features it capture. Using results from [133], we will derive bounds on the value of this entropy and show that the bound is sharp for both regular and highly-irregular
graphs. At the end of this section, we will show the relation between the Kolmogorov-Sinai entropy and the stability of adjacency driven networks.

The Kolmogorov-Sinai entropy is a generalization of information-theoretic notion of the Shannon entropy on a network [134]. It characterizes the rate at which information is generated by the network. In the context of a network, information corresponds to a sequence of nodes visited by some Markov process on the network. The Kolmogorov-Sinai entropy is invariant under transformations that preserve the frequencies with which the network generates time-ordered sequences of nodes.

The stochastic process defining the information source is given by a Markov matrix $M$ satisfying the properties

$$p_{ij} \geq 0, \quad \sum_j p_{ij} = 1, \quad \text{and} \quad \pi = \pi P$$

where $\pi$ is the chain’s stationary distribution. Consider the set of stochastic matrices $M_A$, where $A$ is the adjacency matrix of the network graph, that have the property $a_{ij} = 0$ if and only if $p_{ij} = 0$. Let $\mu_n$ be the dominant eigenvalue of $A$ with eigenvector $v$. Let $M_A$ be a family of stochastic matrices with the property that $p_{ij} = 0$ if and only if $a_{ij} = 0$ in the network adjacency matrix. The Kolmogorov-Sinai Entropy defined presently is a measure that satisfies a variational principle analogous to that of the Gibbs principle in statistical mechanics [135]:

$$\log \mu_n = \sup_{P \in M_A} \left[ -\sum_{ij} \pi_i p_{ij} \log p_{ij} + \sum_{ij} \pi_i p_{ij} \log a_{ij} \right].$$

The supremum over all admissible stochastic matrices is given by the unique matrix $P$ satisfying $p_{ij}^* = \frac{a_{ij}v_i}{\mu_n v_i}$, and hence we have

$$\log \mu_n = -\sum_{ij} \pi_i p_{ij}^* \log p_{ij} + \sum_{ij} \pi_i p_{ij}^* \log a_{ij}. \quad (12.8)$$

By convention, $0 \log 0 = 0$. If the adjacency matrix is unweighted, in the sense that $a_{ij} = 1$ if there is an edge at $(i, j)$ and $a_{ij} = 0$ otherwise, then the term on the right hand side of (12.8) is zero. The term remaining is called the dynamical entropy of the Markov process, and is given by

$$H_P = -\sum_{i=1}^N \pi_i \sum_j p_{ij}^* \log p_{ij}^*.$$

The Kolmogorov-Sinai Entropy is defined as follows [134]:

**Definition 11 (Kolmogorov-Sinai Entropy).** The Kolmogorov-Sinai Entropy of a weighted adjacency matrix $A$ is defined by

$$H = \log \mu_n - \sum_{ij} \pi_i p_{ij}^* \log a_{ij},$$

where $p_{ij}^* = \frac{a_{ij}v_j}{\mu_n v_i}$. For an unweighted graph, the right-hand side vanishes:

$$H = \log \mu_n.$$
In the next part of this section, we will give some bounds on the value of the KS entropy. For the remainder of the chapter, we will assume unweighted adjacency matrices, and hence the KS-entropy is given by $H = \log \mu_n$.

### 12.4.1 Bounds on the Kolmogorov-Sinai Entropy

The Kolmogorov-Sinai entropy is difficult to compute in closed form for an arbitrary graph. Using results by Nikiforov [133], we can compute upper bounds on the Kolmogorov-Sinai entropy which is given in terms topological quantities such as edge count and minimum degree. We can now recall the two theorems by Nikiforov.

**Definition 12 (K$_{p+1}$-Free Graphs).** The graph $G$ is called $p+1$-free if it does not contain a complete subgraph on $p+1$ vertices.

**Theorem 40 (Edge Bound).** Let $G$ be $p+1$-free. Then, the largest eigenvalue $\mu_n$ of the adjacency matrix of $G$ satisfies

$$\mu_n \leq \frac{\sqrt{2|E| - n\delta} + (\delta + 1)^2}{4}.$$  

By the definition of the Kolmogorov-Sinai entropy, we immediately have the following theorem:

**Theorem 42 (Bounds on KS-Entropy).** Let $G$ be $p+1$-free with minimal degree $\delta$. Then, the Kolmogorov-Sinai entropy satisfies the two inequalities:

$$H \leq \frac{1}{2} \log \left[2 \frac{p - 1}{p} |E| \right]$$

$$H \leq \log \left[\frac{\delta - 1}{2} + \frac{\sqrt{2|E| - n\delta} + (\delta + 1)^2}{4}\right].$$

Equation (12.9) is in fact tight for regular graphs, and also highly irregular graphs in the sense of maximizing the degree variance. For example, let $n \geq k$, and let $H_{n,k}$ be the $n$-node complement of the complete graph on $n-k$ nodes $K_{n-k}$. Then, the maximal adjacency eigenvalue of $H_{n,k}$ is $\mu_n(H_{n,k}) = \frac{k-1}{2} + \sqrt{\frac{(k-1)^2}{4} + k(n-k)}$.

Now note that $\delta = k$ and $2|E| = kn-k^2-k$ and so we get equality in (12.9). A result by Bell [136] states that $H_{n,k}$ maximizes the degree variance over all $n$-vertex graphs with the same edge count as $H_{n,k}$, where the degree variance is given by $V = \frac{1}{n} \sum_{i=1}^{n} (d_i - 2|E|/n)^2$. Hence graphs that maximize the KS-entropy include the regular graphs and the highly irregular graphs (namely the ones that maximize the degree variance over graphs with fixed vertex and edge counts).

A lower bound on the maximum eigenvalue follows from the Perron-Frobenius theorem. Let $\langle d \rangle$ denote the average degree of the graph and...
Figure 12.2: Adding self-loops to stabilize the adjacency dynamics.

Figure 12.3: Left: Adding a edge in a cycle increases entropy by adding spanning trees. Middle: Adding an edge to a star graph adds two spanning trees. Right: Creating a cycle out of a path adds \( n - 1 \) spanning trees.

Let \( d_{\text{max}} \) denote its largest degree. Then, the following inequality holds [44], [137]:

\[
\max \{\langle d \rangle, \sqrt{d_{\text{max}}} \} \leq \mu_n \leq d_{\text{max}}.
\]

Hence, we can bound the entropy by

\[
\max \{\log(d), 0.5 \log d_{\text{max}} \} \leq H \leq \log d_{\text{max}}.
\]

Therefore increasing the average degree (say, by adding edges to every vertex) or increasing the maximum degree (adding many edges to a single vertex) will drive the entropy to increase from below.

12.4.2 Kolmogorov-Sinai Entropy of Adjacency-driven Networks

The system-theoretic interpretation of the Kolmogorov-Sinai entropy relates to the adjacency dynamics (12.5). Consider the dynamics where each node is driven by its neighbors as 

\[
\dot{x}_i = \sum_{(i,j) \in E} x_j.
\]

In the matrix form, this is exactly the adjacency dynamics: 

\[
\dot{x} = Ax.
\]

Since \( \text{Tr} A = 0 \) this dynamics is unstable as some eigenvalues are non-negative. One way of stabilizing the system is to add a self-loop with positive weights to each node that drives the system to stability, as shown in Figure 12.2. This corresponds to the dynamics 

\[
\dot{x}_i = \sum_{(i,j) \in E} x_j - \delta x_i,
\]

which in matrix form is 

\[
\dot{x} = (A - \delta I)x.
\]

This is equivalent to taking \( B = I \) in the controlled adjacency dynamics (12.5) with a feedback gain of \( u = -\delta x \).

The minimum value that \( \delta \) has to take in order to drive the system to stability is precisely the largest eigenvalue of \( A \). Hence, the system is stable for any \( \epsilon > 0 \) when 

\[
\dot{x} = (A - (\epsilon + \mu_n)I)x = (A - (\epsilon + \mu_n)I)x.
\]

12.5 Examples of Entropies for Simple Graphs

In Table 12.1, we show some examples of the loop entropy for some simple graphs.
In Figure 12.3, we show how adding edges increases the entropy of some example graphs. A cycle of \(n\) nodes has \(n\) spanning trees since removing any one of the \(n\) edges creates a path on \(n\) nodes and hence a spanning tree. Turning a path into a cycle greatly increases entropy as one goes from one spanning tree to \(n\) spanning trees. Clearly, creating larger cycles in a network increases entropy. In the star graph, connecting any two non-center nodes induces a 3-cycle, which increases the number of spanning trees by two. The star graph is already low-entropy, and it is hence robust against edge perturbations in the sense that adding edges will not significantly increase the number of spanning trees, say compared to joining the ends of a path graph.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Spectrum of (L_G)</th>
<th>Entropy (S_G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>({4 \sin^2 \left(\frac{\pi k}{2n}\right)})</td>
<td>(2 \sum_{k=1}^{n-1} \log \left[2 \sin \left(\frac{\pi k}{2n}\right)\right])</td>
</tr>
<tr>
<td>Complete</td>
<td>({n, 0})</td>
<td>((n - 1) \log n)</td>
</tr>
<tr>
<td>Path</td>
<td>({4 \sin^2 \left(\frac{\pi k}{4n}\right)})</td>
<td>(\sum_{k=1}^{n-1} \log \left[2 \sin \left(\frac{\pi k}{4n}\right)\right])</td>
</tr>
<tr>
<td>Star</td>
<td>({0, 1, n})</td>
<td>(\log n)</td>
</tr>
</tbody>
</table>

Assuming an unweighted graph, we can write the Kolmogorov-Sinai entropy as \(H = \log \mu_n\), where \(A\) is the adjacency matrix of the graph. Using results from the summary article by Brouwer et al. [138], we can write the entropy of some graph families, as shown in Table 12.2.

### 12.6 Conclusion

In this chapter, we examined the notion of designing networks that are amenable for control. First, we did this literally – we examined an algorithm, called whiskering, that preserves the controllability of networks as the graph is built up from scratch. We were able to derive some performance bounds on this algorithm using submodularity properties of the controllability gramian, and we also examined a graph-growing optimization setup that generated well-connected graphs.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Spectrum of (A)</th>
<th>Entropy (H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle</td>
<td>({2 \cos \left(\frac{2\pi j}{n}\right)})</td>
<td>(\log 2)</td>
</tr>
<tr>
<td>Complete</td>
<td>({n - 1, -1})</td>
<td>(\log(n - 1))</td>
</tr>
<tr>
<td>Path</td>
<td>({2 \cos \left(\frac{2\pi j}{n+1}\right)})</td>
<td>(\log \left[2 \cos \left(\frac{\pi}{n + 1}\right)\right])</td>
</tr>
<tr>
<td>Star</td>
<td>({-\sqrt{n - 1}, 0, \sqrt{n - 1}})</td>
<td>(\log \sqrt{n - 1})</td>
</tr>
<tr>
<td>Regular</td>
<td>(</td>
<td>\mu(A)</td>
</tr>
<tr>
<td>Complete Bipartite</td>
<td>({-\sqrt{nm}, 0, \sqrt{nm}})</td>
<td>(\log \sqrt{nm})</td>
</tr>
</tbody>
</table>
Our focus then turned to *matrix-weighted graphs*, and notions of performance on those types of consensus models. For leader-follower consensus, we examined how series-parallel networks can be used to efficiently reason about the performance of the system, as well to efficiently re-weight the network to optimize noise rejection.

In order to address the same question on matrix-weighted networks with time scales, we looked at *edge consensus* and the notion of $\mathcal{H}_2$ performance on that setup. We uncovered a separation principle allowing one to generate optimal time scales and optimal edge weights for consensus to achieve an optimal $\mathcal{H}_2$ performance.

Lastly, we examined two notions of network entropy, and discussed their respective system-theoretic interpretations.
Part IV

INTERACTION
INTRODUCTION

Networked dynamic systems arise in many synthetic and natural systems in science and engineering [5]; in particular multi-agent systems offer an interesting control paradigm. Each agent augments the system with an additional (local) computational resource, motivating the concept of distributed controllers & estimators [53], [60], [139], leading to a notion of local control versus global control. The latter seeks to design control laws to guide groups of agents to a desired objective, and the former seeks to design on-board controllers for each agent to facilitate their role in the global control law.

A class of the more widely used local control laws is called consensus, where each agent averages data from their neighbours to compute a parameter related to their objective – for example, heading, position or a formation center [5], [22], [80]. The attractive feature of consensus is how the interconnections between agents – the so-called network or graph topology – affect the performance and agreement characteristics of the algorithm [140], [141]. Graph-theoretic characteristics, such as symmetry, structural balance and graph spectra provide additional insights into the control-theoretic behaviour of consensus [17], [24], [62], [63], [71].

Global controllers for multi-agent systems take many forms [142], eg., potential field approaches [143], smoothed particle hydrodynamics [144] and density control [145]. In [146], [147], density control with only relative measurements of position between agents is considered, and mean-field control is used to tackle multi-agent interactions by considering a mass flow in the large-N limit [148]–[150].

The main interest area of the current paper is networked dynamic systems in which the underlying network is time-varying. Examples of such systems are switching and proximity-based consensus [5], [80], [151] and the Vicsek flocking model [152]. State-dependency refers to networks in which the underlying graph varies based on the state of the nodes. State-dependent networks have been considered in [83], [140], [148], [150], [153]–[155], but few underlying principles for designing controllers to account for this difficult nonlinearity has been proposed. A wide range of real-life networked systems are state-dependent.

To tackle this problem, we propose a twofold extension of the work in [145]. First, we consider state-dependent networked dynamic systems, instead of single-integrator dynamics. Second, we propose a control method for these systems by using a feedback-based density control law that utilizes optimal mass transport (OMT). OMT was considered for linear systems in [156], nonlinear systems in [157] and for density tracking of non-interacting agents in [158]. Our contribution considers OMT for multi-agent systems, in particular ones with state-dependent dynamics.

In the OMT problem, the initial and final densities $\rho_0$ & $\rho_1$ of the agents are specified. The solution to the problem yields a time-dependent density
profile with boundary conditions imposed by $\rho_0, \rho_1$, and a velocity field that together satisfy the continuity equation.

We aim to use this velocity field as a feedforward control input to the state-dependent multi-agent system, coupled with a density control feedback law. Using a modified form of kernel density estimation (KDE) that takes into account the state-dependent dynamics, we will show that the combination of the two control techniques will allow us to propose a physically realizable control strategy for state-dependent networked dynamic systems.

This chapter is organized as follows. The mathematical preliminaries, including notation, optimal mass transport, and density control with the KDE procedure are reviewed in §13.1. The problem statement and chapter contributions are outlined in §13.2. We present the feedforward controller based on OMT in §13.3, and the state-dependent KDE in §13.4. Examples are provided in §13.5, and the chapter is concluded in §13.6.

13.1 Mathematical Preliminaries

13.1.1 Mathematical Notation & Measure Theory

A measure space $(\Sigma, \mathcal{A}, \mu)$ is a triple containing a sample space $\Sigma \subset \mathbb{R}^n$, a $\sigma$-algebra of subsets of $\Sigma$, and a measure $\mu$ that assigns the 'size' $\mu(A) \in \mathbb{R}_+$ to a set $A \in \mathcal{A}$. The Borel $\sigma$-algebra $\mathcal{B}$ is generated from the countable unions, intersections and complements of open subsets of $\mathbb{R}^n$. The Lebesgue measure $\lambda$ assigns to a closed interval $[a, b] \subset \mathbb{R}$ the 'size' $b - a$; this can be extended to $\mathbb{R}^n$ by considering products of measures. A measure $\mu$ is called absolutely continuous with respect to a measure $\nu$ if $\nu(A) = 0 \Rightarrow \mu(A) = 0$ for $A \in \mathcal{A}$; $\mu$ is called Lebesgue absolutely continuous if $\nu = \lambda$. This is denoted $\mu \ll \nu$. If $\mu \ll \nu$, there exists a function $f : \mathbb{R} \rightarrow \mathbb{R}_+$ called a Radon-Nikodym derivative, or density, of $\mu$ with respect to $\nu$. It is denoted $f := \frac{d\mu}{d\nu}$, and satisfies the property that for all $A \in \mathcal{A}$, $\mu(A) = \int_A f \, dv$. Let $\pi(x, y)$ be a joint measure on $X \times Y$. We denote the set of such joint measures $\pi$ by $P(X, Y)$. The marginal $\pi_x$ of $\pi$ on $X$ is defined as the push-forward under the projection map $X$ on $X$: $\pi_x = X_\ast \pi$, where $X(x, y) = x$. Similarly, the marginal $\pi_y$ of $\pi$ on $Y$ is given by $\pi_y = Y_\ast \pi$, where $Y(x, y) = y$. We denote the convolution of two functions $f, g$ as $f \ast g$, or the convolution of a function $f$ and measure $\mu$ as $f \ast \mu$.

13.1.2 Optimal Mass Transport

Informally speaking, the optimal mass transport problem is to find a mapping between two densities that minimizes some cost. Formally speaking, we consider two measures $\mu_0, \mu_1$ on $\mathbb{R}^n$ with equal mass: $\int_{\mathbb{R}^n} d\mu_0 = \int_{\mathbb{R}^n} d\mu_1$.

---

1 If the measures are Lebesgue absolutely continuous, one can equivalently consider densities $\rho_0, \rho_1$. 
The *optimal mass transport* problem \([159], [160]\) is to find a measurable map \(T : \mathbb{R}^n \to \mathbb{R}^n\) taking \(\mu_0\) to \(\mu_1\) via the following optimization problem:

\[
\begin{align*}
\text{minimize} \quad & \int_{\mathbb{R}^n} c(x, T(x)) \, d\mu_0(x) \\
\text{s.t.} \quad & \int_{x \in A} d\mu_1(x) = \int_{T(x) \in A} d\mu_0(x), \quad \forall A \subset \mathbb{R}^n,
\end{align*}
\] (13.1)

where \(c\) is a cost function depending on the initial and transported masses. The constraint in Problem (13.1) means that \(\mu_1\) is the *push-forward* measure of \(\mu_0\) under the map \(T\), in that for each Borel set \(B \in \mathcal{B} := \sigma(\mathbb{R}^n)\), we have that \(\mu_1(B) = \mu_0(T^{-1}(B))\). This is denoted as \(T_*\mu_0 = \mu_1\).

A generalization of Problem (13.1) by Kantorovich is able to pick out the optimal map \(T\), if it exists, for a certain class of costs \(c\) under the assumption of absolute continuity of the measures [161]. Here, we consider a joint distribution \(\pi(x, y)\) on \(\mathbb{R}^n \times \mathbb{R}^n\) and solve for the optimal admissible measure \(\pi\) given some cost \(c(x, y)\).

The set of admissible measures \(\pi\) are those whose marginals are \(\mu_0\) and \(\mu_1\): \(X_t\pi = \mu_0, \ Y_t\pi = \mu_1\). This is equivalent to requiring

\[
\pi(A \times \mathbb{R}^n) = \mu_0(A), \quad \pi(\mathbb{R}^n \times B) = \mu_1(B)
\] (13.2)

for all measurable \(A \subset \mathbb{R}^n\) and \(B \subset \mathbb{R}^n\). The *Kantorovich relaxed optimal mass transport problem* [161] is given by

\[
\begin{align*}
\text{minimize} \quad & \int_{\mathbb{R}^n \times \mathbb{R}^n} c(x, y) \, d\pi(x, y) \\
\text{s.t.} \quad & \pi \in \{\pi \in P(\mathbb{R}^n, \mathbb{R}^n) \text{ s.t. (13.2) holds}\}.
\end{align*}
\] (13.3)

**Proposition 8** ([159], [160]). For quadratic costs \(c(x, y) = \|x - y\|^2\), the support of the optimal joint measure \(\pi^*(x, y)\) of Problem (13.3) is exactly the graph of the optimal map \(T^*(x)\) minimizing Problem (13.1).

For quadratic costs, Benamou and Brenier formulated an equivalent problem in terms of a constrained fluid mechanics model.

**Proposition 9** ([162]). Given Lebesgue absolutely continuous \(\mu_0, \mu_1\) with Radon-Nikodym densities \(\rho_0, \rho_1\) respectively, Problem (13.3) with quadratic costs is equivalent to

\[
\begin{align*}
\inf_{\rho, \nu} \int_{\mathbb{R}^n} & \int_0^1 \frac{1}{2} \|\nu(t, x)\|^2 \rho(t, x) \, dt \, dx \\
\text{s.t.} \quad & \frac{\partial \rho}{\partial t} + \nabla \cdot (\nu \rho) = 0 \\
& \rho(0, x) = \rho_0(x), \ \rho(1, y) = \rho_1(y).
\end{align*}
\] (13.4)

Furthermore, the solution to Problem (13.4) is of the form \(\nu(t, x) = \nabla \varphi(t, x)\), where \(\varphi(t, x)\) is the Lagrange multiplier of the constraints and the solution to the Hamilton-Jacobi equation \(\partial_t \varphi + \frac{1}{2} \|\nabla \varphi\|^2 = 0\).

**Remark 12.** The optimal map \(T^*\) of Problem (13.3) in the case of quadratic costs can be reconstructed from the variable \(\nu(t, x)\) from the solution of Problem (13.4). This formally establishes the equivalence stated in Proposition 9 [162].
Density Control and Kernel Density Estimation

In [145], a feedback control law to drive a group of single-integrator agents to a desired density profile \( \rho_1(x, t) \) was analyzed. The following feedback law is proposed to compute the velocity field as a function of the error in density \( \Phi(x, t) := \rho(x, t) - \rho_1(x) \) as

\[
v(x, t) = -\alpha \frac{\nabla \Phi(x, t)}{\rho(x, t)} .
\]  

(13.5)

Density control of multi-agent systems is impacted by the ability of individual agents to discern the local density profile from measurements of their neighbours. Since the number of agents is finite, the local density profile must be approximated from finitely many samples \( r_i(t) \).

This can be accomplished using kernel density estimation [163], [164]. The kernel density estimate \( ^\rho \rho(t, x) \) at any point \( x \in \mathbb{R}^n \) and time \( t \in \mathbb{R}_+ \) is given by

\[
^\rho \rho(t, x) = \int_{\mathbb{R}^n} \left[ \prod_{k=1}^d \frac{1}{h_k} K \left( \frac{x_k - \xi_k}{h_k} \right) \right] dP_N(t, \xi) .
\]  

(13.6)

Here, \( K : \mathbb{R} \rightarrow \mathbb{R} \) is called the smoothing kernel, \( h_k \) is called the smoothing parameter, and \( dP_N(t, \xi) \) is a sum of Dirac measures at sample points:

\[
dP_N(t, \xi) = \frac{1}{N} \sum_{r(t) \in S(t)} \delta(\xi - r(t)) d\xi .
\]

Since \( \delta(\cdot) \) is the Dirac delta functional, Equation (13.6) can be written as

\[
^\rho \rho(t, x) = \frac{1}{N} \sum_{i=1}^N \left[ \prod_{k=1}^d \frac{1}{h_k} K \left( \frac{x_k - r_i^k}{h_k} \right) \right] .
\]

The control law (13.5) then uses the estimate \( ^\rho \rho(x, t) \) in place of knowledge of the true density \( \rho(x, t) \), where the sample points \( r(t) \) are taken to be the agent states:

\[
v(x, t) = -\alpha \frac{\nabla \left( ^\rho \rho(x, t) - \rho_1(x) \right)}{^\rho \rho(x, t)} .
\]

In [145], Gaussian kernels were used – this induces an all-to-all communication; every agent is able to sample the position every other agent. The control law (13.5) has a convergence guarantee listed in Theorem 6 of [145].
13.2 Problem Statement

In this chapter, we consider state-dependent networked dynamic systems with \( N \) agents on a bounded region \( \mathcal{R} \subseteq \mathbb{R}^n \), where the \( i \)th agent’s state evolves according to the dynamics

\[
\dot{x}_i = f(G(x), x) + Bu_i(x, t), \quad x := (x_1, \ldots, x_n).
\]

A prototypical example of such a system is state-dependent consensus

\[
\dot{x}_i = \sum_{j \neq i} A(x_i, x_j) \cdot (x_i - x_j) + u_i,
\]

where the edge weight \( w_{ij} := A(x_i, x_j) \) changes depending on the state of the agent \( i \) and its neighbour \( j \). In general, one can consider an interaction kernel \( H(x) \) that generates a consensus-like dynamics by convolution with the Dirac measure supported at agent states \([148], [150] \):

\[
\mu_N(x) = \frac{1}{N} \sum_{j=1}^{N} 1_{\{x_j\}}(x) = \frac{1}{N} \sum_{j=1}^{N} \delta(x - x_j).
\]

Using Equation (13.8), we can write a general multi-agent system as

\[
\dot{x}_i = (H \star \mu_N)(x_i) + u_i.
\]

A simple example motivated by robotics is proximity-based edge switching, where \( A(x_i, x_j) = 1 \) if \( \|x_i - x_j\| \leq r \) and 0 otherwise, where \( r \) is some communication radius. This corresponds to an interaction kernel \( H(x) = x1_{\|x\| \leq r}(x) \). One can also examine second-order systems, such as ones of the form

\[
\dot{x}_i = v_i
\]

\[
\dot{v}_i = (H \star \mu_N)(x_i, v_i) + u_i
\]

If we let the interaction kernel be

\[
H(x, v) = a(|x|) v, \quad a \in C^1([0, +\infty)),
\]

then this generates Cucker-Smale-like flocking:

\[
\dot{x}_i = v_i
\]

\[
\dot{v}_i = \frac{1}{N} \sum_{j \neq i} a(|x_j - x_i|)(v_j - v_i) + u_i.
\]

As the number of agents \( N \) grows sufficiently large, one can consider the time-dependent density of agents \( \rho(x, t) \) over a region of the state space. In
Figure 13.1: Block diagram of density control scheme

the context of mean-field control, the formal large-$N$ limit of the dynamics (13.7) produces the \textit{mean field dynamics} [148], [150].

$$\frac{\partial \rho}{\partial t} + \nabla_x \cdot \left[ (\mathcal{P}(\rho(x,t),t) + u) \rho \right] = 0$$

(13.9)

$$\mathcal{P}(\rho(x,t),x) = \int A(x,y)(y-x)\rho(y,t)dy.$$  

We now state the contributions of this chapter, namely the feedforward OMT scheme with kernel density estimation shown in Figure 13.1.

\textit{Contribution 1: Feed-Forward Density Control With Optimal Mass Transport.}

We consider a generalization of the Brenier-Benamou OMT problem (13.4) with the continuity equation constraint replaced by the mean-field dynamics (13.9):

$$\inf_{\rho, \nu} \int \int_0^1 \frac{1}{2} \| u(t,x) \|^2 \rho(t,x) dt dx$$

s.t.  

$$\frac{\partial \rho}{\partial t} + \nabla_x \cdot \left[ (\mathcal{P}(\rho(x,t),x) + \nu) \rho \right] = 0$$

$$\mathcal{P}(\rho(x,t),x) = \int A(x,y)(y-x)\rho(y,t)dy$$

$$\rho(0, x) = \rho_0(x), \; \rho(1, y) = \rho_1(y).$$  

(13.10)

The solution to Problem (13.10) yields two variables with important physical interpretations. The time-varying density $\rho(t,x)$ represents the mass of agents with dynamics (13.7) with inputs $u_i(x,t) := v(x_i,t)$; the velocity field $v(x,t)$ is precisely the input $u_i$ given to agent $i$ at position $x$ at time $t$.

Problem (13.10) assumes that the initial and final masses of agents are distributed in the mean-field limit according to densities $\rho_0$ and $\rho_1$, respectively. When considering finitely many agents, any initial density will take the form of Equation (13.8) - namely, it will be a Dirac measure supported at the agent states, also called the \textit{empirical density}.

Hence, in general, the boundary conditions on the density in Problem (13.10) can be either \textit{deterministic} (in the case of Dirac measures supported at the agent states), or \textit{probabalistic} (in the sense that the initial/final agent states $x_i(0)$ and $x_i(1)$ are randomly distributed according to the densities $\rho_0$ and $\rho_1$). In the latter case, as $N \rightarrow \infty$, the Dirac measure supported at the agent states at time $t$ converges in a formal sense to the density $\rho(\cdot, t)$ [148], [150].

In either case, we consider the velocity field $v(x,t)$ as a feed-forward input to the dynamics (13.7). Since the number of agents is finite, the empirical density at time $t$ will only approximate the density $\rho(x,t)$ from the solution of Problem (13.10).

In a state-dependent networked dynamic system, e.g., Equation (13.7), the existence of an edge indicates some notion of information transfer between agents. Hence, a physical estimation scheme and density control law can only allow \( i \) to sample those agents \( j \) such that \( A(x_i, x_j) \neq 0 \). The second contribution of this chapter is to extend the KDE procedure in [145] by solving a quadratic program for an optimal kernel that takes into account the state-dependent communication constraints.

13.3 Feedback Control of State-Dependent Networked Dynamical Systems

Consider the state-dependent consensus dynamics (13.7). Let \( v_1(x, t) \) denote the velocity field from the solution to Problem (13.10), and let \( v_2(x, t) \) denote the velocity field from the control law (13.5). Our proposed control law is then given by the velocity field (with \( \alpha > 0 \)),

\[
\begin{aligned}
u(x, t) &= \begin{cases} 
\frac{v_1(x, t) - \alpha \nabla (\rho(x, t) - \rho_1(x))}{\rho(x, t)} & 0 \leq t \leq 1 \\
v_2(x, t) - \mathcal{P}(\mu(x, t), x) & t \geq 1
\end{cases}
\end{aligned}
\]

(13.11)

Of course, the switch at \( t = 1 \) is completely arbitrary, and can be altered by changing the time horizon of Problem (13.10).

The main result of this section is the following theorem, an extension of Theorem 6 in [145]. Informally, it states that as the number of agents in the system tends as \( N \to \infty \), the velocities of the agents performing the state-dependent control law (13.11) will vanish asymptotically.

**Theorem 43.** Consider a system of \( N \) agents \( S(t) \) on a bounded region \( \mathcal{R} \subset \mathbb{R}^n \) with individual dynamics given by (13.7) and with control law (13.11). Further suppose that the initial swarm density \( \rho(x, t) \) and target density \( \rho_1(x) \) satisfy the boundary condition \( \nabla \Phi(x, t) = 0 \) on \( \partial \mathcal{R} \). As \( t \to \infty \), for sufficiently large \( N \) the error density \( \Phi(x, t) \) converges to zero: \( \lim_{t \to \infty} \Phi(x, t) = 0 \), for \( x \in \mathcal{R} \) and so \( \dot{\rho}(x, t) \to \rho_1(x) \). Furthermore, the velocities of all agents vanish asymptotically: \( \lim_{t \to \infty} \dot{x}(t) = 0 \), for \( x \in \mathcal{R} \).

**Proof.** First, recall the following theorem about consistency of the estimate \( \hat{\rho}(x, t) \).

**Theorem 44 ([165]).** Consider a kernel density estimation scheme for the target density \( \rho(x) \). Suppose the smoothing parameter \( h \) is chosen as a function of the number of samples \( N \): \( \lim_{N \to \infty} Nh(N) = \infty \). Then, at each point of continuity \( x \) of \( \rho \), the estimator \( \hat{\rho}_N(x) \) is weakly consistent in that for all \( \epsilon > 0 \),

\[
\lim_{N \to \infty} \mathbb{P}(|\hat{\rho}_N(x) - \rho(x)| > \epsilon) = 0.
\]

By Theorem 44, under the assumption on the smoothing parameter \( h \), we have that as \( N \to \infty \), \( \hat{\rho}(t, x) \to \rho(t, x) \) with probability 1 for any finite \( t \).
Consider the following Lyapunov function:

\[ V(t) = \int_{\mathbb{R}} \left( \frac{\rho(x,t)}{\alpha} \right)^2 \dot{x}^T \dot{x} \, dx. \]  

(13.12)

As \( N \to \infty \), the velocity field \( \dot{x} \) approaches the mean-field limit [148]: \( \dot{x} = \mathcal{P}(\mu, t) + u(x, t) \), where \( \mathcal{P}(\mu, x) = \int A(x, y)(y-x)\mu(y, t)dy \), and \( \mu := \mu(y, t) \) is the measure satisfying \( \partial_t + \nabla \cdot (\mathcal{P}(\mu, t) + u(x, t))\mu = 0 \). Under the control law (13.11), it follows that for sufficiently large \( t \), the Lyapunov function (13.12) can be written as

\[ V(t) = \int_{\mathbb{R}} \nabla \Phi(x, t)^T \nabla \Phi(x, t) \, dx. \]

The time derivative of \( V(t) \) is then given by

\[ \dot{V}(t) = \alpha \int_{\mathbb{R}} \dot{\xi}(x, t)^T \Delta \xi(x, t) \, dx, \]

where \( \xi(x, t) := \nabla \Phi(x, t) \). Since \( \nabla \Phi(x, t) = 0 \) on \( \partial \mathbb{R} \), we have that \( \xi(x, t) = 0 \) on \( \partial \mathbb{R} \) which is a Dirichlet boundary condition. It follows that \( \dot{V}(t) < 0 \) since the Dirichlet problem for the Laplace operator has strictly negative eigenvalues [166].

Therefore, by LaSalle’s Invariance Principle, we can conclude that

\[ \lim_{t \to \infty} \nabla \Phi(x, t) = 0, \]

and so \( \lim_{t \to \infty} \Phi(x, t) = \) constant. However, since \( \nabla \Phi(x, t) = 0 \) on \( \partial \mathbb{R} \), the mass \( \int_{\mathbb{R}} \Phi(x, t) \, dx \) is conserved for all \( t > 0 \) (in that \( \int_{\mathbb{R}} \Phi(x, t) \, dx = 0 \) and so we have that \( \int_{\mathbb{R}} \dot{\rho}(x, t) \, dx = \int_{\mathbb{R}} \rho(x, t) \, dx \). Consequently, it follows that \( \lim_{t \to \infty} \Phi(x, t) = 0 \), and hence \( \lim_{t \to \infty} \rho(x, t) = \lim_{t \to \infty} \dot{\rho}(x, t) \) which completes the proof.

### 13.4 Density Estimation for Kernels with Compact Support

Consider a state-dependent consensus dynamics as in Equation (13.7) where \( A(x_i, x_j) \) is a state-dependent edge weight. In order to implement the density control law, each agent must be able to estimate the density of nearby agents to generate the correct velocity field. In this section, we discuss optimal kernels designed to achieve this task that are subject to the state-dependent constraints imposed by \( A(x_i, x_j) \).

The state-dependent constraints in some (informal) sense denote ‘information transfer’ between agents \( i \) and \( j \). If \( A_{ij} = 0 \), then agents \( i \) and \( j \) cannot detect each other, and the KDE procedure should reflect this. To illustrate this notion, consider Figure 13.2.

In 1D kernel density estimation, there are two parameters selected \textit{a priori} that influence the quality of the estimated probability density function, namely the kernel \( K \) and the smoothing parameter \( h \). We consider an optimal selection of \( K \) subject to the state-dependent constraints; we leave the task
of selecting $h$ for future work. For now, we just need the following assumption on $h$ as a function of the number of samples: $\lim_{N \to \infty} Nh(N) = \infty$. The standard metric for measuring the quality of the estimated probability density function is given by the mean integrated square error $E_MISE := \mathbb{E} \left[ \int (\hat{\rho}(x) - \rho(x))^2 \, dx \right]$.

By extracting out the dependence on the number of samples $N$, and the choice of smoothing parameter $h$, one can obtain the asymptotic mean integrated square error (AMISE) 

$$E_MISE := E_{AMISE} + o \left( (hn)^{-1} + h^4 \right).$$

One can factor the AMISE into a product of two terms, one depending on $h$ and one depending on $K$: $E_{AMISE} = C_1(K)C_2(h)$, where

$$C_1(K) := \left[ \left( \int K(x)^2 \, dx \right)^4 \left( \int x^2K(x) \, dx \right)^2 \right]^{1/5}.$$

Hence, by fixing $a := \int x^2K(x) \, dx$ depending on the length of the boundary of our estimation horizon, the only parameter left to optimize is the roughness of $K(x)$: $\int K(x)^2 \, dx$. We can write an optimization problem as follows:

$$\begin{align*}
\text{minimize} & \quad \int K(x)^2 \, dx \\
\text{s.t.} & \quad \int K(x) = 1, \ \int xK(x) = 0 \\
& \quad \int x^2K(x) = a^2 < \infty, \ K(x) \geq 0.
\end{align*}$$

(13.13)

In one dimension, the solution to Problem (13.13) is given by [163],

$$K^a_d(x) = \frac{3}{4} \frac{1}{a\sqrt{5}} \left[ 1 - \left( \frac{x}{a\sqrt{5}} \right)^2 \right] \mathbb{1}_{\{|x| \leq a\sqrt{5}\}}.$$

(13.14)

We wish to find the solution to a modified version of this problem where we enforce a compact support constraint of the form $\{K(x) = 0, \ x \in \mathcal{A}, \ \mathcal{A}^c \text{ compact}\}$. To this end, notice that Problem (13.13) can be numerically solved by discretizing it as follows.

Denote the region of the problem as $X = \{x : |x| \leq B\}$. Discretize $X$ into $N$ points spaced $dx$ apart. Let the vector $x := \{x_i\}_{i=1}^N \in [-B, B]^N$ consist of these points, and let $k := \{k_i\}_{i=1}^N$ be the vector of the kernel $K$ evaluated...
13.4 Density Estimation for Kernels with Compact Support

Figure 13.3: Optimal kernels with unconstrained support, and support constrained to $[-2, 2] \setminus [1/4, 3/4]$, with second moment $a = 5^{1/2}$. The unconstrained kernel solution is exactly given by the Equation (13.14).

at these points, i.e. $k_i = K(x_i)$. Discretizing the integrals yields a quadratic program of the form:

$$\begin{align*}
\text{minimize} & \quad k^T k \\
\text{s.t.} & \quad \sum_{i=1}^N k_i dx = 1, \sum_{i=1}^N x_i k_i = 0 \\
& \quad \sum_{i=1}^N x_i^2 k_i dx = a^2, \quad k_i \geq 0, \quad 1 \leq i \leq N.
\end{align*}$$

As discussed before, an agent’s state-dependent density estimate will depend on sampling points from agents that have an edge between them. Hence, in the density estimate for agent $i$, the kernel $K$ will only depend upon the state of agent $i$ and its neighbours $N_i$. The density estimate of agent $i$ is written as

$$\hat{\rho}_i(t, x(t)) = \frac{1}{N h^d} \sum_{j \in N_i} \left[ \prod_{k=1}^d K_k \left( \frac{x_i(t) - x_j(t)}{h}, x_j \right) \right]$$

where the support of the kernel is restricted to the support of the state-dependent edge weight $A(x_i, x_j)$:

$$A(x_i, x_j) = 0 \implies K(h^{-1}(x_i - x_j), x_j) = 0.$$ 

To extend Problem (13.15) to multi-dimensional systems, we consider multiplicative kernels for $x_i \in \mathbb{R}^n$, where each dimension is estimated independently: $K(x) = \prod_{k=1}^N K_k(x_k)$. This yields the final optimal kernel problem with compact support constraint. For brevity, we show the explicit form for
the 2D problem, as it is clear (yet notationally cumbersome) how to write the general ND problem:

$$\text{minimize } \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} k_{ij}^2$$

$$\begin{align*}
\text{s.t. } & \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} k_{ij} \, dx \, dy = 1, \quad k_{ij} \geq 0, \quad \forall i, j \\
& \sum_{i=1}^{N_x} x_i k_{ij} = 0, \quad \sum_{i=1}^{N_x} x_i^2 k_{ij} \, dx = a^2, \quad \forall j \\
& \sum_{j=1}^{N_y} y_j k_{ij} = 0, \quad \sum_{j=1}^{N_y} y_j^2 k_{ij} \, dx = a^2, \quad \forall i \\
& k_{ij} = 0 \text{ if } (x_i, y_j) \in A, \quad A^c \text{ compact.}
\end{align*}$$

It is important to note that removing a compact interval from the kernel may bias the density estimate - this is unavoidable. The compact support constraint defines a selection-biased distribution (SBD); each agent samples the distribution $d(x) = w(x) \rho(x) / \mu$, with $w(x) = 1_{A^c}(x)$, $\mu = \int w(x) \rho(x) \, dx$. The standard unbiased kernel density estimate of a SBD involves multiplying the kernel by a factor of $\mu / w(x)$ [167], [168], which is unbounded for our choice of $w(x)$. Techniques for unbiasing $\hat{\rho}(x)$ will be left for future work.

### 13.5 Examples

We numerically simulate $N = 200$ agents with interaction kernel $H(x) = x_1 \|x\| \leq 0.01(x)$. The velocity field $P[\mu](x)$ is evaluated with the Dirac measure (13.8), effectively yielding $N$ single-integrator agents that are able to only sample agents a short distance away from each other.

The optimal mass transport problem was solved using open-source code, utilizing a primal-dual algorithm [169], [170]. The density profile and velocity field was calculated over a $100 \times 100 \times 100$ grid in $x, y$ and $t$ space. The initial density $\rho_0(x)$ was a 2D Gaussian at the center of a $[0, 1]^2$ grid, and the target density was a ring of 2D Gaussians, as shown in Figure 13.4. The superimposed optimal density profile over time, and the states of the agents over time integrating the feedback and feedforward control law are shown in Figure 13.5. As one can see, the agents are more organized around the final density distributions when using the feedback law as opposed to just integrating the feedforward law, as shown in Figure 13.6.

### 13.6 Conclusion

In this chapter, we examined density control of state-dependent networked dynamic systems. We utilized the optimal mass transport problem to design
13.6 conclusion

Figure 13.5: Optimal density profiles over time, and superimposed agent states using the feedback density control law.

Figure 13.6: Optimal density profiles over time, and superimposed agent states with only the feedforward control.
a feed-forward velocity field propelling agents with initial conditions sampled from a density profile $\rho_0$ to some target density $\rho_1$. We then tackled the problem of using a density feedback control law with sparse measurements dictated by the state-dependent edge switching constraints of the agents. We utilized kernel density estimation to convert measurements of neighboring agents into a local estimate of the swarm density, which was then used to calculate a feedback density control law. In particular, a quadratic program was designed to find the optimal kernel subject to the state-dependent edge switching.

There are many open problems remaining, here we discuss several. First, the selection of an optimal interaction distance $r = h$ for proximity-based edge switching. This will depend on, for example, $\rho_0$, $\rho_1$ and $N$. If $h$ is large, this will require more on-board computation and sensing capability; if $h$ is small, agents will be isolated. Second, one can consider the task of determining a state-dependent kernel yielding an unbiased estimate of $\rho$. 
Part V

CONCLUSION AND FUTURE WORK
CONCLUDING REMARKS

The purpose of this thesis was to examine various notions of how the underlying network structure affects the behaviour and performance of, and ability to control, networked dynamical systems. To this end, we examined three questions:

1. How does the network affect the control-theoretic properties of the underlying system?

2. How can one design a network \textit{a priori} to achieve a certain performance criterion? Given an existing network, what metrics give us the correct notion of performance?

3. How does one interact with a networked dynamical system?

Each part of this thesis has addressed one of these three questions. In the first part, we examined how different types of interactions affect the controllability of the consensus protocol. We showed that a certain distribution of positive versus negative interactions between agents, coupled with network symmetries, causes the resulting protocol to be uncontrollable. We extended this to the nonlinear regime in the context of the accessibility problem.

The question of identifying communities of agents with friendly interactions was also addressed. Data-driven system identification techniques involving the Koopman operator were used to identify polarized clusters of agents – in each cluster, only positive interactions exist, and between clusters only negative interactions exist.

In the second part of this thesis, we examined the problem of measuring the performance of networks, and designing networks to achieve certain performance criteria. Algorithms for generating controllable and well-connected networks were considered. Next, the performance of matrix-weighted consensus networks were examined in the context of noise rejection and the $H_2$ norm.

In two different contexts, we used the paradigm of series-parallel networks to address efficient $H_2$ norm computation and fast, decentralized network re-weighting to optimize said norm. We considered edge consensus in order to analyze how the $H_2$ norm behaves when considering systems with multiple time scales. In both cases, matrix-weighted networks were considered – these graphs have agents with vector-valued states, and more complex coupling between states.

Lastly, we examined notions of density control of networked dynamical systems. In cases where one allows some uncertainty in the agent states, or perhaps just wants the agent states to occupy some distribution, one can employ the field of \textit{density control} to steer the agents from one distribution to another. We examined notions of feed-forward control (optimal mass transport) as well as feedback control (density control with kernel density estimation) in the case of edge-switching dynamics. This allowed us to design
control laws for systems where the network changes with time as a function of the states of the agents.
In this chapter, we discuss the bright outlook for the future of research in networked dynamical systems, in particular pertaining to the extensions of work done in this dissertation. As in the spirit of this thesis, we outline open problems in the areas of control, design, and interaction with networked dynamical systems.

15.1 Control

Controllability is one of the most important properties of the consensus algorithm, as it allows for single users to influence the overall behaviour of large networked dynamical systems by injecting signals at boundary nodes.

One interesting question is regarding matrix-weighted graphs. Suppose that the state of the $i$th node is $x_i(t) \in \mathbb{R}^k$:

$$x_i(t) = \begin{bmatrix} x_i(t)_1 \\ \vdots \\ x_i(t)_k \end{bmatrix}$$

Then, the matrix weighted consensus protocol on the $i$th node in very explicit form is given by

$$\frac{d}{dt} \begin{bmatrix} x_i(t)_1 \\ \vdots \\ x_i(t)_k \end{bmatrix} = - \sum_{j \in N_i} W_{ij} \begin{bmatrix} x_i(t)_1 \\ \vdots \\ x_i(t)_k \end{bmatrix} - \begin{bmatrix} x_j(t)_1 \\ \vdots \\ x_j(t)_k \end{bmatrix}.$$ 

If we designate a leader node $l$, and suppose that the $i$th node is connected to $l$, then we can write

$$\frac{d}{dt} \begin{bmatrix} x_i(t)_1 \\ \vdots \\ x_i(t)_k \end{bmatrix} = \begin{bmatrix} x_l(t)_1 \\ \vdots \\ x_l(t)_k \end{bmatrix} + \begin{bmatrix} x_j(t)_1 \\ \vdots \\ x_j(t)_k \end{bmatrix} - \sum_{j \in N_i \setminus \{l\}} W_{ij} \begin{bmatrix} x_i(t)_1 \\ \vdots \\ x_i(t)_k \end{bmatrix} - \begin{bmatrix} x_j(t)_1 \\ \vdots \\ x_j(t)_k \end{bmatrix}.$$ 

Here, the state of the leader node $x_l$ is identified as our control input. A natural question is can we steer the ath state of node $i$ by driving the bth state of $i$ via $l$?
In other words, if we view the matrix-weighted graph as a network of $k$ layers, then can we exert control over the $a$th layer by injecting a signal into the $b$th layer?

Another natural question to ask is the extension of signed networks into higher-order interactions. For example, a colour-charge type interaction where each node is assigned one of $C$ colours, and nodes of the same colour interact cooperatively, or antagonistically otherwise. Formally, let $\chi : \mathcal{N} \rightarrow \{C_1, \ldots, C_n\}$ be a colouring function. Then, for every edge $\{i,j\} \in \mathcal{E}$,

$$W_{\{i,j\}} = \begin{cases} 1 & \chi(i) = \chi(j) \\ -1 & \chi(i) \neq \chi(j). \end{cases}$$

This would be one way to generalize the bipartition corresponding to structural balance into a multipartite system.

Similarly, we can consider complex phase interactions, where the states of each node are now complex-valued, and the colouring function separates each node into a characteristic phase

$$\chi_P(j) = e^{\chi(j)2\pi i/(N+1)}.$$ 

Then, the weight between two neighbours is given by

$$W_{\{l,m\}} = e^{\chi(l)2\pi i/(N+1)}e^{\chi(m)2\pi i/(N+1)}.$$ 

When the number of colours is two, this reduces down to the case of signed networks.

Another question is on the modeling usage of matrix-weighted graphs. The relevant chapters in this thesis discussed some applications from the literature on how matrix-weighted consensus protocols are used in practice, but it would be interesting to see if some interesting networked dynamical systems can be modelled by them, specifically oscillator dynamics. By extending the matrix weights to Hermitian matrices, and the node states to complex valued, one can view the matrix as a coupling between amplitude and phase over a network of oscillators. What specific oscillator models can be captured in such a setting is an open problem.

15.2 design

The network design problems in this paper were primarily focused on edge weight and time scale design for noise rejection in controlled consensus networks. A natural extension would be to consider similar design problems for the dual task of estimation. In particular, edge weight and time scale design has the potential to be useful for designing Kalman consensus filters over large sensor networks. Seminal work by Olfati-Saber has provided an angle for approaching this sort of design problem [38], [39], [41].
As seen in an earlier chapter of this dissertation, agents over a sensor network can achieve consensus on a state estimate of that system by adding a consensus step to their estimate propagation:

\[
\hat{x}_i^{k+1} = \hat{x}_i^k + K_i^k \left( \hat{y}_i^k - C_i \hat{x}_i^k \right)
\]

\[
\hat{x}_i^{k+1} = \hat{x}_i^k + K_i^k \left( \hat{y}_i^k - C_i \hat{x}_i^k \right) + \sum_{j \in N_i} (\hat{x}_j^k - \hat{x}_i^k).
\]

What has not been explored is a weight assignment or weight update scheme of the form

\[
e_i^k \hat{x}_i^{k+1} = \hat{x}_i^k + K_i^k \left( \hat{y}_i^k - C_i \hat{x}_i^k \right) + \sum_{j \in N_i} W_{ij} (\hat{x}_j^k - \hat{x}_i^k),
\]

where the weights \(W_{ij}\)'s are chosen according to a gradient weight update on the minimization of an appropriate cost, perhaps related to the steady-state error covariance of the estimate, or a stability-based criterion, and similarly with the timescales \(e_i^k\).

Work by Barooah & Hespanha has shown that the electrical networks interpretation considered in the weight design section of this thesis has an interpretation in certain estimation tasks. In particular, the state estimate covariance of a certain type of (non-recursive) estimator is related to the matrix-valued effective resistance over a matrix-valued graph representing a sensor network \([94], [115], [128]\). If an analogous relation could be derived for a Kalman consensus filter setup, then it is very likely that a distributed series-parallel algorithm for adaptive weight design could be implemented in such a setting.

15.3 Interaction

Optimal transport has received a large amount of attention from both the mathematical and control theory communities. Two fields medalists in recent years, Cédric Villani and Alessio Figalli, have made substantial contributions to the field. The work in this dissertation has focused on applications of optimal transport theory in controlling distributions of agent states in networked dynamical systems, but there is a plethora of other questions one can pose that extends the results presented here.

One fundamental problem with the optimal transport approach, in particular when coupled with the gradient feedback, is the assumption of the mean-field approximation. In some formal sense, the continuity equation

\[
\frac{\partial}{\partial t} \rho(x, t) + \nabla_x \cdot \left( (\mathcal{P}(\rho, x) + \nu(x, t)) \rho(x, t) \right) = 0
\]
arises from a *mean-field limit* where we take the number of agents \( n \rightarrow \infty \), yielding some notion of the following convergence of the empirical measure supported at agent states:

\[
\lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i(t)) = \rho(x, t).
\]

Of course, real systems do not have infinitely many agents, and so the performance of the methods of the work presented in this dissertation will be affected by how many (or rather, how few) agents are in the system. The natural question to ask is what is the right “error metric” to use when comparing the vanilla continuum to the discrete counterpart. A naïve choice may be, for example

\[
\varepsilon(x, t) = \left\| n^{-1} \sum_{i=1}^{n} \delta(x - x_i(t)) - \rho(x, t) \right\|.
\]

It would be of significant interest to identify what sort of “error metric” gives nice control-theoretic guarantees on the performance of these methods when applied to finitely many agents.

An alternate approach would be to explicitly discretize the optimal transport problem in such a way that the discretized density can be explicitly represented as the empirical measure supported at agent states. The issue with this approach is that then the discretized density is essentially a time-evolving grid when solving the continuity equation, and so care must be taken to make the solution nicely behaved.

When implementing these techniques on formation control of aerial or robotic agents, the task of collision avoidance becomes paramount. A natural question to ask is what conditions on \( v(x, t) \) can be imposed that provide some bound on how two single-integrator particles solving the ODE

\[
\dot{x}_1 = v(x_1(t), t), \quad \dot{x}_2 = v(x_2(t), t)
\]

will not come within some radius \( r \) from each other. For example, if \( \|x_1(0) - x_2(0)\| \geq r \), what conditions on \( v(x, t) \) will ensure that \( \|x_1(t) - x_2(t)\| \geq r \) for all \( 0 \leq t \leq T \)? Can these conditions be implemented in a convex optimization program compatible with optimal transport-type programs?

Lastly, hardware implementation of these methods will serve to both verify the mass transport techniques as well as expose the shortcomings and limitations of them. For example, actuator saturation and hardware speed limits will impair the ability of vehicles to properly track the velocity fields prescribed by the optimal transport and gradient feedback. Imposing constraints that solve these issues on the optimal transport optimization problem is certainly a topic of interest.

Another pressing issue related to hardware implementation is computational efficiency. Currently, the optimal transport control has to be solved off-line, and in a rather computationally expensive way. The algorithm by [170] solves the vanilla Brenier-Benamou mass transport problem by changing to flux coordinates \( F(x, t) = v(x, t)\rho(x, t) \) and then discretizing the problem
space in \((x, t)\). The discretization of the Brenier-Benamou problem yields a cost function of the form

\[
\min_{F, \rho} \sum_{x, t} \frac{\|F(x, t)\|^2}{\rho(x, t)},
\]

which is miraculously convex, as well as linear constraints arising from the discretization of

\[
\frac{\partial}{\partial t} \rho(x, t) + \nabla_x \cdot F(x(t), t) = 0,
\]

which yields a set of linear equality constraints. Unfortunately, the large solution space resulting from this discretization makes it computationally expensive. Research in online algorithms, or approximate solutions that are cheap, would be of significant interest.
BIBLIOGRAPHY


BIBLIOGRAPHY


COLOPHON

This document was typeset using the classicthesis template developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography "The Elements of Typographic Style". The fonts used are Linux Libertine for prose, and newtxmath for mathematics.

Final Version as of June 20, 2019.