SOME GRAPH THEORETIC METHODS FOR DISTRIBUTED CONTROL OF COMMUNICATING AGENT NETWORKS

By

KRISTIN HERLUGSON

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

Master of Science in Electrical Engineering

at

WASHINGTON STATE UNIVERSITY
Department of Electrical Engineering and Computer Science

December 2004

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of KRISTIN HERLUGSON find it satisfactory and recommend that it be accepted.

_____

Co-Chair

_____

Co-Chair

_____

_____

Acknowledgements

I would first like to acknowledge my two amazing advisors, Ali Saberi and Sandip Roy. Dr. Saberi introduced me to an entire world of control theory I did not know existed. He guided me with patience and compassion that I will always value. Sandip is a natural teacher that not only knew just how to teach me, but also how to guide my work so as to drive the passions that brought me to graduate school. Both Dr. Saberi and Sandip have amazing work ethic that pushed me through my time at WSU but they also forced me to explore my interests outside of my research.

I would like to thank the remaining members of my committee, Thomas Fischer and Bernie Lesieutre, for their time and valuable feedback.

I have had two important academic mentors during my time in New Mexico that have contributed greatly to my success in graduate school. Scott Teare of New Mexico Tech told me every day of my senior year that I was going to graduate school, no matter how much I resisted. Rush Robinett at Sandia National Laboratories taught me there is nothing more valuable than following your passions in both your work and your life. Rush sparked my first interests in controls and in controls research and continues to be an amazing life coach and an even more amazing friend.

My work was funded by the Office of Naval Research and generous feedback was provided by researchers at the University of Idaho.

I have several friends that supported me through undergraduate and graduate school that I could not have been without. My Masters degree would not have been anywhere near as rewarding and exciting without Jasmine. Thank you for teaching me how to attack a problem with patience and clarity, and for being a constant source of stimulating conversation. Thank you to Jeremiah and to Andrea for reminding me what is really important in my life and keeping me laughing through even the most difficult times. I have an amazing family that encouraged me to be an engineer in a family full of biologists and students of business.

And finally, thank you to Andreas for encouraging and supporting me to become the engineer, and the person, I want to be. Thank you for everything.

SOME GRAPH THEORETIC METHODS FOR DISTRIBUTED CONTROL OF COMMUNICATING AGENT NETWORKS

Abstract

by Kristin Herlugson, M.S.
Washington State University
December 2004

Co-Chairs: Ali Saberi and Sandip Roy

Our work is motivated by the increasing application for fleets of autonomous agents; specifically the design of local feedback laws for global action. First, we consider formation and alignment of distributed sensing agents with double-integrator dynamics and saturating actuators. We explore the role of the agents sensing architecture on their ability to complete formation and alignment tasks. We also consider design of static controllers for the network of agents, and find that static control is indeed possible for a large class of sensing architectures. Second, we present a control-theoretic perspective on the design of distributed agreement protocols. We explore agreement-protocol analysis and design for a network of agents with single-integrator dynamics and arbitrary linear observations. We explore agreement in a quasi-linear model with a stochastic protocol, which we call the controlled voter model. Finally, we present a stochastic protocol for decision-making or agreement for a network of sensing agents subject to communication faults. Throughout, several examples are developed, to motivate our formulation and illustrate our results.

Contents

List of Figures

Dedication

To my parents, Chris and Mary Lou,

to my sisters, Erika and Brita

and to Andreas

Chapter 1

INTRODUCTION

Embedded systems are playing a much larger role in our lives now as our cars, computers, and houses become increasingly more sophisticated. As the embedded systems become more advanced, the need arises for network systems to perform more complicated tasks. The increase in network complexity has generated renewed interest in distributed control of networked agents. It is known that the coordination of several systems outperforms the work of a single system. While there are several important applications of networked control systems, there is one underlying ambition; local action to achieve a global goal. How can we design controllers for the individuals in the network so that the network can achieve a global goal?

Our work, which was sponsored by the Office of Naval Research (grant number $N000140310848$), is to design a communication structure for a network of underwater unmanned autonomous vehicles (UAVs). As technology advances, engineers search for ways to increase human safety in dangerous fields. A major appeal of UAVs is their potential for performing repetitive, dangerous, and information gathering tasks in hazardous environments. While we are primarily interested in UAV control, other applications of our work include distrib-

uted sensor network control, automated air and highway traffic control, analysis of arrays of micro-devices, and automated factory control.

We are interested in designing controllers to achieve three global tasks: formation (the settling of agents to specific locations or fixed-velocity trajectories), alignment (the partial stabilization of the agents dynamics), and agreement (the convergence of each agent's state to the same fair value). The agents we are interested in controlling are coupled by their task rather than dynamically coupled, so the key design aspect of the network becomes the communication structure. Due to limited bandwidth and small communication radius, and in some cases limited memory, each agent cannot observe and store information about every other agent in the network. In this thesis, we study what communication structures allow for control to achieve the network's global goal.

As each agent would receive information, say the position and velocity of some neighboring agents relative to its own, it is natural to model this communication structure as a graph, as introduced in [1]. We consider, as in [1], systems in which the agents establishing a formation, or alignment, or reaching agreement are coupled through a task rather than dynamically coupled. This characteristic of cooperative vehicle control causes the interconnection structure of the network to be the key to network stability rather than the individual agent dynamics. We consider networks with a fixed topology; the agents are always in communication with the same set of neighbors. While Fax and Murray require that each agent has the same controller, our work allows for each agent to have a different static controller. Previously, the focus of decentralized control research was on the dynamics

of the individual agents; the graph-theoretic approach treats the agents as simple systems and directly analyzes the interplay between the network's communication structure and its task dynamics.

In addition to the contributions by Fax and Murray to distributed control, the following papers have provided useful results and modeling techniques in our controller design. In [8], Wang and Davison present influential results in the area of decentralized control system stabilization. The authors consider stabilization of a linear time-invariant decentralized system with local feedback control laws and develop the notion of fixed modes which is used extensively in our work. In the final chapter of this thesis, we present a first attempt at a control theory for distributed decision-making in a network in which the opinion of each agent is discrete-valued. We use the influence model developed in [17, 18] to describe our network and, from there, develop a stochastic agreement protocol for the network. The influence model is advantageous in that its structure allows for significant characterization of the asymptotics of the global dynamics of the system from low-order recursions.

This thesis is a collection of papers either accepted or submitted for publication as summarized here.

1. S. Roy, A. Saberi, and K. Herlugson, "Formation and alignment of distributed sensing agents with double-integrator dynamics and actuator saturation," accepted for publication in an IEEE Press monograph entitled Sensor Network Applications, Sep. 2004.

2. S. Roy, A. Saberi, and K. Herlugson, "A control-theoretic perspective on the design of distributed agreement protocols," submitted to Automatica, Jul. 2004.

3. S. Roy, K. Herlugson, and A. Saberi, "A control-theoretic perspective on distributed discrete-valued decision-making in networks of sensing agents," submitted to IEEE Transactions on Mobile Computing, Dec. 2004.

Each chapter is a submitted or published paper, with Chapter 2 being a slightly extended version of the published paper. The remainder of this thesis is organized as follows: in Chapters 2 and 3 we present approaches to controller design for formation and alignment of a network of autonomous agents and agreement protocols, respectively; and in Chapter 4 we consider the case in which communication faults are present as the network tries to reach agreement. Throughout each chapter, illustrative examples and simulations are discussed as well as future directions of study.

Chapter 2

FORMATION AND ALIGNMENT OF DISTRIBUTED SENSING AGENTS WITH DOUBLE-INTEGRATOR DYNAMICS AND ACTUATOR SATURATION

In this article, we consider formation and alignment of distributed sensing agents with double-integrator dynamics and saturating actuators. First, we explore the role of the agents' sensing architecture on their ability to complete formation and alignment tasks. We develop necessary and sufficient conditions on the sensing architecture, for completion of formation and alignment tasks using linear dynamic control. We also consider design of static controllers for the network of agents, and find that static control is indeed possible for a large class of sensing architectures. Next, we extend the control strategies developed for completion of formation tasks to simultaneously achieve collision avoidance. In particular, we consider formation stabilization with collision avoidance for sensing agents that move in the plane. The control paradigm that we develop achieves avoidance and formation

together, by taking advantage of the multiple directions of motion available to each agent. Our explorations show that collision avoidance can be guaranteed, given some weak constraints on the desired formation and the distance that must be maintained between the agents. Throughout, several examples are developed, to motivate our formulation and illustrate our results.

Keywords: formation, alignment, sensing architecture, distributed control, collision avoidance, stabilization.

2.1 Introduction

A variety of natural and engineered systems comprise networks of communicating agents that seek to perform a task together. In such systems, individual agents have access to partial information about the system's state, from which they attempt to actuate their own dynamics so that the system globally performs the required task. Recently, much effort has been given to developing plausible models for systems of interacting agents and to constructing decentralized controllers for such systems (e.g., [3, 1, 4, 2, 5]). These studies vary widely, in the tasks completed by the agents (including formation stabilization and collision avoidance), the intrinsic dynamics and actuation of the agents, the communication protocol among the agents, and the structure of the controllers.

Our research efforts are focused on understanding, in as general a manner as possible, the

role of the communication/sensing network structure in allowing the network to perform the required task. In this first study, we consider systems with simple but plausible local dynamics (double-integrator dynamics with saturating actuators) and task aims (settling of agents to specific locations or fixed-velocity trajectories in a Euclidean space without collision avoidance, henceforth called **formation stabilization**). Within this simple context, we assume a quite general sensing network architecture[1], and specify necessary and sufficient conditions on this architecture for the existence of a decentralized dynamic LTI controller that achieves formation stabilization. Using our formulation, we are also able to identify a broad class of sensing architectures for which static decentralized control is possible. While the agent dynamics considered here are limited, we believe that our approach is promising because it clearly extracts the role of the sensing architecture in completing tasks and hence facilitates development of both appropriate sensing architectures and controllers for them. Further, we are able to extend our control design to achieve collision avoidance in addition to stabilization, for agents defined in the plane.

The goals of our analysis are clearly illustrated with an example. Let's say that three coordinating vehicles seek to locate themselves to the West, East, and South of a target. We aim to achieve this task by controlling the accelerations of the vehicles. Our studies aim to determine the class of observation topologies (ways in which the vehicles observe the target location and/or each others' locations) for which the formation stabilization task can be achieved, without collision among the vehicles.

[1]We feel that our observation architecture is more accurately viewed as a sensing architecture rather than a communication architecture, because measurements are assumed to be instantaneous; hence, we will use the term sensing architecture, though our formulation may quite possibly provide good representation for certain communication architectures also.

Throughout our studies, we aim to delineate the connections between our formulation and results, and those found in the existing literature on vehicle task dynamics. Broadly, our key contributions to this literature are as follows:

- Our studies consider an arbitrary linear observation topology for the sensing architecture, that significantly generalizes the sensing architectures that we have seen in the literature. Of particular interest is the consideration of multiple observations for each agent; we find that multiple observations can sometimes permit stabilization even when a single observation that is an average of these observations does not.

- We consider actuator saturation, which we believe to be realistic in many systems of interest.

- We are able to develop explicit necessary and sufficient conditions on the sensing architecture for formation stabilization. This analysis also serves to highlight that the seminal research on decentralized control done by Wang and Davison [8] is central in the study of distributed task dynamics. From this viewpoint, our work buttresses the analysis of [3], by extending the application of [8] to sensing architectures beyond leader-follower ones.

- We show that static stabilizers can be designed for a wide class of sensing architectures, and we explore system performance upon static stabilization through simulations.

2.2 Model Formulation

In this section, we describe the model of distributed, mobile sensing agents that is studied throughout this article. The model is formulated by first specifying the local dynamics of each agent and then developing a sensing architecture for the agents. A vector representation for the model is also presented.

2.2.1 Local Dynamics

Each of the n agents in our system is modeled as moving in a 1-dimensional Euclidean space. We denote the position of agent i by $r_i \in \mathbf{R}$. The position of agent i is governed by the differential equation $\ddot{r}_i = \sigma(u_i)$, where $u_i \in \mathbf{R}$ is a decentralized control input and $\sigma()$ represents (without loss of generality) the standard saturation function. We also sometimes consider double-integrator dynamics without saturation, so that $\ddot{r}_i = u_i$.

One note about our model is of particular importance: in our simulations, we envision each agent as moving in a multi-dimensional Euclidean space, yet agents in our model are defined as having scalar positions. We can do so without loss of generality because the internal model for the agents in each coordinate direction is decoupled (in particular, a double-integrator). Hence, we can simply redefine each agent in a multi-dimensional system as a set of agents with scalar positions, each of which track the location of the original agent in one coordinate direction. We will discuss shortly how observations in a multi-dimensional model can be captured using a scalar reformulation.

2.2.2 Sensing Architecture

We define the **sensing architecture** for the system quite generally: each agent has available one or more linear observations of the positions and velocities of selected agents. Formally, we denote the number of linear observations available to agent i by m_i. The $m_i \times n$ **graph matrix**

$$G_i \triangleq \begin{bmatrix} g_{11}(i) & \dots & g_{1n}(i) \\ \vdots & & \vdots \\ g_{m_i 1}(i) & \dots & g_{m_i n}(i) \end{bmatrix}$$

specifies the linear observations that are available to agent i. In particular, the jth ($1 \leq j \leq m_i$) observation available to agent i is the average

$$\mathbf{a}_{ij} = g_{j1}(i) \begin{bmatrix} r_1 \\ v_1 \end{bmatrix} + \ldots + g_{jn}(i) \begin{bmatrix} r_n \\ v_n \end{bmatrix}, \tag{2.1}$$

where $v_i = \dot{r}_i$ is the velocity of agent i. Agent i's m_i observations can be concatenated into a single observation vector:

$$\mathbf{a}_i^T \triangleq \begin{bmatrix} \mathbf{a}_{i1}^T & \vdots & \mathbf{a}_{im_i}^T \end{bmatrix}$$

In vector form, the observation vector for agent i can be written in terms of the state vector as

$$\mathbf{a}_i = C_i \mathbf{x}, \tag{2.2}$$

where

$$C_i = \begin{bmatrix} G_i & 0 \\ 0 & G_i \end{bmatrix}. \tag{2.3}$$

Sometimes, we find it convenient to append the graph matrices for individual agents into a single matrix. We define the **full graph matrix** for the agents as $G^T = \begin{bmatrix} G_1^T & \dots & G_n^T \end{bmatrix}$.

A couple notes about our sensing architecture are worthwhile:

- We can represent the graph-Laplacian sensing architecture described in, e.g., [1]. Graph-Laplacian observation topologies are applicable when agents know their positions relative to other agents. More specifically, each agent i's observation is assumed to be a average of differences between i's position and other agents' positions. To capture Laplacian observations using our sensing architecture, we constrain each agent to have available a single average (i.e., $m_i = 1$ for all i), and specify the graph matrix entries for agent i as follows:

$$\begin{aligned} g_{1i}(i) &= 1 \\ g_{1j}(i) &= -\frac{1}{|\mathcal{N}_i|}, \quad j \in \mathcal{N}_i \\ g_{1j}(i) &= 0, \quad \text{otherwise}, \end{aligned} \tag{2.4}$$

  where $\mathcal{N}_i$ are the neighbors of agent i and $|\mathcal{N}_i|$ are the number of neighbors of agent i (see [1] for details). Note that the full graph matrix for a Laplacian architecture is square, has unity entries on the diagonals, has negative off-diagonal entries, and has

row sums of 0.

When we consider Laplacian observation topologies, we will often use a **grounded Laplacian** to represent the sensing architecture. A grounded Laplacian represents a sensing architecture in which the agents associated with each connected component of the full graph matrix have available at least one absolute position measurement of some sort. Mathematically, the full graph matrix has unity diagonal entries and negative off-diagonal entries, but each connected component of the full graph matrix is assumed to have at least one row that sums to a strictly positive value. The difference between a grounded Laplacian architecture and a Laplacian architecture is that each agent's absolute position can be deduced from the observations for the grounded Laplacian architecture, but not for the Laplacian architecture. In most applications, it is realistic that absolute positions can be deduced in the frame-of-reference of interest. In, e.g., [1], some systems with a Laplacian architecture are shown to converge in a relative frame, which can equivalently be viewed as absolute convergence of state vector differences given a grounded Laplacian architecture. Here, we will explicitly distinguish between these two viewpoints by considering absolute and partial stabilization of our systems.

Our sensing architecture is more general than the graph-Laplacian architecture, in that arbitrary combinations of agents' states can be observed, and multiple observations are possible. Consideration of multiple observations is especially important, in that it allows comparison of controllers that use averaged measurements with those that use multiple separate measurements. Our analyses show that stabilization is

sometimes possible when multiple observations are used, even though it might not be possible when an average of these observations is used.

- When an agent with a vector (multi-dimensional) position is reformulated as a set of agents with scalar positions, each of these new agents must be viewed as having access to the same information as the original agent. Hence, the graph matrices for these newly-defined agents are identical. We note that this formulation allows observations that are arbitrary linear combinations of state variables associated with different coordinate directions.

- Notice that we have structured the model so that the observation architecture is identical for position and velocity measurements (i.e., whenever a particular position average is available, the same velocity average is also available). The stabilization results that we present in the next section do not require identical observation architectures for positions and velocities: in fact, only the sensing architecture for positions is needed to verify stabilization. However, because static stabilization of the model is simplified, we adopt this assumption (which we believe to be quite reasonable in many applications). In some of our results, we will wish to distinguish that only the position observation structure is relevant. For such results, we shall use the term **position sensing architecture** to refer to the fact that the graph structure applies to only positions, while velocity observations may be arbitrary (or nonexistent).

The types of communication topologies that can be captured in our formulation are best illustrated and motivated via several examples.

Example: Vehicle Coordination, String First, let's return to the vehicle formation example discussed in the introduction. Let's assume that the vehicles move in the plane, and (without loss of generality) that the target is located at the origin. A reasonable assumption is that one vehicle knows its position relative to the target, and hence knows its own position. Assuming a string topology, the second vehicle knows its position relative to the first vehicle, and the third vehicle knows its position relative to the second vehicle. To formulate the graph matrices for this example, we define agents to represent the x and y positions of each vehicle. The agents are labeled $1x$, $1y$, $2x$, $2y$, $3x$, and $3y$. The graph matrices for the six agents are as follows:

$$
\begin{aligned}
G_{1x} = G_{1y} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
G_{2x} = G_{2y} &= \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix} \\
G_{3x} = G_{3y} &= \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{2.5}
$$

Notice that the sensing architecture for this example is a grounded Laplacian architecture.

Example: Vehicle Coordination Using an Intermediary Again consider a set of three vehicles in the plane that are seeking to reach a target at the origin. Vehicle 1 knows the x-coordinate of the target, and hence effectively knows its own position in the x direction. Vehicle 2 knows the y-coordinate of the target, and hence effectively knows its own

position in y direction. Both the vehicles 1 and 2 know their position relative to the intermediary vehicle 3, and Vehicle 3 knows its position relative to Vehicles 1 and 2. We would like to determine whether or not all three vehicles can be driven to the target.

We can use the following graph matrices to capture the sensing topology described above:

$$G_{1x} = G_{1y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G_{2x} = G_{2y} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}$$

$$G_{3x} = G_{3y} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}. \tag{2.6}$$

Example: Measurement Failures Three aircraft flying along a (straight-line) route are attempting to adhere to a pre-set fixed-velocity schedule. Normally, each aircraft can measure its own position and velocity and so can converge to its scheduled flight plan. Unfortunately, because of a measurement failure on one of the aircraft, the measurement topology on a particular day is as follows. Aircraft 1 can measure its own position and velocity, as well as its position and velocity relative to Aircraft 2 (perhaps through visual

inspection). Aircraft 3 can measure its own position and velocity. Aircraft 2's measurement devices have failed. However, it receives a measurement of Aircraft 3's position. Can the three aircraft stabilize to their scheduled flight plans? What if Aircraft 2 instead receives a measurement of the position of Aircraft 1?

We assume that the aircraft are well-modeled as double integrators. Since each aircraft seeks to converge to a fixed-velocity trajectory, this problem is a formulation-stabilization one. We can again specify the graph matrices for the three aircraft from the description of the sensing topology:

$$\begin{aligned} G_1 &= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \end{bmatrix} \\ G_2 &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \\ G_3 &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{2.7}$$

If Aircraft 2 instead receives the location of Aircraft 1, then $G_2 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$.

Example: Vehicle Coordination, Leader-Follower Architecture As in the string of vehicles example, we assume that the vehicles move in the plane and seek a target at the origin. However, we assume a leader-follower sensing architecture among the agents, as described in [3]. In particular, Vehicle 1 knows its position relative to the target, and hence knows its own position. Vehicles 2 and 3 know their relative positions to Vehicle 1. The following graph matrices can be used for this sensing topology:

$$G_{1x} = G_{1y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$G_{2x} = G_{2y} = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$G_{3x} = G_{3y} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

2.2.3 Vector Representation

In state-space form, the dynamics of agent i can be written as

$$\begin{bmatrix} \dot{r_i} \\ \dot{v_i} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \sigma(u_i), \tag{2.9}$$

where $v_i \triangleq \dot{r}_i$ represents the velocity of agent i. It is useful to assemble the dynamics of the n agents into a single state equation:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} \sigma(\mathbf{u}), \tag{2.10}$$

where

$$\mathbf{x}^T = (\begin{bmatrix} r_1 & \dots & r_n & | & v_1 & \dots & v_n \end{bmatrix})^T$$

and

$$\sigma(\mathbf{u}^T) = (\begin{bmatrix} \sigma(\mathbf{u}_1^T) & \dots & \sigma(\mathbf{u}_n^T) \end{bmatrix}).$$

We also find it useful to define a **position vector** $\mathbf{r}^T = (\begin{bmatrix} r_1 & \dots & r_n \end{bmatrix})^T$ and a **velocity vector** $\mathbf{v} = \dot{\mathbf{r}}$.

We refer to the system with state dynamics given by (2.10) and observations given by (2.2) as a **double-integrator network with actuator saturation**. We shall also sometimes refer to an analogous system that is not subject to actuator saturation as a **linear double-integrator network**. If the observation topology is generalized so that the graph structure applies to only the position measurements, we shall refer to the system as a **position-sensing double-integrator networks**. (with or without actuator saturation). We shall generally refer to such systems as **double-integrator networks**.

2.3 Formation Stabilization

Our aim is to find conditions on the sensing architecture of a double-integrator network, such that the agents in the system can perform a task. The necessary and sufficient conditions that we develop below represent a first analysis of the role of the sensing architecture on the achievement of task dynamics; in this first analysis, we restrict ourselves to formation tasks, namely those in which agents converge to specific positions or to fixed-velocity trajectories. Our results are promising because they clearly delineate the structure of the sensing architecture required for stabilization.

We begin our discussion with a formal definition of formation stabilization.

Definition 1 *A double-integrator network can be (semi-globally[2]) formation stabilized to* $(\mathbf{r}_0, \mathbf{v}_0)$ *if a proper linear time-invariant dynamic controller can be constructed for it, so that the velocity* $\dot{r}_i$ *is (semi-globally) globally asymptotically convergent to* v_{i0} *and the relative position* $r_i - v_{i0}t$ *is (semi-globally) globally asymptotically convergent to* r_{i0} *for each agent* i*.*

Our definition for formation stabilization is structured to allow for arbitrary fixed-velocity motion and position offset in the asymptote. For the purpose of analysis, it is helpful for us to reformulate the formation stabilization problem in a relative frame, in which all velocities and position offsets converge to the origin. The following theorem achieves this reformulation:

Theorem 2.1 *A double-integrator network can be formation stabilized to* $(\mathbf{r}_0, \mathbf{v}_0)$ *if and only if it can be formation stabilized to* $(\mathbf{0}, \mathbf{0})$*.*

Proof. Assume that network can be formation stabilized to $(\mathbf{0}, \mathbf{0})$. Then for every initial position vector and velocity vector, there exists a control signal $\mathbf{u}$ such that the agents converge to the origin. Now let's design a controller that formation stabilizes the network to $(\mathbf{r}_0, \mathbf{v}_0)$. To do so, we can apply the control that formation stabilizes the system to the origin when the initial conditions are computed relative to $\mathbf{r}_0$ and $\mathbf{v}_0$. It is easy to check that the relative position offsets and velocities satisfy the initial differential equation, so that the control input achieves the desired formation. The only remaining detail is to verify that the

[2] *We define semi-global to mean that the initial conditions are located in any* a priori *defined finite set.*

control input can still be found from the observations using an LTI controller. It is easy to check that the same controller can be used, albeit with an external input that is in general time-varying.

The argument can be reversed to prove that the condition is necessary and sufficient. □

We are now ready to develop the fundamental necessary and sufficient conditions relating the sensing architecture to formation stabilizability. These conditions are developed by applying decentralized stabilization results for linear systems ([8]) and for systems with saturating actuators ([6]).

Theorem 2.2 *A linear double-integrator network is formation stabilizable to any formation using a proper dynamic linear time invariant (LTI) controller if and only if there exist vectors* $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ *such that* $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *are linearly independent.*

Proof. From Theorem 2.1, we see that formation stabilization to any $(\mathbf{r}_0, \mathbf{v}_0)$ is equivalent to formation stabilization to the origin. We apply the result of [8] to develop conditions for formation stabilization to the origin. Wang and Davison ([8]) prove that a decentralized system is stabilizable using a linear dynamic controller if and only if the system has no unstable (or marginally stable) **fixed modes**. (We refer the reader to [8] for details on fixed modes.) Hence, we can justify the condition above, by proving that our system has no unstable fixed modes if and only if there exist vectors $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are linearly independent.

It is easy to show (see [8]) that the fixed modes of a decentralized control system are a

subset of the modes of the system matrix, in our case $\begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix}$. The eigenvalues of this system matrix are identically 0, so our system is stabilizable if and only if 0 is not a fixed mode. We can test whether 0 is a fixed mode of the system by using the determinant-based condition of [8], which reduces to the following in our example: the eigenvalue 0 is a fixed mode if and only if

$$det\left(\begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} K \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix}\right) = 0, \tag{2.11}$$

for all K of the form $\begin{bmatrix} K_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & K_n \end{bmatrix}$, where each K_i is a real matrix of dimension $1 \times 2m_i$.

To simplify the condition (2.11) further, it is helpful to develop some further notation for the matrices $K_1, \ldots, K_n$. In particular, we write the matrix K_i as follows:

$$K_i = \begin{bmatrix} \mathbf{k}_p(i) & \mathbf{k}_v(i) \end{bmatrix},$$

where each of the four submatrices are length-m_i row vectors. Our subscript notation for these vectors represents that these control gains multiply positions (p) and velocities (v), respectively.

In this notation, the determinant in condition (2.11) can be rewritten as follows:

$$det\left(\begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} K \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix}\right) = det\left(\begin{bmatrix} 0 & I_n \\ Q_p & Q_v \end{bmatrix}\right), \tag{2.12}$$

where

$$Q_p = \begin{bmatrix} \mathbf{k}_p(1)(G_1) \\ \vdots \\ \mathbf{k}_p(n)(G_n) \end{bmatrix}$$

and

$$Q_v = \begin{bmatrix} \mathbf{k}_v(1)(G_1) \\ \vdots \\ \mathbf{k}_v(n)(G_n) \end{bmatrix}.$$

This determinant is identically zero for all K if and only if the rank of Q_p is less than n for all K, so the stabilizability of our system can be determined by evaluating the rank of Q_p.

Now let's prove the necessity and sufficiency of our condition.

Necessity Assume that there is not any set of vectors $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are linearly independent. Then there is row vector $\mathbf{w}$ such that $\sum_{i=1}^{n} \mathbf{k}_i G_i \neq \mathbf{w}$ for any $\mathbf{k}_1, \ldots, \mathbf{k}_n$. Now consider linear combinations of the rows of Q_p. Such linear

combinations can always be written in the form

$$\sum_{i=1}^{n} \alpha_i \tilde{\mathbf{k}}_p(i) G_i \tag{2.13}$$

Thus, from the assumption, we cannot find a linear combination of the rows that equals the vector $\mathbf{w}$. Hence, the n rows of Q_p do not span the space $\mathbf{R}^n$ and so are not all linearly independent. The matrix Q_p therefore does not have rank n, so 0 is a fixed mode of the system, and the system is not formation stabilizable.

Sufficiency Assume that there is a set of vectors $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are linearly independent. Let $\mathbf{k}_1, \ldots, \mathbf{k}_n$ be the row vectors such that $\mathbf{k}_i G_i = \mathbf{b}_i^T$. Now let's choose the control matrix K in our system as follows: $\tilde{\mathbf{k}}_p(i) = \mathbf{k}_i$. In this case, the matrix Q_p can be written as follows:

$$Q_p = \begin{bmatrix} k_1 G_1 \\ \vdots \\ k_n G_n \end{bmatrix} = \begin{bmatrix} b_1^T \\ \vdots \\ b_n^T \end{bmatrix}. \tag{2.14}$$

Hence, the rank of Q_p is n, 0 is not a fixed mode of the system, and the system is formation stabilizable.

□

By applying the results of [6], we can generalize the above condition for stabilization of

linear double-integrator networks to prove semi-global stabilization of double-integrator networks with input saturation.

Theorem 2.3 *A double-integrator network with actuator saturation is semi-globally formation stabilizable to any formation using a dynamic LTI controller if and only if there exist vectors* $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ *such that* $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *are linearly independent.*

Proof. Again, formation stabilization to any $(\mathbf{r}_o, \mathbf{v}_o)$ is equivalent to formation stabilization to the origin. We now apply the theorem of [6], which states that semi-global stabilization of a decentralized control system with input saturation can be achieved if and only if

- The eigenvalues of the open-loop system lie in the closed left-half plane.
- All fixed modes of the system when the saturation is disregarded lie in the open left-half plane.

We recognize that the open-loop eigenvalues of the double-integrator network are all zero, and so lie in the closed left-half plane. Hence, the condition of [6] reduces to a check for the presence or absence of fixed modes in the linear closed-loop system. We have already shown that all fixed modes lie in the OLHP if and only if there exist vectors $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ such that $\mathbf{b}_1, \ldots, \mathbf{b}_n$ are linearly independent. Hence, the theorem is proved. □

We mentioned earlier that our formation-stabilization results hold whenever the position observations have the appropriate graph structure, regardless of the velocity measurement

topology. Let us formalize this result:

Theorem 2.4 *A position-measurement double-integrator network (with actuator saturation) is (semi-globally) formation stabilizable to any formation using a dynamic LTI controller if and only if there exist vectors* $\mathbf{z}_1 \in Ra(G_1^T), \ldots, \mathbf{z}_n \in Ra(G_n^T)$ *such that* $\mathbf{z}_1, \ldots, \mathbf{z}_n$ *are linearly independent.*

Proof. The proof of Theorem 2.2 makes clear that only the topology of the position measurements play a role in deciding the stabilizability of a double-integrator network. Thus, we can achieve stabilization for a position-measurement double-integrator network by disregarding the velocity measurements completely, and hence the same conditions for stabilizability hold. □

The remainder of this section is devoted to remarks, connections between our results and those in the literature, and examples.

Remark, Single Observation Case In the special case in which each agent makes a single position observation and a single velocity observation, the condition for stabilizability is equivalent to simple observability of all closed right-half-plane poles of the open-loop system. Thus, in the single observation case, centralized linear and/or state-space form nonlinear control do not offer any advantage over our decentralized control in terms of stabilizability. This point makes clear the importance of studying the multiple-observation scenario.

Remark, Networks with Many Agents We stress that conditions required for formation stabilization do not in any way restrict the number of agents in the network or their relative positions upon formation stabilization. That is, a network of any number of agents can be formation stabilized to an arbitrary formation, as long as the appropriate conditions on the full graph matrix G are met. The same holds for the other notions and means for stabilization that we discuss in subsequent sections; it is only for collision avoidance that the details of the desired formation become important. We note that the performance of the controlled network (e.g., the time required for convergence to the desired formation) may have some dependence on the number of agents. We plan to quantify performance in future work.

Connection to [1] Earlier, we discussed that the Laplacian sensing architecture of [1] is a special case of our sensing architecture. Now we are ready to compare the stabilizability results of [1] with our results, within the context of double-integrator agent dynamics. Given a Laplacian sensing architecture, our condition in fact shows that the system is not stabilizable; this result is expected, since the Laplacian architecture can only provide convergence in a relative frame. In the next section, we shall explicitly consider such relative stabilization. For comparison here, let us equivalently assume that relative positions/velocities are being stabilized, so that we can apply our condition to a grounded Laplacian. We can easily check that we are then always able to achieve fomation stabilization. It turns out that the same result can be recovered from the simultaneous stabilization formulation of [1] (given double-integrator dynamics), and so the two analyses match. However, we note

that, for more general communication topologies, our analysis can provide broader conditions for stabilization than that of [1], since we allow use of different controllers for each agent. Our approach also has the advantage of producing an easy-to-check necessary and sufficient condition for stabilization.

2.3.1 Examples

It's illuminating to apply our stabilizability condition to the examples introduced above.

Example: String of Vehicles Let's choose vectors $\mathbf{b}_{1x}$, $\mathbf{b}_{2x}$ and $\mathbf{b}_{3x}$ as the first rows of G_{1x}, G_{2x}, and G_{3x}, respectively. Let us also choose $\mathbf{b}_{1y}$, $\mathbf{b}_{2y}$ and $\mathbf{b}_{3y}$ as the second rows of G_{1y}, G_{2y}, and G_{3y}, respectively. It is easy to check that $\mathbf{b}_{1x}$, $\mathbf{b}_{2x}$, $\mathbf{b}_{3x}$, $\mathbf{b}_{1y}$, $\mathbf{b}_{2y}$ and $\mathbf{b}_{3y}$ are linearly independent, and so the vehicles are stabilizable. The result is sensible, since Vehicle 1 can sense the target position directly, and Vehicles 2 and 3 can indirectly sense the position of the target using the vehicle(s) ahead of it in the string. Our analysis of a string of vehicles is particularly interesting, in that it shows we can complete task dynamics for non-leader-follower architectures, using the theory of [8]. This result complements the studies of [3], on leader-follower architectures.

Example: Coordination Using an Intermediary We again expect the vehicle formation to be stabilizable, since both the observed target coordinates can be indirectly sensed by the other vehicles, using the sensing architecture. We can verify stabilizability by choosing

vectors in the range spaces of the transposed graph matrices, as follows:

$$\begin{aligned}
\mathbf{b}_{1x}^T &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathbf{b}_{1y}^T &= \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{b}_{2x}^T &= \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \\
\mathbf{b}_{2y}^T &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
\mathbf{b}_{3x}^T &= \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
\mathbf{b}_{3y}^T &= \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{2.15}$$

It is easy to check that these vectors are linearly independent, and hence that the system is stabilizable.

Consideration of this system leads to an interesting insight on the function of the intermediary agent. We see that stabilization using the intermediary is only possible because this agent can make two observations, and has available two actuators. If the intemediary is constrained to make only one observation or can only be actuated in one direction (for example, if it is constrained to move only on the x axis), then stabilization is not possible.

Example: Measurement Failures It is straightforward to check that decentralized control is not possible if Aircraft 2 has access to the position and velocity of Aircraft 3, but is possible if Aircraft 2 has access to the position and velocity of Aircraft 1. This example is

interesting because it highlights the restriction placed on stabilizability by the decentralization of the control. In this example, the full graph matrix G has full rank for both observation topologies considered, and hence we can easily check the centralized control of the aircraft is possible in either case. However, decentralized control is not possible when Aircraft 2 only knows the location of Aircraft 3, since there is then no way for Aircraft 2 to deduce its own location.

2.4 Alignment Stabilization

Sometimes, a network of communicating agents may not require formation stabilization, but instead only require that certain combinations of the agents' positions and velocities are convergent. For instance, flocking behaviors may involve only convergence of differences between agents' positions or velocities (e.g., [4]). Similarly, a group of agents seeking a target may only require that their center of mass is located at the target. Also, we may sometimes only be interested stabilization of a double-integrator network from some initial conditions—in particular, initial conditions that lie in a subspace of R^n. We view both these problems as **alignment stabilization** problems because they concern partial stabilization and hence alignment rather than formation of the agents. As with formation stabilization, we can employ the fixed-mode concept of [8] to develop conditions for alignment stabilization.

We begin with a definition for alignment stabilization:

Definition 2 *A double-integrator network can be aligned with respect to a $\widehat{n} \times n$ weighting matrix Y if a proper linear time-invariant (LTI) dynamic controller can be constructed for it, so that the $Y\mathbf{r}$ and $Y\dot{\mathbf{r}}$ are globally asymptotically convergent to the origin.*

One note is needed: we define alignment stabilization in terms of (partial) convergence of the state to the origin. As with formation stabilization, we can study alignment to a fixed point other than the origin. We omit this generalization for the sake of clarity.

The following theorem provides a necessary and sufficient condition on the sensing architecture for alignment stabilization of a linear double integrator network.

Theorem 2.5 *A linear double-integrator network can be aligned with respect to Y if and only if there exist $\mathbf{b}_1 \in Ra(G_1^T), \ldots, \mathbf{b}_n \in Ra(G_n^T)$ such that the eigenvectors/generalized eigenvectors of $V \triangleq \begin{bmatrix} \mathbf{b}_1^T \\ \cdots \\ \mathbf{b}_n^T \end{bmatrix}$ that correspond to zero eigenvalues all lie in the null space of Y.*

Proof. For clarity and simplicity, we prove the theorem in the special case that each agent has available only one observation (i.e., $G_1, \ldots, G_n$ are all row vectors). We then outline the generalization to this proof to the case of vector observations.

In the scalar-observation case, the condition above reduces to the following: a linear double-integrator network can be aligned with respect to Y if and only if the eigenvectors and generalized eigenvectors of G corresponding to its zero eigenvalues lie in the null space of

Y. We prove this condition in several steps:

1. *We characterize the fixed modes of the linear double-integrator network (see [8] for background on fixed modes).* In particular, assume that the network has a full graph matrix G with $\overline{n}$ zero eigenvalues. Then the network has $2\overline{n}$ fixed modes at the origin. That is, the matrix $A_c = \begin{bmatrix} 0 & I \\ K_1 G & K_2 G \end{bmatrix}$ has $2\overline{n}$ eigenvalues of zero for any diagonal K_1 and K_2. To show this, let's consider any eigenvector/generalized eigenvector $\mathbf{w}$ of G corresponding to a 0 eigenvalue. It is easy to check that the vector $\begin{bmatrix} \mathbf{w} \\ 0 \end{bmatrix}$ is an eigenvector/generalized eigenvector of A_c with eigenvalue 0, regardless of K_1 and K_2. We can also check that $\begin{bmatrix} 0 \\ \mathbf{w} \end{bmatrix}$ is a generalized eigenvector of A_c with eigenvalue zero. Hence, since G has $\overline{n}$ eigenvectors/generalized eigenvectors associated with the zero eigenvalue, the network has at least $2\overline{n}$ fixed modes. To see that the network has no more than $2\overline{n}$ fixed modes, we choose $K_1 = I_n$ and $K_2 = [0]$; then the eigenvalues of A_c are the square roots of the eigenvalues of G, and so A_c has exactly $2\overline{n}$ zero eigenvalues. Notice that we have not only found the number of fixed modes of the network but also specified the eigenvector directions associated with these modes.

2. *We characterize the eigenvalues and eigenvectors of the closed-loop system when decentralized dynamic feedback is used to control the linear double-integrator network.* For our model, the closed-loop system when dynamic feedback is used is given by $A_{dc} =$

$\begin{bmatrix} 0 & I & 0 \\ K_1G & K_2G & Q \\ R_1G & R_2G & S \end{bmatrix}$, where R_1, R_2, and S are appropriately-dimensioned block diagonal matrices (see [8] for details). It has been shown in [8] that the eigenvalues of A_{dc} that remain fixed regardless of the controller used are identical to the number of fixed modes of the system. Further, it has been shown that the remaining eigenvalues of A_{dc} can be moved to the OLHP through sufficient choice of the control gains. Hence, for the double-integrator network, we can design a controller such that all but $2\overline{n}$ eigenvalues of A_{dc} lie in the OLHP and the remaining $2\overline{n}$ eigenvalues are zero. In fact, we can determine the eigenvectors of A_{dc} associated with these zero eigenvalues. For each eigenvector/generalized eigenvector $\mathbf{w}$ of G, the vectors $\begin{bmatrix} \mathbf{w} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$ and $\begin{bmatrix} \mathbf{0} \\ \mathbf{w} \\ \mathbf{0} \end{bmatrix}$ are eigenvectors/generalized eigenvectors of A_{dc} corresponding to eigenvalue 0. Hence, we have specified the number of zero modes of A_{dc}, and have found that the eigenvector directions associated with these modes remain fixed (as given above) regardless of the controller used.

3. Finally, we can prove the theorem. Assume that we choose a control law such that all the eigenvalues of A_{dc} except the fixed modes at the origin are in the OLHP—we can always do this. Then it is obvious that $Y\mathbf{r}$ is globally asympotically convergent to

the origin if and only if $\begin{bmatrix} Y & 0 & 0 \end{bmatrix}$ is orthogonal to all eigenvectors associated with eigenvalues of A_{dc} at the origin. Considering these eigenvectors, we see that global asymptotic stabilization to the origin is possible if and only if the all eigenvectors of G associated with zero eigenvalues lie in the nullspace of Y.

In the vector-observation case, proof of the condition's sufficiency is straightforward: we can design a controller that combines the vector observations to generate scalar observations for each agent, and then uses these scalar observations to control the system. The proof of necessity in the vector case is somewhat more complicated, because the eigenvectors of A_c and A_{dc} corresponding to zero eigenvalues change direction depending on the controller used. Essentially, necessity is proven by showing that using a dynamic control does not change the class of fixed-mode eigenvectors, and then showing that the possible eigenvector directions guarantee that stabilization is impossible when the condition is not met. We leave the details of this analysis to a future work; we believe strongly that our approach for characterizing alignment (partial stabilization) can be generalized to the class of decentralized control systems discussed in [8], and hope to approach alignment from this perspective in future work.

□

2.4.1 Examples of Alignment Stabilization

Laplacian Sensing Topologies As discussed previously, formation stabilization is not achieved when the graph topology is Laplacian. However, by applying the alignment stabilization condition, we can verify that differences between agent positions/velocities in each connected graph component are indeed stabilizable. This formulation recovers the results of [1] (within the context of double-integrator networks), and brings our work in alignment (no pun intended) with the studies of [4]. It more generally highlights an alternate viewpoint on formation stabilization analysis given a grounded Laplacian topology.

We consider a specific example of alignment stabilization given a Laplacian sensing topology here, that is similar to the flocking example of [4]. The graph matrix[3] in our example is

$$G = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \tag{2.16}$$

The condition above can be used to show alignment stabilization of differences between agents' positions or velocities, e.g., with respect to $Y = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \end{bmatrix}$.

[3] To be precise, in our simulations we assume that agents move in the plane. The graph matrix shown here specifies the sensing architecture in each vector direction.

Matching Customer Demand Say that three automobile manufacturers are producing sedans to meet fixed, but unknown, customer demand for this product. An interesting analysis is to determine whether or not these manufacturer can together produce enough sedans to exactly match the consumer demand. We can view this problem as an alignment stabilization one, as follows. We model the three manufacturers as decentralized agents, whose production outputs r_1, r_2, and r_3, are modeled as double integrators. Each agent clearly has available its own production output as an observation. For convenience, we define a fourth agent that represents the fixed (but unknown) customer demand; this agent necessarily has no observations available and so cannot be actuated. We define r_d to represent this demand. We are concerned with whether $r_1 + r_2 + r_3 - r_d$ can be stabilized. Hence, we are studying an alignment problem.

Using the condition above, we can trivially see that alignment is not possible if the agents only measure their own position. If at least one of the agents measures $r_1 + r_2 + r_3 - r_d$ (e.g., through surveys), then we can check that stabilization is possible. When other observations that use r_d are made, then the alignment may or may not occur; we do not pursue these further here. It would be interesting to study whether or not competing manufacturers such as these in fact use controllers that achieve alignment stabilization.

2.5 Existence and Design of Static Stabilizers

Above, we developed necessary and sufficient conditions for the stabilizability of a group of communicating agents with double-integrator dynamics. Next, we give sufficient conditions for the existence of a *static* stabilizing controller[4]. We further discuss several approaches for designing good static stabilizers, that take advantage of some special structures in the sensing architecture.

2.5.1 Sufficient Condition for Static Stabilization

The following theorem describes a sufficient condition on the sensing architecture for the existence of a static stabilizing controller.

Theorem 2.6 *Consider a linear double-integrator network with graph matrix G. Let $\mathcal{K}$ be the class of all block diagonal matrices of the form* $\begin{bmatrix} \mathbf{k}_1 & & \\ & \ddots & \\ & & \mathbf{k}_n \end{bmatrix}$, *where $\mathbf{k}_i$ is a row vector with m_i entries (recall that m_i is the number of observations available to agent i). Then the double-integrator system has a static stabilizing controller (i.e., a static controller that achieves formation stabilization) if there exists a matrix $K \in \mathcal{K}$ such that the eigenvalues of KG are in the open left-half-plane (OLHP).*

[4]We consider a controller to be static if the control inputs at each time are linear functions of the concurrent observations (in our case the position and velocity observations).

Proof. We prove this theorem by constructing a static controller for which the overall system's closed-loop eigenvalues are in the OLHP whenever the eigenvalues of KG are in the OLHP. Based on our earlier development, it is clear that the closed-loop system matrix takes the form

$$A_c = \begin{bmatrix} 0 & I \\ K_1 G & K_2 G \end{bmatrix}, \tag{2.17}$$

where K_1 and K_2 are control matrices that are constrained to be in $\mathcal{K}$, but are otherwise arbitrary.

Given the theorem's assumption, it turns out we can guarantee that the eigenvalues of A_c are in the OLHP by choosing the control matrices as $K_1 = K$ and $K_2 = aK$, where a is a sufficiently large positive number. With these control matrices, the closed loop system matrix becomes

$$A_c = \begin{bmatrix} 0 & I \\ KG & aKG \end{bmatrix}. \tag{2.18}$$

To show that the $2n$ eigenvalues A_c are in the OLHP, we relate these eigenvalues to the n eigenvalues of KG.

To relate the eigenvalues, we construct the left eigenvectors of A_c. In particular, we consider vectors of the form $\mathbf{w}' = \begin{bmatrix} \mathbf{v}' & \alpha\mathbf{v}' \end{bmatrix}$, where $\mathbf{v}'$ is a left eigenvector of KG. Then

$$\mathbf{w}' A_c = \begin{bmatrix} \alpha\rho\mathbf{v}' & \mathbf{v}'(1 + \alpha a\rho), \end{bmatrix} \tag{2.19}$$

where ρ—the eigenvalue of KG corresponding to the eigenvector $\mathbf{v}'$—lies in the OLHP.

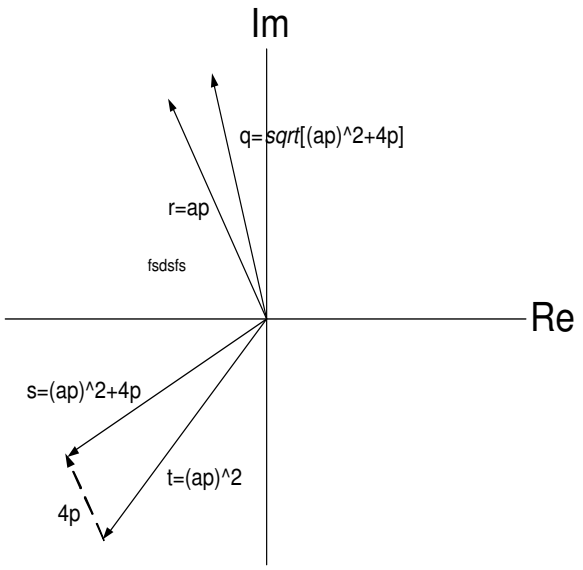

Figure 2.1: We introduce the notation used for the geometric proof that the eigenvalues of A_c are in the OLHP.

This vector is a left eigenvector of A_c with eigenvalue λ if and only if $\alpha\rho = \lambda$ and $1 + \alpha a\rho = \alpha\lambda$. Substituting $\alpha = \frac{\lambda}{\rho}$ into the second equation and rearranging, we find that $\lambda^2 - a\rho\lambda - \rho = 0$. This quadratic equation yields two solutions, and hence we find that two eigenvalues and eigenvectors of A_c can be specified using each left eigenvector of KG. In this way, all $2n$ eigenvalues of A_c can be specified in terms of the n eigenvalues of KG.

Finally, it remains to be shown that the eigenvalues of A_c are negative. Applying the quadratic formula, we find that each eigenvalue λ of A_c can be found from an eigenvalue ρ of KG, as $\lambda = \frac{a\rho \pm \sqrt{(a\rho)^2 + 4\rho}}{2}$. Without loss of generality, let's assume that ρ lies in the second quadrant of the complex plane. (It's easy to check that eigenvalues of A_c corresponding to ρ in the third quadrant are complex conjugates of eigenvalues corresponding to ρ in the second quadrant). We can show that the eigenvalues λ will have negative real parts, using

a geometric argument. The notation that we use in this geometric argument is shown in Figure 2.1. In this notation, $\lambda = \frac{r \pm q}{2}$. Since r has negative real part, we can prove that λ has negative real part by showing that the magnitude of the real part of q is smaller than the magnitude of the real part of r. To show this, first consider the complex variables s and t. Using the law of cosines, we can show that the length (in the complex plane) of s is less than the length of t whenever

$$a > \sqrt{\frac{2}{|\rho| cos(90 - tan^{-1} \frac{-Re(\rho)}{Im(\rho)})}}. \tag{2.20}$$

In this case, the length (magnitude) of q is also less than the magnitude of r. Hence, the magnitude of the real part of q is less than the magnitude of the real part of r (because the phase angle of q is smaller), and the eigenvalue λ has negative real part.

Thus, if we choose

$$a > \max_{\rho} \sqrt{\frac{2}{|\rho| cos(90 - tan^{-1} \frac{-Re(\rho)}{Im(\rho)})}}, \tag{2.21}$$

then all eigenvalues of A_c are guaranteed to be negative. Q.E.D.

□

Our proof demonstrates how to design a stabilizing controller, whenever we can find a matrix K such that the eigenvalues of KG are in the OLHP. Since this stabilizing controller uses a high gain on the velocity measurements, we henceforth call it a *high velocity gain* (HVG) controller.

Using a simple scaling argument, we can show that static stabilization is possible in a semi-global sense whenever we can find a matrix K such that the eigenvalues of KG are in the OLHP, even if the actuators are subject to saturation. We present this result in the following theorem.

Theorem 2.7 *Consider a double-integrator network with actuator saturation that has graph matrix G. Let $\mathcal{K}$ be the class of all block diagonal matrices of the form* $\begin{bmatrix} \mathbf{k}_1 & & \\ & \ddots & \\ & & \mathbf{k}_n \end{bmatrix}$*, where $\mathbf{k}_i$ is a row vector with m_i entries. Then the double-integrator network with actuator saturation has a semi-global static stabilizing controller (i.e., a static controller that achieves formation stabilization in a semi-global sense) if there exists a matrix $K \in \mathcal{K}$ such that the eigenvalues of KG are in the open left-half-plane (OLHP).*

Proof. In the interest of space, we present an outline of the proof here. Since we are proving semi-global stabilization, we can assume that the initial system state lies within some finite-sized ball. Notice that, if the double-integrator network were not subject to input saturation, we could find a static stabilizing controller. Further note that, if the static stabilizing controller were applied, the trajectory of the closed-loop system would remain within a larger ball. Say that the closed-loop system matrix for the linear network's stabilizing controller is

$$A_c = \begin{bmatrix} 0 & I \\ KG & aKG \end{bmatrix}. \tag{2.22}$$

Then it is easy to check that the closed-loop system matrix

$$\widehat{A}_c = \begin{bmatrix} 0 & I \\ \frac{KG}{\zeta^2} & \frac{aKG}{\zeta} \end{bmatrix}. \tag{2.23}$$

also represents a static stabilizer for the linear network. Further, the trajectory followed by the state when this new controller is used is identical to the one using the original controller, except that the time axis for the trajectory is scaled by ζ. Hence, we know that the trajectory is bounded within a ball. Thus, if we choose large enough ζ, we can guarantee that the input magnitude is always strictly less than 1 (i.e., that the actuators never saturate), while also guaranteeing stabilization. Such a choice of controller ensures stabilization even when the actuators are subject to saturation, and hence the theorem has been proved.

□

Remark Multiple Integrator Case Now we give sufficient conditions for a static stabilizing controller in a network of agents with multiple integrator dynamics. The closed loop system matrix of the network becomes

$$A_c = \begin{bmatrix} 0 & I & & & \\ & & \ddots & & \\ & & & & I \\ K_1G & K_2G & K_3G & \cdots & K_nG \end{bmatrix} \tag{2.24}$$

Given the assumption from Theorem 6 that the eigenvalues of KG are in the OLHP, we can guarantee the eigenvalues of A_c are in the OLHP by choosing the control matrices as $K_n = a^{n-1}K$. Here we are taking advantage of the multi time-scale behavior of dynamical systems with fast and slow transients in their response to an input caused by singular perturbation.

For the sake of clarity, let's first consider the case of $n = 3$, in other words the input is an acceleration. In this case, the closed loop system matrix is

$$A_c = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \\ KG & aKG & a^2KG \end{bmatrix} \tag{2.25}$$

We relate the eigenvalues of A_c to the eigenvalues of KG we again consider the left eigenvectors of A_c as vectors of the form $\mathbf{w}' = \begin{bmatrix} \mathbf{v}' & \alpha\mathbf{v}' & \beta\mathbf{v}' \end{bmatrix}$ where $\mathbf{v}'$ is a left eigenvector of KG corresponding to the eigenvalue ρ. Then

$$\mathbf{w}'A_c = \begin{bmatrix} \beta\rho\mathbf{v}' & (1 + a\beta\rho)\mathbf{v}' & (\alpha + \beta a^2\rho)\mathbf{v}' \end{bmatrix} \tag{2.26}$$

where ρ lies in the OLHP. This vector is the left eigenvector of A_c corresponding to eigenvalue λ if and only if $\beta\rho = \lambda$, $1 + a\beta\rho = \alpha\lambda$, and $\alpha + \beta a^2\rho = \beta\lambda$. By rearranging these relationships, we form the cubic $\lambda^3 - a^2\rho\lambda^2 - a\rho\lambda - \rho = 0$. Using the Routh-Hurwitz stability criterion, the stability condition on the control matrix gain a is $a^3 > -1/\rho$. Note that since ρ lies in the OLHP, a is a positive constant.

Using the same method as above, the general condition on the control matrix gain, a, for n-integrator dynamics is

$$a^n > -1/\rho \tag{2.27}$$

If a is sufficiently large, a network of agents with n-integrator dynamics can be stabilized with a static controller.

2.5.2 Design of Static Stabilizers

Above, we showed that a static stabilizing controller for our decentralized system can be found, if there exists a control matrix K such that the eigenvalues of KG are in the OLHP. Unfortunately, this condition does not immediately allow us to design the control matrix K or even to identify graph matrices G for which a HVG controller can be constructed, since we do not know how to choose a matrix K such that the eigenvalues of KG are in the OLHP.

In this section, we discuss approaches for identifying from the graph matrix whether an HVG controller can be constructed, and for designing the control matrix K. First, we show (trivially) that HVG controllers can be developed when the graph matrix has positive eigenvalues, and give many examples of sensing architectures for which the graph matrix eigenvalues are positive. Second, we discuss some simple variants on the class of graph matrices with positive eigenvalues, for which HVG controllers can also be developed. Third, we use an eigenvalue sensitivity-based argument to show that HVG con-

trollers can be constructed for a very broad class of graph matrices. (For convenience and clarity of presentation, we restrict the sensitivity-based argument to the case where each agent has available only one observation, but note that the generalization to the multiple-observation case is straightforward). Although this eigenvalue sensitivity-based argument sometimes does not provide good designs (because eigenvalues are guaranteed to be in the OLHP only in a limiting sense), the argument is important because it highlights the broad applicability of static controllers and specifies a systematic method for their design.

2.5.2.1 Graph Matrices with Positive Eigenvalues

If each agent has available one observation (so that the graph matrix G is square) and the eigenvalues of G are strictly positive, then a stabilizing HVG controller can be designed by choosing $K = -I_n$.

The proof is immediate: the eigenvalues of $KG = -G$ are strictly negative, so the condition for the existence of a stabilizing HVG controller is satisfied.

Here are some examples of sensing architectures for which the graph matrix has strictly positive eigenvalues:

- A grounded Laplacian matrix is known to have strictly positive eigenvalues (see, e.g., [7]). Hence, if the sensing architecture can be represented using a grounded Laplacian graph matrix, then the a static control matrix $K = -I$ can be used to stabilize the system.

- A wide range of matrices besides Laplacians are also known to have positive eigenvalues. For instance, any strictly **diagonally-dominant matrix**—one in which the diagonal entry on each row is larger than the sum of the absolute values of all off-diagonal entries—has positive eigenvalues. Diagonally-dominant graph matrices are likely to be observed in systems in which each agent has considerable ability to accurately sense its own position.

- If there is a positive diagonal matrix L such that GL is diagonally dominant, then the eigenvalues of G are known to be positive. In some examples, it may be easy to observe that a scaling of this sort produces a diagonally-dominant matrix.

2.5.2.2 Eigenvalue Sensitivity-Based Controller Design, Scalar Observations

Using an eigenvalue sensitivity (perturbation) argument, we show that HVG controllers can be explicitly designed for a very wide class of sensing architectures. In fact, we find that only a certain sequential full-rank condition is required to guarantee the existence of a static stabilizer. While our condition is not a necessary and sufficient one, we believe that it captures most sensing topologies of interest.

The statement of our theorem requires some further notation:

- Recall that we label agents using the integers $1, \ldots, n$. We define an *agent list* $\mathbf{i} = \{i_1, \ldots, i_n\}$ to be an ordered vector of the n agent labels. (For instance, if there are 3 agents, $\mathbf{i} = \{3, 1, 2\}$ is an agent list).

- We define the kth *agent sublist* of the agent list $\mathbf{i}$ to be a vector of the first k agent labels in $\mathbf{i}$, or $\{i_1, \ldots, i_k\}$. We use the notation $\mathbf{i}_{1:k}$ for the kth agent sublist of $\mathbf{i}$.

- We define the kth **subgraph matrix** associated with the agent list to be the $k \times k$ submatrix of the graph matrix corresponding to the agents in the kth agent sublist. More precisely, we define the matrix $D(\mathbf{i}_{1:k})$ to have k rows and n columns. Entry i_w of each row w is assumed to be unity, and all other entries are assumed to be 0. The kth subgraph matrix is given by $D(\mathbf{i}_{1:k})GD(\mathbf{i}_{1:k})'$.

The condition on the graph matrix required for design of a HVG controller using eigenvalue sensitivity arguments is given in the following theorem:

Theorem 2.8 *If there exists an agent list* $\mathbf{i}$ *such that the* k*th subgraph matrix associated with this agent list has full rank for all* k*, then we can construct a stabilizing HVG controller for the decentralized control system.*

Proof. We prove the theorem above by constructing a control matrix K such that the eigenvalues of KG are in the OLHP. More specifically, we construct a sequence of control matrices, for which more and more of the eigenvalues of KG are located in the OLHP and hence prove that there exists a control matrix such that all eigenvalues of KG are in the OLHP.

Precisely, we show how to iteratively construct a sequence of control matrices $K(\mathbf{i}_{1:1}), \ldots, K(\mathbf{i}_{1:n})$, such that $K(\mathbf{i}_{1:k})G$ has k eigenvalues in the OLHP. In constructing the control matrices, we use the agent list $\mathbf{i}$ for which the assumption in the theorem is

satisfied.

First, let's define the matrix $K(\mathbf{i}_{1:1})$ to have a single non-zero entry: the diagonal entry corresponding to i_1, or K_{i_1}. Let's choose K_{i_1} to equal $-sgn(G_{i_1,i_1})$—i.e., the negative of the sign of the (non-zero) diagonal entry of G corresponding to agent i_1. Then $K(\mathbf{i}_{1:1})G$ has a single non-zero row, with diagonal entry $-G_{i_1,i_1}sgn(G_{i_1,i_1})$. Hence, $K(\mathbf{i}_{1:1})G$ has one negative eigenvalue, as well as $n-1$ zero eigenvalues. Note that the one non-zero eigenvalue is simple (non-repeated).

Next, let's assume that there exists a matrix $K(\mathbf{i}_{1:k})$ with non-zero entries $K_{i_1}, \ldots, K_{i_k}$, such that $K(\mathbf{i}_{1:k})G$ has k simple negative eigenvalues, and $n-k$ zero eigenvalues. Now let's consider a control matrix $K(\mathbf{i}_{1:k+1})$ that is formed by adding a non-zero entry $K_{i_{k+1}}$ (i.e., a non-zero entry corresponding to agent i_{k+1}) to $K(\mathbf{i}_{1:k})$, and think about the eigenvalues of $K(\mathbf{i}_{1:k+1})G$. The matrix $K(\mathbf{i}_{1:k+1})G$ has $k+1$ non-zero rows, and so has at most $k+1$ non-zero eigenvalues. The eigenvalues of $K(\mathbf{i}_{1:k+1})G$ are the eigenvalues of its submatrix corresponding to the kth agent sublist, or $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k+1})GD(\mathbf{i}_{1:k+1})'$. Notice that this matrix can be constructed by scaling the rows of the $(k+1)$th agent sublist, and hence has full rank. Next, note that we can rewrite $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k+1})GD(\mathbf{i}_{1:k+1})'$ as $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k})GD(\mathbf{i}_{1:k+1})' + D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{k+1})GD(\mathbf{i}_{1:k+1})'$, where $K(\mathbf{i}_{k+1})$ only has diagonal entry $K_{i_{k+1}}$ nonzero. Thus, we can we can view $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k+1})GD(\mathbf{i}_{1:k+1})'$ as a perturbation of $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k})GD(\mathbf{i}_{1:k+1})'$—which has the same eigenvalues as $K(\mathbf{i}_{1:k})G$—by the row vector $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{k+1})GD(\mathbf{i}_{1:k+1})'$. Because

$D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k+1})GD(\mathbf{i}_{1:k+1})'$ has full rank, we can straightforwardly invoke a standard eigenvalue-sensitivity result to show that $K_{i_{k+1}}$ can be chosen so that all the eigenvalues of

$D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k+1})GD(\mathbf{i}_{1:k+1})'$ are simple and negative. Essentially, we can choose the sign of the smallest eigenvalue—which is a perturbation of the zero eigenvalue of $D(\mathbf{i}_{1:k+1})K(\mathbf{i}_{1:k})GD(\mathbf{i}_{1:k+1})'$—by choosing the sign of $K_{i_{k+1}}$ properly, and we can ensure that all eigenvalues are positive and simple by choosing small enough $K_{i_{k+1}}$. Thus, we have constructed $K(\mathbf{i}_{1:k+1})$ such that $K(\mathbf{i}_{1:k+1})G$ has $k+1$ simple non-zero eigenvalues. Hence, we have proven the theorem by recursion, and have specified broad conditions for the existence and limiting design of static (HVG) controllers.

□

2.5.3 Examples of Static Control

We develop static controllers for the four examples that we introduced earlier. Through simulations, we explore the effect of the static control gains on performance of the closed-loop system.

2.5.3.1 Example: Coordination Using an Intermediary

The following choice for the static gain matrix K places all eigenvalues of KG in the OLHP, and hence permits stabilization with a HVG controller:

$$K = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.28}$$

As K is scaled up, we find that the vehicles converge to the target faster and with less overshoot. When the HVG controller parameter a is increased, the agents converge much more slowly but also gain a reduction in overshoot. Figures 2.2, 2.3, and 2.4 show the trajectories of the vehicles, and demonstrate the effect of the static control gains on the performance of the closed-loop system.

Also, if a is made sufficiently large, the eigenvalues of A_c move close to the imaginary axis. This is consistent with the vehicles' slow convergence rate for large a.

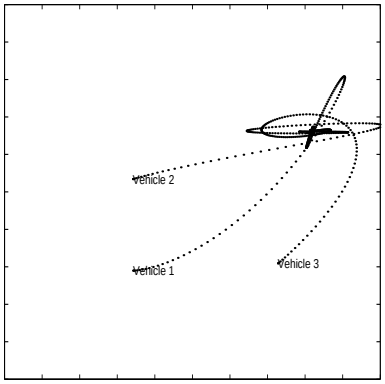

Figure 2.2: Vehicles converging to a target: coordination with an intermediary.

2.5.3.2 Example: Vehicle Velocity Alignment, Flocking

Finally, we simulate an example, with some similarity to the examples of [4], that demonstrates alignment stabilization. In this example, a controller is designed so that five vehicles with relative position and velocity measurements converge to the same (but initial-condition dependent) velocity. Figures 2.5 demonstrates the convergence of the velocities in the x-direction for two sets of initial velocities. We note the different final velocities achieved by the two simulations.

2.6 Collision Avoidance in the Plane

Imagine a pedestrian walking along a crowded thoroughfare. Over the long term, the pedestrian completes a task—i.e., she/he moves toward a target or along a trajectory. As the pedestrian engages in this task, however, she must constantly take evasive action to

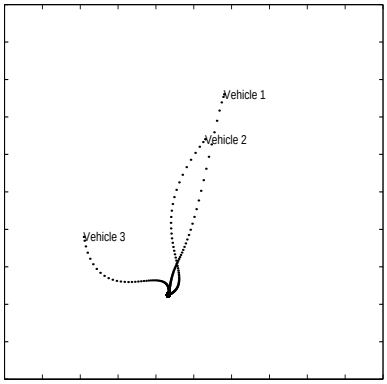

Figure 2.3: Vehicles converging to the target. The gain matrix is scaled up by a factor of 5 as compared to Figure 2.2.

avoid other pedestrians, who are also seeking to complete tasks. In this context, and in the context of many other distributed task dynamics that occur in physical spaces, *collision avoidance*—i.e., prevention of collisions between agents through evasive action—is a required component of the task. In this article, we explore collision avoidance among a network of distributed agents with double-integrator dynamics that are seeking to complete a formation task. Like pedestrians walking on a street, agents in our double-integrator network achieve collision avoidance by taking advantage of the multiple directions of motion available to them.

We now specifically consider networks of agents that move in the plane and seek to converge to a fixed formation. We develop a control strategy that uses localized sensing information to achieve collision avoidance, in addition to using remote sensing to achieve the formation task. We show that static and dynamic controllers that achieve both formation stabilization and collision avoidance can be designed, under rather general conditions on

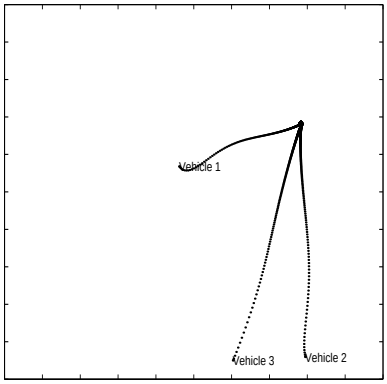

Figure 2.4: Vehicles converging to the target when the HVG controller parameter a is increased from 1 to 3.

the desired formation, the size of the region around each agent that must be avoided by the other agents, and the remote sensing topology,

We strongly believe that our study represents a novel viewpoint on collision avoidance, in that avoidance is viewed as a localized sub-task within the broader formation stabilization task. Our approach is in sharp contrast to the potential function-based approaches of, e.g., [4], in which the mechanism for collision avoidance also simultaneously specifies or constrains the final formation.

We recently became aware of the study of [9], which—like our work—seeks to achieve collision avoidance using only local measurements. We believe that our study builds on that of [9], in the following respect: we allow for true formation (not only swarming) tasks, which are achieved through decentralized sensing and control rather than through the use of a set of potential functions. To this end, our collision avoidance mechanism uses a combination of both gyroscopic and repulsive forces, in a manner that leaves the formation

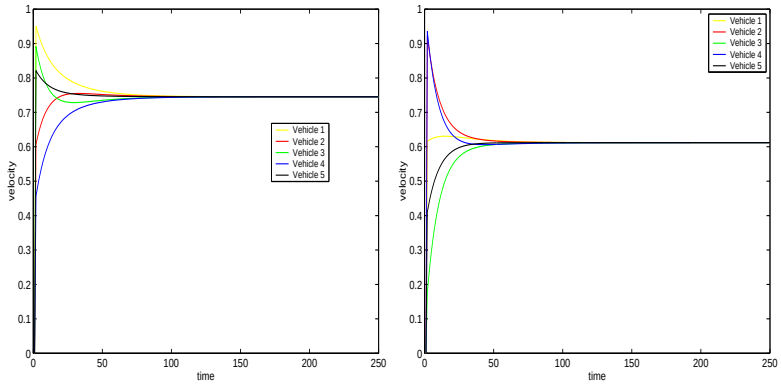

Figure 2.5: We show alignment of x-direction velocities in two simulations. Notice that the final velocity is different for the two simulations.

dynamics completely unaffected in one direction of motion. This approach has the advantage that the collision avoidance and formation tasks can be decoupled, so that rigorous analysis is possible, even when the task dynamics are not specified by potentials (as in our case).

We believe that our study of collision avoidance is especially pertinent for the engineering of multi-agent systems (e.g., coordination of autonomous underwater vehicles), since the task dynamics required for the design are typically specified globally and independently of the collision avoidance mechanism. Our study here shows how such a system can be stabilized to a pre-set desired formation, while using local sensing information to avoid collisions.

2.6.1 Our Model for Collision Avoidance

In our discussion of collision avoidance, we specialize the double-integrator network model to represent the dynamics of agents moving in the plane, whose accelerations are the control inputs. We also augment the model by specifying collision avoidance requirements and allowing localized sensing measurements that permit collision avoidance. We find it most convenient to develop the augmented model from scratch, and then to relate this model to the double-integrator network in a manner that facilitates analysis of formation stabilization with collision avoidance. We call the new model the *plane double-integrator network* (PDIN).

We consider a network of n agents with double-integrator dynamics, each moving in the Euclidean plane. We denote the $x-$ and $y-$ positions of agent i by r_{ix} and r_{iy}, respectively, and use $\mathbf{r}_i \triangleq \begin{bmatrix} r_{ix} \\ r_{iy} \end{bmatrix}$[5]. These n agents aim to complete a formation stabilization task; in particular, they seek to converge to the coordinates $(\overline{r}_{1x}, \overline{r}_{1y}), \ldots, (\overline{r}_{nx}, \overline{r}_{ny})$, respectively. The agents use measurements of each others' positions and velocities to achieve the formation task. We assume that each agent has available two position and two velocity observations, respectively. In particular, each agent i is assumed to have available a vector of position observations of the form $\mathbf{y}_{pi} = C_i \sum_{j=1}^{n} g_{ij}\mathbf{r}_j$, and a vector of velocity observation $\mathbf{y}_{vi} = C_i \sum_{j=1}^{n} g_{ij}\dot{\mathbf{r}}_j$. That is, we assume that each agent has available two position measurements that are averages of its neighbors' positions and two velocity measurements

[5]Notice that agents in our new model can have vector state; because we need to explicitly consider the dynamics of the agents in each coordinate direction, it is convenient for us to maintain vector positions for the agents rather than reformulating each agent as two separate agents.

that are averages of its neighbors' velocities. Each of these measurements may be weighted by a **direction-changing matrix** C_i, which models that each agent's observations may be made in a rotated frame of reference. We view the matrix $G = [g_{ij}]$ as specifying a **remote sensing architecture** for the plane double-integrator network, since it specifies how each agent's observations depend on the other agents' states[6]. Not surprisingly, we will find that success of the formation task depends on the structure G, as well as the structure of $C \triangleq \begin{bmatrix} C_1 & 0 & \dots \\ & \ddots & 0 \\ \dots & 0 & C_n \end{bmatrix}$.

Our aim is to design a decentralized controller that places the agents in the desired formation, while providing collision avoidance capability. Collision avoidance is achieved using local measurements that warn each agent of other nearby agents. In particular, we assume that each agent has a circle of radius q about it, called the **repulsion ball** (see Fig. 2.6). **Collision avoidance** is said to be achieved if the agents' repulsion balls never intersect. (We can alternately define collision avoidance to mean that no agent ever enters another agent's repulsion ball, as has been done in [4]. The analysis is only trivially different in this case; we use the formulation above because it is a little easier to illustrate our controller graphically.) We also assume that each agent has a circle of radius $t > 2q$ about it, such that the agent can sense its relative position to any other agent in the circle. We call these circles the **local sensing balls** for the agents (see Fig. 2.6). That is, agent i has available the

[6]Two remarks should be made here. First, we note that our notation for the remote sensing architecture differs from the notation for the sensing architecture presented before, because agents are considered to have vector states rather than being reformulated as multiple agents with scalar statuses. Second, we note that the analyses described here can be adopted to some more general sensing observations (e.g., with different numbers of measurements for each agent), but we consider this structured formulation for clarity.

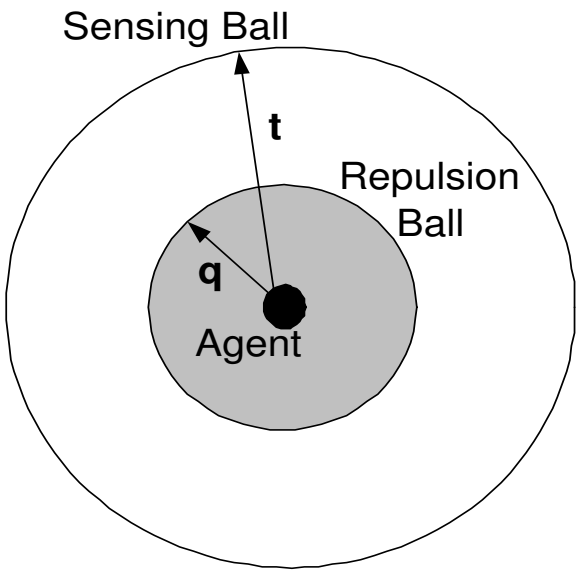

Figure 2.6: The repulsion ball and local sensing ball are illustrated.

observation $r_i - r_j$, whenever $||r_i - r_j||_2 < t$. The agents can use these **local warning measurements**, as well as the position and velocity measurements, to converge to the desired formation while avoiding collisions.

We view a plane double-integrator network as being specified by the four parameters (G, C, q, t), since these together specify the remote sensing and collision avoidance requirements/capabilities of the network. We succinctly refer to a particular network as $PDIN(G, C, q, t)$.

2.6.2 Formation Stabilization with Collision Avoidance: Static and Dynamic Controllers

We formulate controllers for PDINs that achieve both formation stabilization (i.e., convergence of agents to their desired formation positions) and collision avoidance. In particular, given certain conditions on the remote sensing architecture, the size of the repulsion ball, and the desired formation, we are able to prove the existence of dynamic controllers that achieve both formation stabilization and collision avoidance. Below, we formally state and prove theorems giving sufficient conditions for formation stabilization with collision avoidance. First, however, we describe in words the conditions that allow formation stabilization with guaranteed collision avoidance.

In order to achieve formation stabilization with collision avoidance for a PDIN, it is requisite that the remote sensing topology permit formation stabilization. Using Theorem 2.2, we can easily show that a sufficient (and in fact necessary) condition for formation stabilization using the remote sensing architecture is that G and C have full rank. Our philosophy for concurrently assuring that collisions do not occur is as follows (Fig. 2.7). We find a vector direction along which each agent can move to its formation position without danger of collision. As agents move toward their formation positions using a control based on the remote sensing measurements, we apply a second control that serves to prevent collisions; the novelty in our approach is that this second control is chosen to only change the trajectories of the agents in the vector direction described above. Hence, each agent's motion in the orthogonal direction is not affected by the collision avoidance control, and the

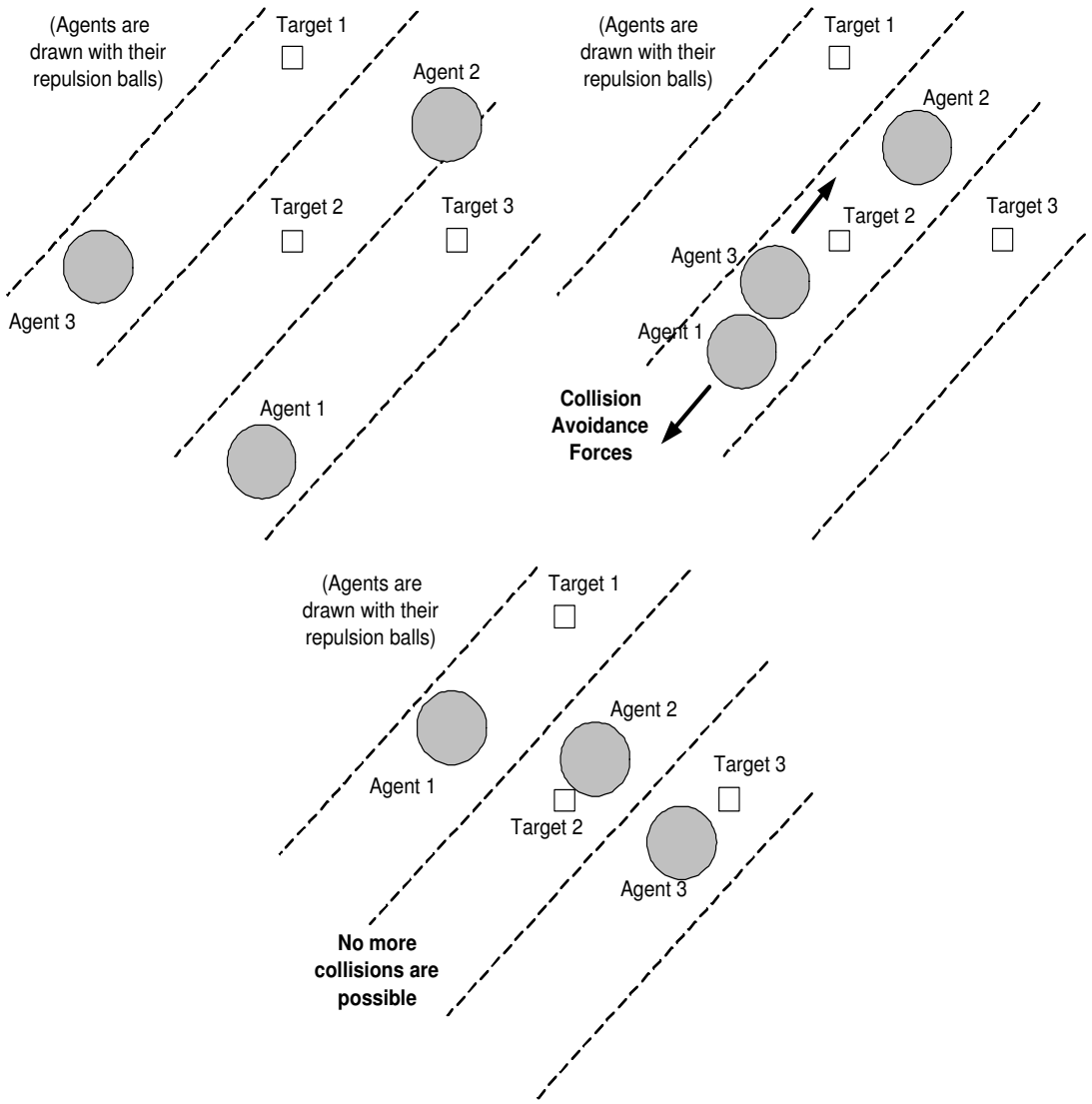

Figure 2.7: Our approach for formation stabilization with collision avoidance is illustrated, using snapshots at three time points.

component of their positions in this orthogonal direction can be guaranteed to converge to its final value. After some time, the agents are no longer in danger of collision, and so can be shown to converge to their formation positions. What is required for this approach to collision avoidance is that a vector direction exists in which the agents can move to their formation positions without danger of collision. This requirement is codified in the definitions below.

Definition 3 *Consider a* $PDIN(G, C, q, t)$ *whose agents seek to converge to the formation* $\bar{\mathbf{r}}$*, and consider a vector direction specified by the unit vector* (a, b)*. We shall call this direction* **valid***, if when each agent is swept along this direction from its formation position, the agents' repulsion balls do not ever intersect (in a strict sense), as shown in Fig. 2.7. From simple geometric arguments, we find that the direction* (a, b) *is valid if and only if* $min_{i,j}| - b(r_{ix} - r_{jx}) + a(r_{iy} - r_{jy})| > 2q$*. If* $PDIN(G, C, q, t)$ *has at least one valid direction for a formation* $\bar{\mathbf{r}}$*, that formation is called* **valid***.*

Theorem 2.9 $PDIN(G, C, q, t)$ *has a dynamic time-invariant controller that achieves both formation stabilization to* $\bar{\mathbf{r}}$ *(i.e., convergence of the agents' positions to* $\bar{\mathbf{r}}$*) and collision avoidance if* G *and* C *have full rank, and* $\bar{\mathbf{r}}$ *is a valid formation.*

Proof. The strategy we use to prove the theorem is to first design a controller, in the special case that $C = I$ and the valid direction is the x-direction $(1, 0)$. We then prove the theorem generally, by converting the general case to the special case proven first. This conversion entails viewing the PDIN in a rotated frame of reference.

To be more precise, we first prove the existence of a dynamic controller for the planar double integrator network $PDIN(G, I, q, t)$, when G has full rank and the direction $(1, 0)$ is valid. For convenience, let's define the **minimum coordinate separation** f to denote the minimum distance in the y direction between two agents in the desired formation. Note that $f > 2q$.

Let us separately consider the stabilization of the agents' y-positions and x-positions to their formation positions. We shall design a controller for the agents' y-positions, that uses only the remote sensing measurements that are averages of y-direction positions and velocities. Then we can view the agents' y-positions as being a standard double-integrator network with full graph matrix G. As we showed in our associated article, formation stabilization of this double-integrator network is possible using a decentralized dynamic LTI controller whenever G has full rank. Let us assume that we use this stabilizing controller for the agents' y-positions. Then we know for certain that the y-positions of the agents will converge to their formation positions.

Next, we develop a decentralized controller that determines the x-direction control inputs from the x-direction observations and the warning measurements. We show that this decentralized controller achieves stabilization of the agents' x-positions, and simultaneously guarantees collision avoidance among the agents. To do so, note that we can develop a decentralized dynamic LTI controller that achieves convergence of the agents' x-positions to their desired formation (in the same manner as for the y-direction positions). Say that

the stabilizing decentralized controller is

$$\dot{\mathbf{z}} = A_x\mathbf{z} + B_x\mathbf{y}_x, \tag{2.29}$$

$$\mathbf{u}_x = C_x\mathbf{z} + D_x\mathbf{y}_x,$$

where $\mathbf{y}_x$ are the x-direction observations (i.e., measurements of x-direction positions and velocities for each agent) , $\mathbf{z}$ is the state of the feedback controller, $\mathbf{u}_x$ are the x-direction inputs, and the coefficient matrices are appropriately structured to represent a decentralized controller. To achieve formation stabilization and collision avoidance, we use the following controller:

$$\begin{aligned} \dot{\mathbf{z}} &= A_x\mathbf{z} + B_x\mathbf{y}_x, \\ \mathbf{u}_x &= C_x\mathbf{z} + D_x\mathbf{y}_x + \begin{bmatrix} \sum_{j=1}^{n} g(\mathbf{r}_1 - \mathbf{r}_j) \\ \vdots \\ \sum_{j=1}^{n} g(\mathbf{r}_n - \mathbf{r}_j) \end{bmatrix}, \end{aligned} \tag{2.30}$$

where the non-linear input term $g()$, which uses the warning measurements, is described in detail in the next paragraph[7].

The non-linear term $g(\mathbf{r}_i - \mathbf{r}_j)$ acts to push away agent i from agent j whenever agent j is too close to agent i. Of course, it can only be non-zero if agent j is in the sensing ball of agent i. In particular, we define the function $g(\mathbf{r}_i - \mathbf{r}_j)$ to be non-zero for $2q \leq ||\mathbf{r}_i - \mathbf{r}_j||_2 \leq \epsilon$

[7]Since the right-hand side of the CL system is discontinuous, we henceforth assume that solutions to our system are in the sense of Filipov.

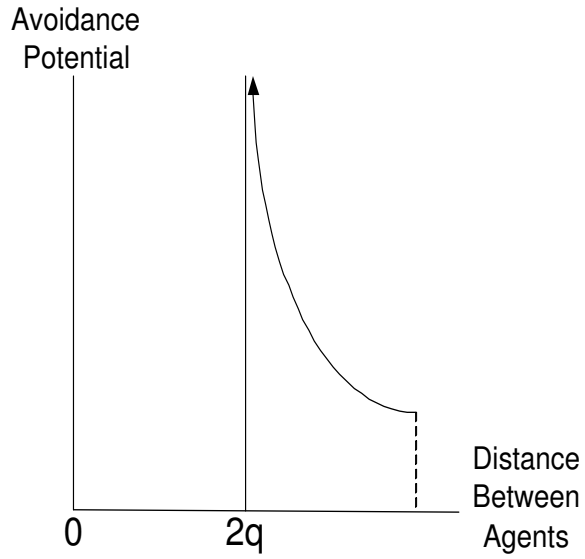

Figure 2.8: An example potential function for collision avoidance is shown.

and 0 otherwise, where ϵ is strictly between $2q$ and $min(f,t)$. Furthermore, we define $g(\mathbf{r}_i - \mathbf{r}_j)$ to take the form $sgn(r_{ix} - r_{iy})\widehat{g}(||\mathbf{r}_i - \mathbf{r}_j||_2)$, where $\widehat{g}(||\mathbf{r}_i - \mathbf{r}_j||_2)$ is a decreasing function of $||\mathbf{r}_i - \mathbf{r}_j||_2$ in the interval between q and ϵ, and $\int_{t=q}^{\epsilon} g(t)$ is infinite, for any ϵ. An example of a function $g()$ is shown in Fig. 2.8.

To prove that this controller achieves collision avoidance, let us imagine that two agents i and j are entering each others' local sensing balls. Now assume that agent i were to follow a path such that its repulsion ball intersected the repulsion ball of agent j. Without loss of generality, assume that $r_{ix} < r_{jx}$ at the time of intersection of the repulsion balls. In this case, agent i would have an infinite velocity in the negative-x direction (since the integral of the input acceleration would be infinite over any such trajectory). Thus, the movement of the agent would be away from the repulsion ball, and the agent could not

possibly intersect the ball. In the case where more than two agents interact at once, we can still show collision avoidance by showing that at least one agent would be moving away from the others at infinite velocity if the repulsion balls were to collide. Thus, collisions will always be avoided.

Next, let's prove that the x-positions of the agents converge to their desired formation positions. To do so, we note that there is a finite time T such that $|r_{iy}(t) - r_{jy}(t)| > \epsilon$ for all $t \geq T$ and for all pairs of agents (i, j), since the y-positions of the agents are stable and $(1, 0)$ is a valid direction. Thus, for $t \geq T$,the collision avoidance input is zero for all agents. Hence, the dynamics of the agents' x-positions are governed by the stabilizing controller, and the agents converge to the desired formation. Hence, we have shown the existence of a controller that achieves formation stabilization and collision avoidance.

We are now ready to prove the theorem in the general case, that is for a $PDIN(G, C, q, t)$ whose agents seek to converge to a valid formation $\bar{\mathbf{r}}$, and for which G and C have full rank. Without loss of generality, let us assume that (a, b) is a valid direction. We will develop a controller that stabilizes each agent's **rotated position** $\mathbf{s}_i = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \mathbf{r}_i$ and **rotated velocity** $\dot{\mathbf{s}}_i$. Stabilization of the rotated positions and velocities imply stabilization of the original positions and velocities, since the two are related by full rank transformations. It is easy to check that the rotated positions and velocities are the positions and velocities of a different PDIN. In the rotated frame, each agent i aims to converge

to the position $\bar{\mathbf{s}}_i = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \bar{\mathbf{r}}_i$, so the network seeks to converge to the formation $\bar{\mathbf{s}} \triangleq (I_n \otimes \begin{bmatrix} a & b \\ -b & a \end{bmatrix})\bar{\mathbf{r}}$. Also, each agent has available the observations $\mathbf{y}_{pi} = C_i \sum_{j=1}^{n} g_{ij}\mathbf{s}_j$ and $\mathbf{y}_{vi} = C_i \sum_{j=1}^{n} g_{ij}\dot{\mathbf{s}}_j$. Hence, in the rotated frame, the agents' positions are governed by $PDIN\left(G, C(I_n \otimes \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1}), q, t\right)$, where we implicitly assume that the agents will back-calculate the accelerations in the original frame of reference for implementation. The advantage of reformulating the original PDIN in this new frame of reference is that the new PDIN has a valid direction $(1, 0)$. Also, we note that a controller (that achieves formation stabilization to $\bar{\mathbf{s}}$ and collision avoidance) can be developed for $PDIN\left(G, C(I_n \otimes \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1}), q, t\right)$ whenever a controller can be developed for $PDIN\,(G, I, q, t)$, because we can simply pre-multiply (in a decentralized fashion) the measurements of the first PDIN by $C(I_n \otimes \begin{bmatrix} a & b \\ -b & a \end{bmatrix}^{-1})^{-1}$ to obtain the measurements for the second PDIN. Hence, from the lemma above, we can show formation stabilization with collision avoidance. Hence, we have converted $PDIN(G, C, q, t)$ to a form from which we can guarantee both formation stabilization and collision avoidance. □

2.6.3 Discussion of Collision Avoidance

To illustrate our analyses of formation stabilization with collision avoidance, we make a couple remarks on the results obtained in the above analysis and present several simulations of collision avoidance.

Existence of a Valid Direction We have shown that formation stabilization with collision avoidance can be achieved whenever there exists a valid direction—one in which agents can move to their formation positions without possibility of collision with other agents. For purpose of design, we can check the existence of a valid direction by scanning through all possible direction vectors, and checking if each is valid (using the test described in Definition 2.6.2). More intuitively, however, we note that the existence of a valid direction is deeply connected with the distances between agents in the formation, the size of the repulsion ball, and the number of agents in the network. More precisely, for a given number of agents, if the distances between the agents are all sufficiently large compared to the radius of the repulsion ball, we can guarantee existence of a valid direction. As one might expect, the required distance between agents in the formation tends to become larger as the repulsion ball becomes larger, and as the number of agents increases.

We note that existence of valid direction is by no means necessary for achieving collision avoidance; however, we feel that, philosophically, the idea of converging to a valid direction is central to how collision avoidance is achieved: extra directions of motion are exploited to prevent collision while still working toward the global task. Distributed agents,

such as pedestrians on a street, do indeed find collision avoidance difficult when they do not have an open (i.e., valid) direction of motion, as our study suggests.

Other Formation Stabilizers In our discussion above, we considered a dynamic LTI controller for formation stabilization, and overlayed this controller with a secondary controller for collision avoidance. It is worthwhile to note that condition needed for collision avoidance—i.e., the existence of a valid direction—is generally decoupled from the type of controller used for formation stabilization. For instance, if we restrict ourselves to the class of (decentralized) static linear controllers, we can still achieve collision avoidance if we can achieve formation stabilization and show the existence of a valid direction. In this case, our result is only different in the sense that a stronger condition on G is needed to achieve static stabilization.

Different Collision Avoidance Protocols Another general advantage of our approach is that we can easily replace our proposed collision avoidance mechanism with another, as long as that new protocol exploits the presence of multiple directions of motion to achieve both formation and avoidance. Of particular interest, protocols may need to be tailored to the specifics of the available warning measurements. For instance, if the warning measurements only flag possible collisions and do not give detailed information about the distance between agents, we may still be able to achieve collision avoidance, by enforcing that agents follow curves in the valid direction whenever they sense the presence of other agents. We leave it to future work to formally prove convergence for such controllers, but

we demonstrate their application in an example below.

Avoidance of Moving Obstacles Although we have focused on collision avoidance among controllable agents, our strategy can adapted to networks with uncontrollable agents (obstacles). Fundamentally, this adaptation is possible because only one agent is required to diverge from its standard trajectory to avoid a collision. Hence, collisions between controllable agents and obstacles can be prevented by moving the controllable agents in a valid direction. Some work is needed to specify the precise conditions needed to guarantee collision avoidance (e.g., we must ensure two obstacles do not converge, and that the obstacle does not constantly hinder stabilization). We leave this work for the future.

Simulation We illustrate our strategy for achieving both collision avoidance and formation stabilization using the examples below. Fig. 2.9 shows a direct implementation of the collision avoidance strategy developed in this article. The figure verifies that collision avoidance and formation are both achieved, but exposes one difficulty with our approach: the agents' trajectories during collision avoidance tend to have large overshoots, because we must make the repulsion acceleration arbitrarily large near the repulsion ball to guarantee that collisions are avoided.

A simple approach for preventing overshoot after collision avoidance is to apply a braking acceleration as soon as an agent is no longer in danger of collision. We choose this braking acceleration to be equal in magnitude and opposite in direction to the total acceleration applied during collision avoidance. Fig. 2.10 shows a simulation of collision avoidance,

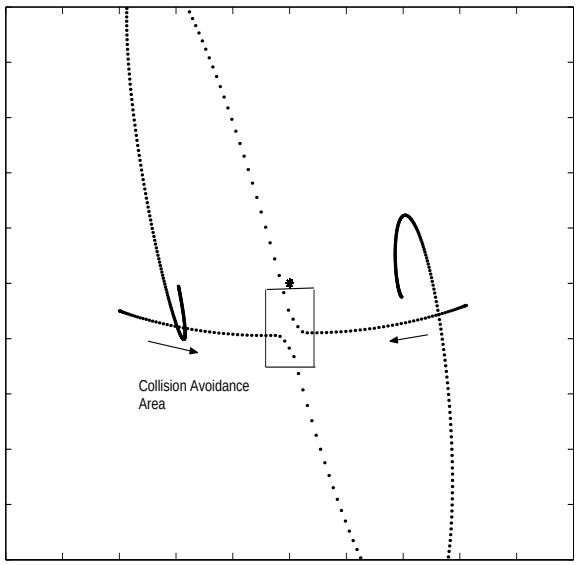

Figure 2.9: Formation stabilization with collision avoidance is shown.

when a braking force is applied after the collision avoidance maneuver. We note that the our proof for formation stabilization with collision avoidance can easily be extended to the case where braking is used.

Fig. 2.11 shows a more advanced approach to collision avoidance. In this example, the each agent is guided along curve in space, as soon it has detected the presence of another agent in its local sensing ball. Curve-following can allow the formulation of much more intricate, and optimized, collision avoidance protocols. Some more work is needed, however, to develop curve-following protocols that can be analytically shown to achieve formation and collision avoidance.

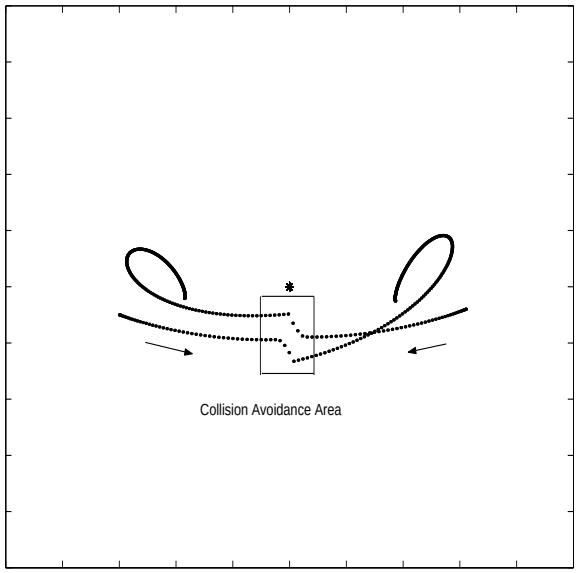

Figure 2.10: Another protocol for collision avoidance is simulated. Here, we have eliminated the overshoot after collision avoidance using a braking acceleration.

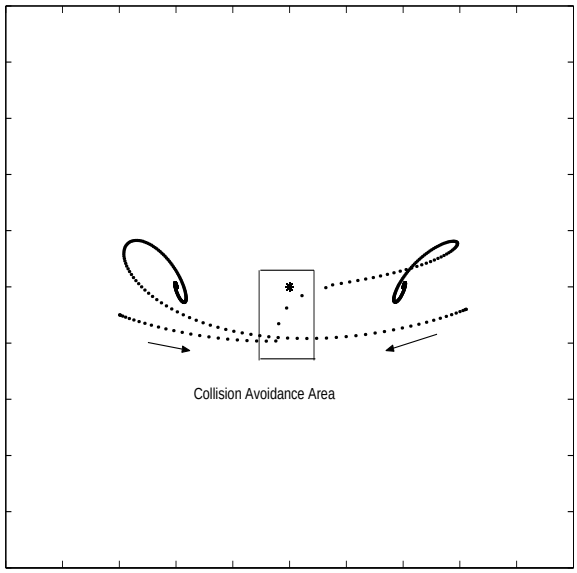

Figure 2.11: In this simulation, we achieve collision avoidance by guiding the agents along curves in space, once a potential collision is detected.

Chapter 3

A CONTROL-THEORETIC PERSPECTIVE ON THE DESIGN OF DISTRIBUTED AGREEMENT PROTOCOLS

We present a control-theoretic perspective on the design of distributed agreement protocols. First, we explore agreement-protocol analysis and design for a network of agents with single-integrator dynamics and arbitrary linear observations. One key contribution of our work is the analysis of protocols for networks with quite general observation topologies, including with multiple observations made by each agent. Another contribution is the development of techniques for agreement law design—i.e., for assignment of the dependence of the agreed-upon value on the initial states of the agents. Second, we explore agreement in a quasi-linear model with a stochastic protocol, which we call the controlled voter model. We motivate our study of this model, develop tests for whether agreement is achieved, and consider design of the agreement law. Finally, we provide some further thoughts regarding our control-theoretic perspective on agreement, including ideas for fault-tolerant protocol design using our approach.

3.1 Introduction

Agreement tasks among distributed agents are performed in several applications, such among processors in a computer network and among unmanned aerial vehicles. In this article, we approach the problem of agreement from a control-theoretic perspective. That is, we represent the agreement problem as a linear decentralized control problem for a network of sensing/communicating agents. Using this control-theoretic representation, we consider agreement in both a purely deterministic model and one with a stochastic protocol.

While agreement and agreement protocols are well-studied (see [10] for a thorough development), the control-theoretic approach is relatively new ([2] and [11] are two significant contributions) and, we believe, capable of providing fresh insight into agreement protocol design. Our control-theoretic approach allows us to make the following contributions:

- We show how agreement laws—functions that relate the agents' initial states to their final, agreed-upon value—can be designed: in particular, we show how to construct static agreement protocols (memoryless controllers) that achieve pre-specified linear agreement laws, for our models. In this respect, our work builds on the analysis of [2], which discusses how to check whether a static linear agreement protocol is successful (in the same model as our deterministic one), but does not consider agreement law design. Agreement protocol design using control-theoretic methods has also been considered in [11], though the focus there is on optimizing the convergence

rate rather than designing the agreement law.

- We develop agreement protocols for networks in which each agent makes multiple observations. We find that the multiple observations can provide greater flexibility in agreement law design. We also briefly discuss other issues related to agreement of linear networks, including fault-tolerant design and use of protocols with memory.

The remainder of the Chapter is organized as follows. In Section 3.2, we explore agreement in a network with agents having single-integrator dynamics and with a deterministic protocol. In Section 3.3, we motivate and introduce a network model with stochastic protocol which we call the *controlled voter model*, and then study agreement in this model. Finally, Section 3.4 briefly discusses further benefits (e.g., fault-tolerant protocol design) of our control-theoretic perspective.

3.2 Agreement in a Single-Integrator Network

We seek to identify whether or not agreement can be achieved among a group of communicating or sensing agents with single-integrator dynamics, using a static linear agreement protocol. Agreement protocols for single-integrator networks have also been considered in [2]. Our work differs from [2] in that we design protocols for achieving a desired agreement law, instead of simply checking whether a given protocol can achieve an agreement law such as average consensus. Our design-based philosophy is more similar in spirit to the approach of [11], but we focus on agreement law design rather than optimization of

the protocol for a given agreement law.

3.2.1 Model Formulation

We consider a network of n **agents**, where each agent i has a scalar **state** x_i. We assume that each agent's state is the integral of a **control input** u_i:

$$\dot{x}_i = u_i, \tag{3.1}$$

where the control input u_i is determined by a protocol (described precisely below) used by agent i. This single-integrator model for individual agents is representative of various physical systems (e.g., the velocity of a vehicle in a platoon is governed by an acceleration input according to a single-integrator model), and so constitutes a useful context for studying agreement. Also, our studies (as well as those of, e.g., [11] and [2]) highlight that agreement in networks of single-integrators is quite feasible, and hence that a single-integrator model may be a reasonable choice when an agent's dynamic update can be designed along with (or as part of) its protocol (e.g., in a computer network). For convenience, we also define a **state vector**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

and an **input vector**

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}.$$

Each agent i has available m_i observations, which are used by its protocol to determine its control input. The observations made by agent i may in general be arbitrary linear combinations of the state variables. That is, the vector $\mathbf{y}_i$ of observations made by agent i has the form

$$\mathbf{y}_i = G_i \mathbf{x}, \tag{3.2}$$

where each G_i is an $m_i \times n$ matrix. Because the matrix G_i specifies how agent i's observations are influenced by the other agents in the network, we call G_i the **graph matrix** for agent i. We find it convenient to define the **full graph matrix**

$$G = \begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix}.$$

We also find it useful to stack the observation vectors in a single vector:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix}. \tag{3.3}$$

In this notation, $\mathbf{y} = G\mathbf{x}$.

Thus, we have specified the state update and observation processes for our network of agents. We refer to the complete model as a **single-integrator network**.

3.2.2 Protocols, Agreement Protocols, and Agreement Laws

We are interested in developing protocols, or mappings between observations and inputs, that achieve agreement in a single-integrator network. In this article, we shall consider static linear protocols, as defined below:

A single-integrator network is said to be governed by (or to have) the static linear protocol $(K, \mathbf{z})$*, where* K *is the block-diagonal matrix*

$$K = \begin{bmatrix} \mathbf{k}_1' & & \\ & \ddots & \\ & & \mathbf{k}_n' \end{bmatrix},$$

and $\mathbf{z}$ *is an* n*-component vector, if each agent* i*'s input* u_i *is given by* $u_i = \mathbf{k}_i'\mathbf{y}_i + z_i$*.*

Notice that $\mathbf{k}_i$ is a column vector with m_i elements, so that the protocol K is a matrix of dimension $n \times \sum_{i=1}^{n} m_i$. In words, a single-integrator network is governed by (has) a static linear protocol, if each agent's input at each time is an affine combination of its observations at that time.

We are concerned with understanding whether a single-integrator network with full graph matrix G and protocol $(K, \mathbf{z})$ achieves agreement. We define agreement as follows:

A single-integrator network with graph matrix G and protocol $(K, \mathbf{z})$ is in agreement, if the states of all agents in the network converge to the same (but in general initial condition-dependent) value for all initial conditions. We refer to the value α reached by the agents, which is in general a function of the initial conditions, as the agreement value. A protocol $(K, \mathbf{z})$ that achieves agreement given a full graph matrix G is said to be an agreement protocol or valid agreement protocol.

We note that our definition for agreement is identical to that of [2].

At the most basic level, we are interested in identifying whether or not a protocol is an agreement protocol, for a single-integrator network with given graph matrix G. Once we know that a protocol is an agreement protocol, we aim to characterize the agreement value α for the protocol—in particular, to identify the dependence of α on the initial states of the agents. By doing so, we can aim to design protocols that achieve desired dependencies on the initial conditions. To this end, we define the notion of an agreement law:

Consider a single-integrator network with graph matrix G and agreement protocol $(K, \mathbf{z})$. Because the single-integrator network has linear dynamics, the agreement value for this network is an affine

function of the agents' initial states:

$$\alpha = \mathbf{p}' \begin{bmatrix} x_1(0) \\ \vdots \\ x_n(0) \end{bmatrix} + q \tag{3.4}$$

We refer to the pair $(\mathbf{p}, q)$ *as the agreement law for the network.*

The notion of an agreement law captures, in a general manner, the dependence of the agreed-upon value on the initial conditions of the single-integrator network.

Because the idea of an agreement law is central to our development, and because it is novel, we find it worthwhile to briefly discuss some examples. In [2] and [11], protocols that achieve *average consensus* are studied. These are agreement protocols for which the agreement value is the arithmetic average of the agents' initial conditions. In our terminology, a network that reaches average consensus is one that has the agreement law ($\mathbf{p} = \begin{bmatrix} \frac{1}{n}, & \ldots, & \frac{1}{n} \end{bmatrix}'$, $q = 0$). While average consensus is indeed a reasonable design goal for some networks, other design goals may be desired for some networks. For instance, a network may require that all agents converge to the initial value of one of the agents, say Agent 1. In our terminology, this design goal can be stated as the goal of achieving the agreement law ($\mathbf{p} = \begin{bmatrix} 1 & 0 & \ldots 0 \end{bmatrix}'$, $q = 0$). Yet other design goals may be agreement on an initial-condition independent value (agreement law $(\mathbf{0}, q)$ for some q), or agreement on a particular weighted average of agents' initial states (agreement law $(\mathbf{p}, 0)$ for some $\mathbf{p}$).

3.2.3 Test for Agreement and Identification of Agreement Laws

In this section, we specify tests for determining whether a particular static linear protocol is in fact an agreement protocol, and calculate the agreement laws when these agreement protocols are used. To specify these test, we first note that the state vector of a single-integrator network with graph matrix G and protocol $(K, \mathbf{z})$ satisfies the following differential equation:

$$\dot{\mathbf{x}} = KG\mathbf{x} + \mathbf{z}. \tag{3.5}$$

From this closed-loop system equation, we can straightforwardly identify whether or not the single-integrator network reaches agreement, and determine the agreement law if it does. We find it useful to differentiate between two cases, in specifying conditions for agreement. In particular, we specify conditions for agreement to a value that is independent of the initial conditions, and then separately specify conditions for agreement to an initial-condition dependent value. These conditions are described in the following theorem:

Theorem 3.1 *A single-integrator network with graph matrix G and protocol $(K, \mathbf{z})$ reaches agreement if and only if one of the following two conditions hold:*

- *all the eigenvalues of KG lie in the open left half plane (i.e., have strictly negative real parts), and $-(KG)^{-1}\mathbf{z} = \alpha\mathbf{1}$ for some α. In this case, the agreement law for the network is $(\mathbf{0}, \alpha)$. We refer to this scenario as Type 1 Agreement.*

- *KG has one eigenvalue of 0 with corresponding right eigenvector $\mathbf{1}$, the remaining eigenval-*

ues of KG are in the open left half plane, and $\mathbf{z} = 0$. In this case, the agreement law for the network is $(\mathbf{w}, 0)$ where $\mathbf{w}'$ is the left eigenvector of KG corresponding to the 0 eigenvalue. (In particular, we are referring to the left eigenvector $\mathbf{w}'$ such that $\mathbf{w}'KG = 0$ and $\mathbf{w}'\mathbf{1} = 1$. Henceforth, we refer to such a left eigenvector as a standard *left eigenvector*[1]*.) We refer to this scenario as Type 2 Agreement.*

Proof. First, we verify that the conditions postulated for Type 1 Agreement indeed lead to agreement. When the eigenvalues of KG are in the open left half plane (OLHP), it is well known that the state vector $\mathbf{x}$ of the single-integrator network converges. From Equation 3.5, it is clear that the limiting value of $\mathbf{x}$ is $-(KG)^{-1}\mathbf{z}$ for any set of initial conditions. Hence, agreement is achieved if $-(KG)^{-1}\mathbf{z}$ is a multiple of the vector of all ones (i.e., $\alpha\mathbf{1}$), and the corresponding agreement law is $(\mathbf{0}, \alpha)$. Thus, the conditions postulated for Type 1 Agreement are sufficient.

Next, we verify that the conditions postulated for Type 2 agreement are sufficient. From the modal decomposition of KG, it is clear that the state vector converges from any initial condition, and that its limiting value is given by $\mathbf{1}\mathbf{w}'\mathbf{x}(\mathbf{0})$. Hence, agreement is achieved, and the agreement value is given by $\mathbf{w}'\mathbf{x}(0)$, so $(\mathbf{w}, 0)$ is the agreement law.

To prove that agreement is achieved exclusively in these two scenarios, note that otherwise one of the following are true:

- KG has an eigenvalue in the ORHP.

[1] *Notice that the sum of the components of the agreement law is always* 1.

- KG has multiple eigenvalues on the $j\omega$-axis.
- KG has eigenvalues in the OLHP except a single eigenvalue at 0, but the corresponding right eigenvector is not $\mathbf{1}$.
- KG has eigenvalues in the OLHP except a single eigenvalue at the origin, but $z \neq 0$ (where (K, z) is the protocol).

It is easy to check from the modal decomposition of KG that agreement is not reached in any of these cases. In particular, if KG has ORHP eigenvalues, some state variables are non-convergent, and hence agreement is not reached. If KG has multiple $j\omega$ axis eigenvalues, either some state variables are non-convergent or the state variables do not converge to the same value for all initial conditions. Similarly, if the right eigenvalue of KG corresponding to the zero eigenvalue is not $\mathbf{1}$, the state variables cannot converge to the same value for all initial conditions. If KG has a single zero eigenvalue but $z \neq 0$, the state variables are either non-convergent or do not converge to the same value.

□

Consider a single-integrator network with full graph matrix

$$G = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix}$$

and protocol $(K = -I, z = \mathbf{0})$*. All but one of the eigenvalues of* KG *are negative, and the remaining eigenvalue equals zero. The right eigenvalue of* KG *corresponding to the zero eigenvalue is* $\mathbf{1}$*, and the corresponding left eigenvector is* $\mathbf{1}'$*. Hence, Type* 2 *Agreement is achieved, with a resultant agreement law of* $(p = \mathbf{1}, q = 0)$*. In this case, average consensus is achieved.*

3.2.4 Existence and Design of Agreement Laws

Now that we have developed tests for checking whether agreement is achieved by a given protocol, we are in a position to study the existence and design of protocols that achieve desired agreement laws. In this section, we present results that facilitate design of protocols to achieve such agreement laws. The results fall into one of two categories:

- Given a single-integrator network with full graph matrix G and a set of allowed agreement laws, we specify conditions on G for the existence of a protocol that achieves some agreement law within the set.
- Given a single-integrator network with full graph matrix G and a set of desired agreement laws, we specify conditions on G such that we can design protocols to

achieve all agreement laws within the set. That is, we specify conditions on G for *arbitrary assignability* of the agreement law within the set. As far as we know, our study of agreement law assignment is novel not only among control-theoretic studies but more generally in the computer science community.

Several of the conditions that we specify are of the following form: if G belongs to a certain class of matrices, then existence/assignability of the agreement law is guaranteed.

Let us first consider Type 1 Agreement Laws—i.e., the set of agreement laws $(0, \alpha)$, where $\alpha \in \mathcal{R}$. The following theorem presents a condition for both existence of protocol for achieving an agreement law in this set, and assignability of any arbitrary agreement law in the set.

Theorem 3.2 *Consider a single-integrator network with graph matrix G. A protocol exists such that an agreement law of the form $(0, \alpha)$, where $\alpha \in \mathcal{R}$, is achieved, if and only if there is a block-diagonal matrix K (of the proper dimensions) such that all eigenvalues of KG are in the OLHP. Furthermore, in this case, protocols can be designed to achieve any agreement law of the form $(0, \alpha)$.*

Proof. From Theorem 3.1, Type 1 agreement can be achieved only if the eigenvalues of KG are in the OLHP. Conversely, if we choose the protocol $(K, -\alpha KG\mathbf{1})$, then agreement is achieved, from Theorem 3.1. The resulting agreement law is $(0, \alpha)$, and hence we see that any Type 1 Agreement Law can be designed. □

Thus, we have developed a condition that can (in theory) be checked to determine whether an agreement law of the form $(0, \alpha)$ can be achieved. Notice that this theorem is essentially

a restatement of Theorem 3.1. However, we believe that the re-phrasing is valuable, because it expresses the result from a agreement law design perspective.

The following theorem provides a more applicable test on G, which can be used to determine whether an agreement law of the form $(0, \alpha)$ can be designed. The theorem is only applicable in the special case that G is square (i.e., the case that each agent makes a single observation).

Theorem 3.3 *Consider a single-integrator network with square graph matrix G. A protocol exists such that an agreement law of the form $(0, \alpha)$, where $\alpha \in \mathcal{R}$, is achieved, if there is a permutation of G such that all leading principal minors have full rank. Furthermore, in this case, protocols can be designed to achieve any agreement law of the form $(0, \alpha)$.*

Proof. In [12], we show that, if there is a permutation of G such that all leading principal minors have full rank, we can find diagonal K such that the eigenvalues of KG are in the OLHP. We refer the readers there for details. The required result thus follows from Theorem 3.2.

□

Next, let us consider design of protocols for Type 2 Agreement (i.e., that achieve agreement laws of the form $(\mathbf{p}, 0)$. We first study existence of a protocol that achieves some Type 2 Agreement Law.

Theorem 3.4 *Assume there exists an appropriately-dimensioned block diagonal matrix K such KG has a single zero eigenvalue with right eigenvector $\mathbf{1}$, and the remaining eigenvalues of KG*

are negative. Then we can design a protocol such that some agreement law of the form $(\mathbf{p}, 0)$ *can be achieved. In particular, the protocol* $(K, \mathbf{0})$ *achieves the agreement law* $(\mathbf{w}, 0)$*, where* $\mathbf{w}'$ *is the left eigenvector of* KG *corresponding to the zero eigenvalue.*

Proof. This theorem follows directly from Theorem 3.1. It is valuable as a restatement of Theorem 3.1 from a design perspective. □

Theorem 3.4 can be more easily applied, if its premises are rephrased explicitly in terms of G. We do this rewriting in two steps, in the following two theorems:

Theorem 3.5 *Assume that we can find vectors*

$\mathbf{v}_1 \in Ra(G_1^T), \ldots, \mathbf{v}_n \in Ra(G_n^T)$*, such that*

$$V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

has one zero eigenvalue with corresponding right eigenvector $\mathbf{1}$*, and that there is a diagonal matrix* K *such that all but one of the eigenvalues of* KV *are in the OLHP. Then we can design a protocol such that some agreement law of the form* $(\mathbf{p}, 0)$ *can be achieved.*

Proof. Define vectors $\widehat{\mathbf{k}}_1, \ldots, \widehat{\mathbf{k}}_n$, such that $\mathbf{v}_1 = G_1^T \widehat{\mathbf{k}}_1, \ldots, \mathbf{v}_n = G_n^T \widehat{\mathbf{k}}_n$, and consider the

matrix

$$M = \begin{bmatrix} k_1\widehat{\mathbf{k}}_1 & & \\ & \ddots & \\ & & k_n\widehat{\mathbf{k}}_n \end{bmatrix}. \tag{3.6}$$

It is easy to check that $MG = KV$ has one eigenvalue at 0 with corresponding right eigenvector $\mathbf{1}$, and all other eigenvalues are negative. Thus, from Theorem 3.4, we find that the protocol $(M, 0)$ achieves agreement, to the law $(\mathbf{w}, 0)$, where $\mathbf{w}'$ is the left eigenvector of KV corresponding to the zero eigenvalue. □

Theorem 3.6 *Assume that we can find vectors*

$\mathbf{v}_1 \in Ra(G_1^T), \ldots, \mathbf{v}_n \in Ra(G_n^T)$*, such that*

$$V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}$$

has one zero eigenvalue with corresponding right eigenvector $\mathbf{1}$*, and that there is a permutation of* V *whose leading principal minors have full rank. Then we can design a protocol such that some agreement law of the form* $(\mathbf{p}, 0)$ *can be achieved.*

Proof. This theorem is proved by invoking Theorem 3.6, and then applying a result from [12] (which was also applied to prove 3.3). We omit the details. □

In the remainder of this section, we describe conditions on G that guarantee existence of an agreement law within a particular **quadrant**, as well as conditions on G that guarantee that every agreement law within a quadrant can be achieved using some protocol. (We use the term **quadrant** of a Type 2 agreement law $(\mathbf{p}, 0)$ to refer to the sign pattern of the entries in $\mathbf{p}$. For instance, if all entries in $\mathbf{p}$ are positive, we refer to agreement law as lying in the first or positive quadrant.) For convenience, we assume that each agents makes only a single observation (i.e., that G is square) in these studies. The generalization to non-square G can be achieved in much the same way as for Theorem 3.5 and Theorem 3.6.

We are especially interested in identifying G for which all agreement laws within a quadrant can be achieved, because these are graph matrices for which essentially arbitrary agreement law design is possible[2]. That is, for such graph matrices we can decide on a desired dependence of the agreement value on the initial states of the agents (at least within a quadrant), and find a protocol that achieves this agreement law. Thus, we begin with this case.

We find it most enlightening to relate arbitrary assignment of the agreement law within a quadrant to the notion of D**-semistability** (e.g., [14]), so we begin with a definition of D-semistability.

The matrix A is said to be D-semistable if the eigenvalues of the matrix DA are in the closed left

[2]When we study whether "every" agreement law within a quadrant can be achieved, we implicitly consider only agreement laws $(\mathbf{p}, 0)$ such that $\mathbf{p}'\mathbf{1} = 1$. In other words, since the sum of the entries in $\mathbf{p}$ is 1 for any achievable agreement law, we implicitly assume that this constraint is met for any desired agreement law.

half plane and the eigenvalues of DA on the $j\omega$-axis are simple, for all positive diagonal D. [3]

We shall show that arbitrary assignment of the agreement law is possible when the graph matrix (or another matrix that is closely related with the graph matrix) is D-semistable. The advantage of characterizing arbitrary assignment using D-semistability is that many common classes of matrices are known to be D-semistable, so that we are immediately able to identify classes of matrices for which arbitrary assignment is possible. The relationship between D-semistability and arbitrary assignment is described in the following theorem:

Theorem 3.7 *Consider a single-integrator network in which each agent makes a single observation. We can develop a protocol to achieve any agreement law*[4] *of Type II (i.e., of form $(\mathbf{p}, 0)$) within some quadrant if and only if the following three conditions hold:*

- *The matrix ZG is D-semistable, where* $Z = \begin{bmatrix} -sign(g_{11}) & 0 & \dots \\ 0 & \ddots & 0 \\ \vdots & 0 & -sign(g_{nn}) \end{bmatrix}$.
- *The right eigenvector of G corresponding to the single eigenvalue at the origin is the vector $\mathbf{1}$. Also, the corresponding left eigenvector of G has strictly non-zero entries.*
- *ZG has no eigenvalues on the $j\omega$ axis other than the single eigenvalue at the origin.*

In this case, the quadrant in which any agreement law can be achieved is the one with sign pattern given by $\mathbf{w}'Z$, where $\mathbf{w}'$ is the left eigenvector of G corresponding to the zero eigenvalue.

[3] *Our notion of D-semistability differs from the linear algebra notion, in that we constrain eigenvalues on the $j\omega$ axis to be simple. We believe that this definition for D-semistability is germane in our context because internal stability of linear systems requires that imaginary axis eigenvalues are simple. We shall clarify classes of matrices that are D-semistable by our definition later in the article.*

[4] *As always, we implicitly consider only agreement laws whose components sum to one, since only these are possible.*

Proof. Let us assume that ZG is D-semistable and that the right eigenvector of G corresponding to the eigenvalue at the origin is $\mathbf{1}$. Let us also assume, temporarily, that DZG has no other eigenvalues on the $j\omega$-axis for any positive diagonal D. (We shall subsequently show that this final condition is equivalent to ZG having no eigenvalues on the $j\omega$-axis.) Now consider control of this single-integrator network using any protocol of the form KZ, where K is a positive diagonal matrix. Notice that all the eigenvalues of KZG except one are in the OLHP, and the remaining eigenvalue is a zero eigenvalue with a corresponding right eigenvector $\mathbf{1}$. Hence, from Theorem 3.1, KZ is indeed a valid agreement protocol. The agreement law for this protocol is $\mathbf{w}'Z^{-1}K^{-1}$, where $\mathbf{w}'$ is the left eigenvector of G corresponding to the zero eigenvalue. Thus, since for any positive diagonal K, KZ constitutes a valid agreement protocol, we see that the agreement law can be placed anywhere in the same quadrant as $\mathbf{w}'Z^{-1}$, or equivalently $\mathbf{w}'Z$. In particular, say that we wish to achieve the agreement law $\mathbf{v}$, where $sign(v_i) = sign(w_i)sign(z_{ii})$ for each i. By choosing $k_i = z_{ii}\frac{w_i}{v_i}$, we can achieve this agreement law.

So far, we have assumed that G is structured so that DZG has only a single $j\omega$-axis eigenvalue (at the origin) for any positive diagonal D. Let us now show that this condition is equivalent to the condition that ZG has only a single $j\omega$-axis eigenvalue. To do so, consider the number of $j\omega$-axis eigenvalues of $\widehat{D}ZG$, where $\widehat{D}$ is a particular positive diagonal matrix. We shall prove that this number remains the same for any positive diagonal $\widehat{D}$. First, note that the D-semistability condition on ZG guarantees that all $j\omega$-axis eigenvalues of $\widehat{D}ZG$ are simple. Now consider the eigenvalues of $(\widehat{D}+\Delta)ZG$, where Δ is a small perturbation. Say that this perturbation causes one of the $j\omega$-axis eigenvalues of $\widehat{D}ZG$ to move

into the OLHP. However, if the perturbation is sufficiently small, then the perturbation $-\Delta$ on $\widehat{D}$ will necessarily drive one of the $j\omega$ axis eigenvalues of $\widehat{D}ZG$ into the ORHP, since the $j\omega$-axis eigenvalues are simple. This is impossible, and hence for all sufficiently small perturbations, the number of $j\omega$-axis eigenvalues of $(\widehat{D}+\Delta)ZG$ is the same as the number of $j\omega$-axis eigenvalues of $\widehat{D}ZG$. By induction, we see that number of eigenvalues of DZG on the $j\omega$ axis is the same for all positive diagonal D. Hence, if we know that G has only a single $j\omega$-axis eigenvalue (at the origin), we can be sure that DZG has only a single $j\omega$-axis eigenvalue (at the origin) for any positive diagonal D. Thus, we have proved sufficiency of the conditions above for assignability to a quadrant.

Next, we prove that the three conditions are in fact necessary for arbitrary agreement law placement. Let us consider two cases:

1. If either ZG is not D-semistable, G does not have an eigenvalue at the origin with right eigenvector $\mathbf{1}$, or ZG has some eigenvalues on the $j\omega$-axis other than the single eigenvalue at the origin, then there is at least one positive diagonal $\widehat{K}$ such that is not a valid agreement protocol. Then the agreement law $\mathbf{w}'Z^{-1}K^{-1}$ cannot be achieved by any protocol. Hence, arbitrary assignment of the agreement law to this quadrant is not achieved. It is also easy to check that all agreement laws in another quadrant cannot possibly be achieved by considering the diagonal entries of ZG (see, e.g., [14] for the relevant argument).

2. If the left eigenvector of G corresponding to the zero eigenvalue at the origin has null components, then some components of the agreement law are necessarily zero.

Hence, arbitrary assignment of the agreement law to a quadrant is not possible.

Hence, the theorem is proven.

□

Although the conditions required for arbitrary assignment of the agreement law in a quadrant seem unwieldy, they can straightforwardly be checked because they are phrased directly in terms of the graph matrix G. In particular, the following steps can be followed to identify whether the conditions for arbitrary agreement are met:

1. D-semistability of ZG can be verified by determining that ZG belongs to one of several well-known classes of matrices. These classes of matrices are discussed in some detail below.

2. The remaining conditions can be checked through eigenanalysis of ZG.

The procedure above highlights that D-semistability of the graph matrix must be determined to check whether arbitrary assignment is possible. Unfortunately, there is no systematic procedure for checking D-semistability of a matrix. Luckily, however, there are several broad classes of matrices whose members are known to be D-semistable and also can be easily identified. We list several such classes of matrices, briefly describing techniques for determining whether a matrix is a member of each class as needed. We also illustrate the relationships between these classes of matrices in Figure 3.2.4. We omit the

justifications that matrices in these classes are D-semistable; the reader is referred to [14] for these details.

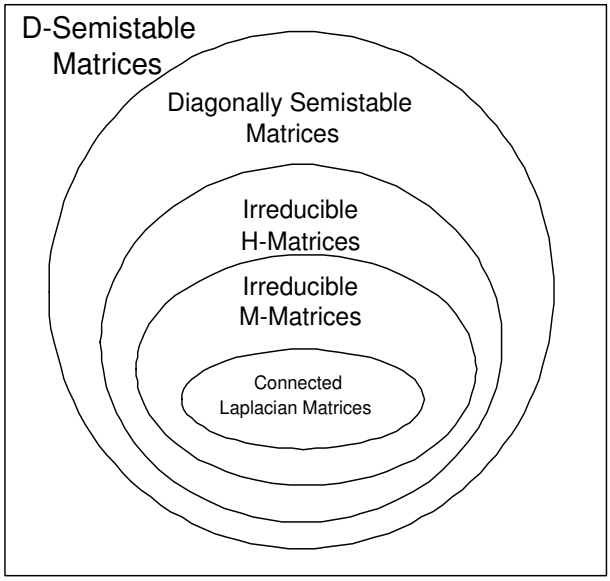

Figure 3.1: A Venn diagram of some classes of D-stable matrices is shown.

- Matrices that are **diagonally semistable** are also D-semistable. The matrix A is said to be diagonally semistable if there exists a positive diagonal matrix D such that $A^T D + DA$ is positive semi-definite. Optimization machinery has been used to develop a test for whether a matrix is diagonally semistable (see [13]).

- If $-A$ is an irreducible H-matrix with nonnegative diagonal entries, then A is diagonally semistable, and hence D-semistable. H-matrices are a fairly straightforward generalization of the class of M matrices (see below).

- If $-A$ is an irreducible M-matrix with nonnegative diagonal entries, then $-A$ is an

irreducible H matrix with nonnegative diagonal entries, and so is diagonally semistable and hence D-semistable. Recall that an M-matrix is one with non-negative principal minors and non-positive off-diagonal entries.

- If $-A$ is an irreducible Laplacian matrix, then $-A$ is an irreducible M matrix, and hence D-semistable. When $-A$ is an irreducible Laplacian matrix, all the other conditions of Theorem 3.7 also hold. Hence, arbitrary assignment of the agreement law to a quadrant (in particular, the all-positive quadrant) is possible.

Again consider

$$G = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix}.$$

Since G is an irreducible Laplacian matrix. ZG is the negative of an irreducible Laplacian matrix, and hence all premises of Theorem 3.8 hold. Thus, arbitrary placement of the agreement law in the first quadrant is possible. For instance, say that we wish to place the agreement law at $\mathbf{p}' = \begin{bmatrix} \frac{1}{12} & \frac{1}{12} & \frac{1}{4} & \frac{1}{4} & \frac{1}{3} \end{bmatrix}$. Noting that the left eigenvector of G is $\mathbf{w}' = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}$, we can

choose the protocol $KZ = \begin{bmatrix} z_{11}\frac{w_1}{v_1} & & \\ & \ddots & \\ & & z_{nn}\frac{w_n}{v_n} \end{bmatrix} = \begin{bmatrix} -\frac{12}{5} & & & & \\ & -\frac{12}{5} & & & \\ & & -\frac{4}{5} & & \\ & & & -\frac{4}{5} & \\ & & & & -\frac{3}{5} \end{bmatrix}$ *to achieve the desired agreement law. Of course, any multiple of this matrix can also be used as the protocol to achieve the desired agreement law.*

We conclude this section with a test for whether some agreement law within a specified quadrant can be achieved using a valid agreement protocol. We present this result without proof, since the proof is fundamentally similar to those of Theorems 3.4 and 3.7. More precisely, we draw on a result of [14] regarding existence of a positive diagonal D such that such that the product of D with a matrix A is stable. Our analysis is summarized in the following theorem:

Theorem 3.8 *Consider a single integrator network with square graph matrix G, and say that we wish to see whether some agreement law with the same sign pattern as a particular n-component vector $\mathbf{v}$ can be achieved. Assume that G has a zero eigenvalue, with corresponding right eigenvector $\mathbf{1}$ and left eigenvector $\mathbf{w}'$. (We assume that the entries of $\mathbf{w}'$ are non-zero; agreement within a quadrant is impossible if they are not.) Also, define a diagonal matrix* $Z = \begin{bmatrix} \frac{sign(w_1)}{sign(v_1)} & & \\ & \ddots & \\ & & \frac{sign(w_n)}{sign(v_n)} \end{bmatrix}$. *Then we can find some agreement protocol such that the agreement law has the same sign pattern as $\mathbf{v}$ if there is a permutation of ZG for which all leading principal minors of order less than n are*

positive.

We note that Theorem 3.8 and Theorem 3.6 are closely related. In particular, if the premise for Theorem 3.6—that there is a permutation of G such that the first $n-1$ leading principal minors have full rank—is satisfied then we can necessarily find a diagonal matrix Z with entries of ± 1 on the diagonal, such that the first $n-1$ leading principal minors of ZG are positive. Thus, we verify that there is a quadrant in which we can place the agreement law, as we would expect.

3.3 Agreement in a Controlled Voter Model

Second, we discuss the design of agreement protocols in the context of a quasi-linear discrete-time, discrete-state stochastic model, which we call the *controlled voter model*. Our motivations for studying agreement in a discrete-state and stochastic model are threefold: first, agreement among agents with discrete-valued states is required in several contexts, such as among jurors deciding on a defendant's guilt or several parallel process comparing the binary output of a computation. Second, protocols that are based on copying or choosing among several decisions—such as the one to be developed for our model—are easy to implement in some applications, since they often require minimal computation. Third, when agreement among agents with discrete-valued states is required, probabilistic decision-making often is necessary to reach agreement in an equitable manner, and hence stochastic models for protocols are relevant.

The controlled voter model provides a realistic context for studying agreement, yet is sufficiently structured to permit significant analysis of state dynamics and design of agreement laws. Essentially, the model is tractable because expected value of the state of the closed-loop system (the system when the protocol is applied) satisfies a linear recursion. Thus, we can re-phrase the problem of agreement as a linear control problem, such as for the single-integrator network.

3.3.1 Model Formulation and Connection to Literature

A controlled voter model comprises a network of n agents, each with a scalar state variable $x_i \in \{0, 1\}$ that is updated in discrete time. The state update of each agent is governed by a stochastic protocol: in particular, the state of agent i at time-$k+1$ is given by

$$x_i[k+1] = 1, \text{ with probability } u_i[k]$$

$$x_i[k+1] = 0, \text{ with probability } 1 - u_i[k],$$

where the **mean input** $u_i[k] \in [0, 1]$ is computed by the protocol from agent i's concurrent observations[5]. Notice that we define the protocol to include both the computation of $u_i[k]$ from the observations, and the stochastic determination of the next-state $x_i[k+1]$ based on $u_i[k]$. That is, the protocol uses the observations to set the probability that the next state will be 1, and then realizes the next state based on this probability. (This is in contrast to the single-integrator network, in which we view the relationship between the input and state

[5]We use the term *mean input* since the expected value of the "actual input" (which is the next-state) is $u_i[k]$.

as part of the intrinsic dynamics of the model rather than the protocol.) For convenience, we define a **state vector**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix},$$

and and a **mean input vector**

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}.$$

Each agent i makes m_i observations. Each observation is a weighted average of the concurrent state variables. That is, the observations made by agent i are given by

$$\mathbf{y}_i = G_i \mathbf{x}, \tag{3.7}$$

where the $m_i \times n$ graph matrix G_i is a **row-stochastic** matrix—i.e., one in which the elements in each row are non-negative and sum to 1. Notice that observations in our formulation can include states variables of single agents and weighted averages of multiple agents' states (e.g., of neighboring agents in a graph). Because we have enforced that G_i is row-stochastic, the entries in each vector $\mathbf{y}_i$ are necessarily in the interval $[0, 1]$ for any

state vector. For convenience, we again define a **full observation vector**

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix}.$$

a **full graph matrix**

$$G = \begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix}.$$

Our protocol calculates each agent's mean input u_i from its observation vector $\mathbf{y}_i$. We assume that this mapping is static and linear: the mean input u_i is determined as

$$u_i = \mathbf{k}_i' \mathbf{y}_i. \tag{3.8}$$

We further enforce that the entries in $\mathbf{k}_i'$ are non-negative and sum to 1, so that u_i is a weighted average of the entries in $\mathbf{y}_i$. This further constraint ensures that each input u_i is in the interval $[0, 1]$ at each time-step, as required. We note, further, that all u_i equal 0 at a given time if all agents' states at that time are zero, and that all u_i equal one if all agents' concurrent states are unity. For convenience, we codify the protocol using the

block-diagonal matrix

$$K = \begin{bmatrix} \mathbf{k}_1' & & \\ & \ddots & \\ & & \mathbf{k}_n' \end{bmatrix}.$$

Notice that $\mathbf{k}_i$ is a column vector with m_i elements, so that the **protocol matrix** K is a matrix of dimension $n \times \sum_{i=1}^{n} m_i$. The protocol matrix relates the observation vector to the mean input vector, as

$$\mathbf{u} = K\mathbf{y}. \tag{3.9}$$

A controlled voter model is specified completely by its graph matrix G and protocol matrix K, and hence we identify a particular model with its graph G and protocol K.

Our controlled voter model is a natural extension (in discrete time) of the *voter model* (equivalently, *invasion process*), originally introduced in [15] and [16] and studied in further detail in [17] and [18]. The stochastic realization of the next-state from the mean input inthese models is identical to ours; they are different in that the mean inputs are prescpecified linear combinations of state variables, rather than being specified as an observation operation followed by a decentralized control operation. Thus, although the closed-loop dynamics of the voter model and controlled voter model are identical, the voter model is viewed as representing a fixed process (that may or may not reach agreement) while the controlled voter model is a tool for designing protocols for agreement. It is our belief that the protocol-design perspective on the voter model is a valuable one. While the voter model may be too simplistic to represent many pre-existing systems, we are free to design

protocols as we see fit, and the voter model turns out to be a good choice because of its tractability and performance.

3.3.2 Definition of Agreement

The notion of agreement in the controlled voter model is essentially the same as for the single-integrator network: a model is in agreement if the states of all the agents asymptotically reach the same value. In the case of a controlled voter model that reaches agreement, this asymptotic value is either zero or one. The asymptotic value reached by the agents is in general stochastic—convergence to either all zeros and all ones is possible on any given trial. Thus, in contrast to the deterministic model, the agents' initial states do not exactly specify the value that is agreed upon by the agents; instead, these initial conditions specify the probability that agreement to the zero state or the unity state is achieved. With this difference in mind, we define the notion of an agreement value and agreement law in terms of the probabilities of reaching either asymptotic state. It is also worth mentioning that the notion of convergence to the same value is now a probabilistic notion (e.g., *convergence in probability* or *convergence with probability* 1, see [19] for instance). We formalize these notions in the following definitions.

A controlled voter model with graph G and protocol K is in agreement, if the states of all agents in the model converge to the same value (i.e., become identical) with probability 1*, for all initial conditions. We refer to the probability α that the agents reach the unity state (which, as our subsequent*

analysis shows, is a function of the initial conditions) as the agreement probability. A protocol K that achieves agreement for a given graph G is said to be an agreement protocol.

Consider a single-integrator network with graph matrix G and agreement protocol K. As shown in a following section, the agreement probability for this network turns out to be a linear function of the agents' initial states:

$$\alpha = \mathbf{p}' \begin{bmatrix} x_1[0] \\ \vdots \\ x_n[0] \end{bmatrix} \tag{3.10}$$

We refer to $\mathbf{p}$ as the agreement law for the network.

Sections 3.3.4 and 3.3.5 describe analysis of agreement protocols and design of protocols to achieve specific agreement laws, respectively. As with the deterministic model, the agreement law achieved by a given protocol, and the possibility for designing protocols, are strongly dependent on the structure of the graph matrix G.

3.3.3 Summary of Graph-Theoretic Concepts

Our study of protocol analysis and design for the controlled voter model turns out to be deeply related to some graph-theoretic concepts for matrices with non-negative entries. Hence, we review these graphical concepts before discussing protocol analysis and design. We refer the reader to, e.g., [20] for more details on graphical representations for matrices.

In particular, let us consider a square matrix $n \times n$ matrix A with non-negative entries. The **pictorial graph** of A, denoted $\Gamma(A)$, comprises n nodes or vertices. A directed edge (arrow) is drawn from vertex i to vertex j, if and only if A_{ij} is non-zero. The vertices and edges together constitute the graph. (Notice that we use the term *pictorial graph* rather than *graph* to distinguish from the graph matrix.)

We now introduce some concepts regarding pictorial graphs that are important for our analysis. Two vertices i and j in the pictorial graph are said to **communicate** if and only if there are a path of directed edges from i to j and from j to i. A set of vertices is called a **class**, if and only if the vertices in the set all communicate with each other, and none of the vertices in the set communicates with any vertex outside the set. A class is classified as **autonomous**, if there is no path from any vertex outside the class to a vertex in the class. A class is classified as **recurrent**, if there is no path from any vertex in the class to any vertex outside the class. All other classes are classified as **transient**. The concept of ergodicity is also important. A class is said to be **ergodic** if, given any sufficiently large length, we can find a path between any two vertices in the class of that length. Classes in a pictorial graph are illustrated in Figure 3.3.3.

The concepts regarding pictorial graphs summarized here have been considered in great detail in the study of Markov chains. We refer the reader to [20] for more about these concepts.

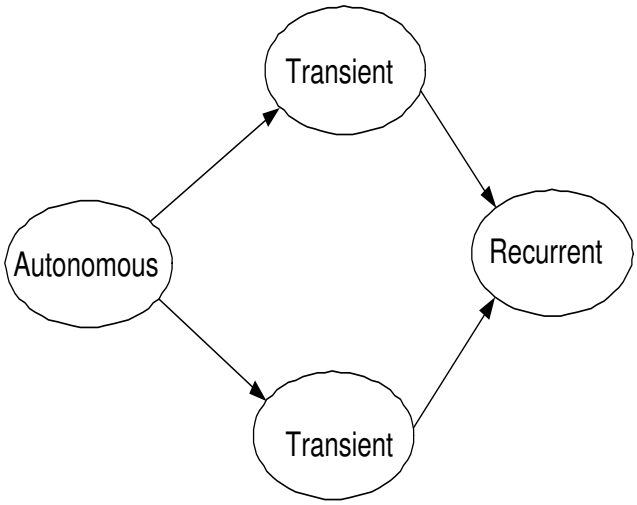

Figure 3.2: Classes in a pictorial graph are illustrated.

3.3.4 Analysis of Protocols

In this section, we consider controlled voter models with given graph G and protocol K, and 1) determine whether the model reaches agreement and, if so, 2) characterize the agreement law for the model. Thus, as with the single-integrator network, we develop conditions that can be used to check whether or not agreement is achieved, and to determine the agreement law achieved by a particular protocol. Because the graph matrix and protocol matrix for the controlled voter model are rather strictly constrained (each matrix is row-stochastic), it turns out that agreement is achieved for almost all G and K: unlike the deterministic model, there is no possibility for strictly unstable closed-loop dynamics, and achievement of agreement is instead solely contingent on whether a single status value can dominate the dynamics. Like the single-integrator network, we find that the agreement law for the controlled voter model can be determined through eigenanalysis of KG.

Our analysis of agreement in the controlled voter model is formalized in two theorems below:

Theorem 3.9 *A controlled voter model with graph G and protocol K reaches agreement, if and only if the pictorial graph $\Gamma((KG)')$ has a single autonomous class and that autonomous class is ergodic.*

Proof. We show that closed-loop dynamics of the controlled voter model are equivalent to the dynamics of a standard voter model with **network graph** $\Gamma((KG)')$ (see [18]). We then invoke a result given in [17] to prove the theorem.

The state update of the closed-loop model (i.e., the model with the protocol implemented) at time $k+1$ can be described using the following two-stage procedure:

- The n-component vector $b[k+1] \triangleq KGx[k]$ is computed. We note that each entry in $b[k+1]$ is in the interval $[0, 1]$.
- Each agent's next state is realized according to the corresponding probability in $b[k+1]$. That is, agent i's next state $x_i[k+1]$ is set to 1 with probability $b_i[k+1]$ and is set to 0 with probability $1 - b_i[k+1]$.

As discussed in [17], this two-stage procedure constitutes the state update of a voter model (there called the *binary influence model*) with network graph $\Gamma((KG)')$.

Once we have established that the closed-loop model can be viewed as a voter model, we

can immediately invoke the results of [17] to justify the theorem above. We refer the reader to [17] for a detailed justification of these results, but a broad outline is valuable to clarify the fundamental constraints on a controlled voter model that are required for agreement. Conditions for agreement are derived in [17] by viewing the entire state vector of a voter model as being governed by a finite-state Markov chain with 2^n possible states (since each agent takes on two states). Agreement is achieved if this **master Markov chain** has only two recurrent states, namely the state corresponding to each agent having individual state of 1 and having individual state of 0, respectively. The existence of only these two recurrent states can be connected with the structure of the network graph (in our case $\Gamma((KG)')$). In particular, existence of a single autonomous class that is ergodic guarantees that the two recurrent states can be reached from any initial condition (and hence that they are only recurrent states). Proof of necessity is similar.

□

The next theorem characterizes the agreement law for a controlled voter model with graph G and agreement protocol K. For convenience, we only consider the case that KG comprises a single ergodic class, although the theorem readily generalizes to the necessary and sufficient case considered in Theorem 3.9.

Theorem 3.10 *Consider a controlled voter model with graph G and protocol K, for which $\Gamma((KG)')$ comprises a single ergodic class. Notice that, from Theorem 3.3.4, K is an agreement protocol. The agreement law for this controlled voter model is $\mathbf{p}$, where $\mathbf{p}'$ is the left eigenvector of KG corresponding to its unity eigenvalue.*

Proof. The controlled voter model has an agreement probability that is a linear function of the initial states of the agents specifically because the expectation of its state vector satisfies a linear recursion. Thus, to characterize this agreement probability (and associated agreement law), we develop the recursion for the mean dynamics of the model. This recursion has already been developed in [17], but it is central to our studies and so we repeat the derivation.

Before we do so, let us clarify how the expected of the controlled voter model relate to its agreement probability. Note that the agreement probability is the limiting value of the probability that any one agent (and hence all agents) is in the unity state, given the initial states of all agents. However, since the state of each agent is an indicator, this probability is equivalent to the limiting value of the agent's expected state, given the initial states of all agents. Thus, we can compute the agreement probability by finding the limiting value of the expected state of an agent, conditioned on the agents' initial states.

A recursion for the expected state vector can be developed as follows:

$$\begin{aligned} & E(\mathbf{x}[k+1] \,|\, \mathbf{x}[0]) \\ & = E(E(\mathbf{x}[k+1] \,|\, \mathbf{x}[k]) \,|\, \mathbf{x}[0]) \\ & = E(KG\mathbf{x}[k] \,|\, \mathbf{x}[0]) \\ & KGE(\mathbf{x}[k] \,|\, \mathbf{x}[0]). \end{aligned} \tag{3.11}$$

Thus, we find that the conditional expectation for the state vector given the initial state sat-

isfies a discrete-time linear equation with state matrix KG. Since KG is a ergodic stochastic matrix, the limiting value of $E(\mathbf{x}[k] \mid \mathbf{x}[0])$ is given by $\mathbf{1}\mathbf{p}'\mathbf{x}[0]$, where $\mathbf{p}'$ is the left eigenvector of KG associated with the simple unity eigenvalue. Thus, the agreement probability is given by $\mathbf{p}'\mathbf{x}[0]$, and the theorem has been proved. □

3.3.5 Design of Agreement Laws

Just as in our deterministic model, design of the agreement law is feasible in the controlled voter model. That is, we can characterize the set of agreement laws that can be achieved by some protocol, for a given graph matrix. The ease with which agreement laws can be designed is a compelling feature of our formulation, since it allows us to design agreement protocols that weight each agent's initial state differently.

The design of agreement laws for the controlled voter model is simpler than for the single-integrator network: because stability is guaranteed for any protocol, the set of allowed agreement laws can be characterized solely by determining how the protocol K impacts the left eigenvector of KG corresponding to the unity eigenvalue[6]. Our characterization of the set of achievable agreement laws is phrased in terms of left eigenvectors of certain submatrices of G. We begin our development by defining appropriate notations for these submatrices and eigenvectors:

[6]To be precise, we further need to check that KG is ergodic, but we envision that ergodicity will be achieved naturally in many applications.

We shall consider $n \times n$ submatrices of G comprising single rows from each agent's graph matrix $G(i)$. In particular, let us consider a matrix whose jth row is row $i_j \in 1, \ldots, m_j$ of $G(j)$. We refer to this matrix as the reduced full graph matrix with observation list $\mathbf{i} = \{i_1, \ldots, i_n\}$, and use the notation $\widehat{G}_{\mathbf{i}}$ for the matrix. We also define the protocol $K_{\mathbf{i}}$ for observation list $\mathbf{i}$ to be the protocol for which each block-diagonal matrix (vector) $\mathbf{k}_j$ is an indicator vector with unity entry at entry i_j. We note that $\widehat{G}_{\mathbf{i}} = K_{\mathbf{i}}G$. If $\widehat{G}_{\mathbf{i}}$ is ergodic, it has a single unity eigenvalue. In such cases, we use the notation $\widehat{\mathbf{p}}'_{\mathbf{i}}$ for the corresponding left eigenvector. Finally, we note that there are $m = \prod_{i=1}^{n} m_i$ reduced full graph matrices (and corresponding unity eigenvectors).

We are now ready to present our main theorem concerning agreement law design:

Theorem 3.11*Consider a controlled voter model with graph matrix G, and assume that all reduced full graph matrices for this controlled voter model are ergodic. Then an agreement law $\mathbf{p}$ can be achieved using some protocol K, if and only if $\mathbf{p}$ can be written in the form*

$$\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \widehat{\mathbf{p}}_{\mathbf{i}}, \tag{3.12}$$

where the m coefficients $\alpha_{\mathbf{i}}$ are positive and sum to one. That is, an agreement law can be achieved if and only if it is a convex combination of the left eigenvectors of the full graph matrices corresponding to the unity eigenvalue.

Proof. We first prove that, if $\mathbf{p}$ can be written as a convex combination of the $\widehat{\mathbf{p}}_i$ (i.e., as $\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \widehat{\mathbf{p}}_{\mathbf{i}}$), then we can design a protocol to make $\mathbf{p}$ the agreement law. To do so, note that $\widehat{\mathbf{p}}_{\mathbf{i}}$ is the agreement law when the protocol $K_{\mathbf{i}}$ is used, or equivalently that $\widehat{\mathbf{p}}'_{\mathbf{i}} K_{\mathbf{i}} G = \widehat{\mathbf{p}}'_{\mathbf{i}}$.

Thus, from linearity, $(\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \widehat{\mathbf{p}}_{\mathbf{i}}' K_{\mathbf{i}}) G = \sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}_{\mathbf{i}}'$. Now consider $\mathbf{q}' = (\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \widehat{\mathbf{p}}_{\mathbf{i}}' K_{\mathbf{i}})$. This vector, which has $\sum_{i=1}^{n} m_i$ components, has the following characteristics:

- All entries in $\mathbf{q}$ are non-negative.
- The sum of the entries of $\mathbf{q}$ corresponding to each agent j is equal to equal to the jth entry of $\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}_{\mathbf{i}}'$, since the entries in each diagonal block (vector) of any protocol matrix $K_{\mathbf{i}}$ sum to unity.

From the two points above, we see that $\mathbf{q}'$ can be written as $(\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}_{\mathbf{i}}') \overline{K}$, for some valid protocol matrix $\overline{K}$. Thus, there exists a protocol matrix $\overline{K}$ such that $(\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}_{\mathbf{i}}')$ is a left eigenvector of $\overline{K}G$ corresponding to a unity eigenvalue. Further, $\overline{K}G$ is ergodic, since an average of ergodic matrices is ergodic. Hence, from Theorem 3.10, we see that the protocol $\overline{K}$ achieves the agreeement law $(\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}_{\mathbf{i}}')$, as desired.

The proof of necessity is based on showing that all possible protocols in fact correspond to particular agreement laws within the specified region. The proof is omitted due to its length.

□

A few further notes about application of the stochastic protocol design are worthwhile:

- To determine whether a given agreement law $\mathbf{p}$ can be achieved, one must check whether $\mathbf{p}$ lies in the convex hull defined by the vectors $\mathbf{p_i}$. Methods for checking are

well-known (see, e.g., [21]). These methods also serve to identify the coefficients $\alpha_{\mathbf{i}}$ that relate $\mathbf{p}$ to the $\mathbf{p}_i$.

- We have identified the set of achievable agreement laws, but have not yet shown how to choose a protocol to achieve a particular agreement law in this set. In fact, the procedure for choosing the protocol is implicitly contained in our sufficiency proof above. In particular, given a desired agreement law $\mathbf{p}$, we need to first compute $\mathbf{q}' = (\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \widehat{\mathbf{p}}'_{\mathbf{i}} K_{\mathbf{i}})$, where the $\alpha_{\mathbf{i}}$ are found as described in the first item in this list. Next, we need to choose $\overline{K}$ so that $\mathbf{q}' = (\sum_{\mathbf{i}} \alpha_{\mathbf{i}} \mathbf{p}'_{\mathbf{i}})\overline{K}$. This can be done easily, by normalizing[7] the components of $\mathbf{q}$ corresponding to find $\mathbf{k}_i$.

We consider a controlled voter model with three agents. The first and third agents each make two observations, while the second agent only makes a single observations. The full graph matrix for this example is the following:

$$G = \left[\begin{array}{ccc} 0.9 & 0.1 & 0 \\ 0.6 & 0.3 & 0.1 \\ \hline 0.2 & 0.5 & 0.3 \\ \hline 0 & 0.2 & 0.8 \\ 0.4 & 0.3 & 0.3 \end{array}\right] \tag{3.13}$$

[7] to a unity sum

We apply Theorem 3.11 *to identify the agreement laws that we can achieve using some protocol. The agreement laws that can be achieved are illustrated in Figure 3.3.5. We only show the first two components* p_1 *and* p_2 *of the agreement law on the plot, since the third component is determined explicitly from the first two.*

Let us say that we wish to achieve the agreement law $\mathbf{p}' = \begin{bmatrix} 0.43 & 0.33 & 0.24 \end{bmatrix}$. *From Figure 3.3.5, we see that this agreement law can be achieved. Applying the steps described above, we find that the protocol that achieves the desired agreement law is* $K = \begin{bmatrix} 0.31 & 0.69 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.33 & 0.67 \end{bmatrix}$.

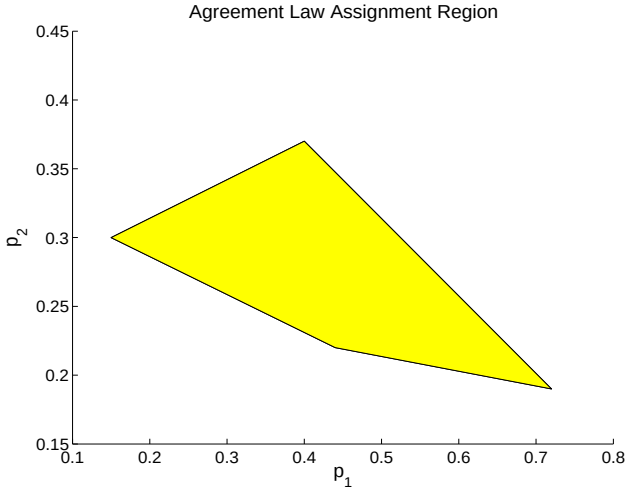

Figure 3.3: The agreement laws that can be achieved using some protocol are illustrated for an example controlled voter model with three agents. We only show the first two components p_1 and p_2 of the agreement law on the plot, since the third component is determined explicitly from the first two.

3.4 Further Directions

We have developed agreement protocols for two applicable dynamic models, focusing in particular on exposing the role of the communication network structure on protocol development. We believe our studies provide a compelling framework for understanding agreement in several dynamic systems. However, some further analyses can significantly expand the applicability of our framework. Here, we briefly list several directions of analysis that we are currently pursuing, along with some initial results.

Fault Tolerance Fault tolerance is often a requirement for an agreement protocol, especially in distributed computing applications (see, e.g., [10]). In these applications, protocols must be tolerant of random loss of communication, as well as purposeful miscommunications. Loss-of-communication faults may also be prevalent in autonomous vehicle applications (e.g., [22]) and other applications that involve transmission through a noisy environment.

We believe that our framework permits study of fault tolerance, because we allow for agents that make multiple observations, and because we explicitly consider design of agreement protocols. Thus, we can hope to design agreement protocols that are robust to common faults. For instance, for the single-integrator network, we can aim to develop a protocol that is robust to a single fault among a set of observations. Another common fault in a single-integrator network may be complete failure of an agent (i.e., loss of control of the agent as well as exclusion of observation of it by the other agents). In such a situa-

tion, our protocol should seek to achieve agreement among the remaining agents despite the lost observations. We expect that the ability to design such a fault-tolerant protocol is deeply connected to the notion of D-semistability, since D-semistable systems are robust to many changes in the graph matrix.

In a similar manner, we can construct a stochastic agreement protocol in order to minimize dependence on communications that are faulty or on potential agent failures. Further, by assuming a stochastic model for faults, we can characterize the expected dynamics of our network once a protocol is applied.

More Complicated Network Dynamics Another direction that we are currently pursuing is the development of agreement protocols for networks with more complex intrinsic dynamics. In particular, agents with double-integrator dynamics (rather than single-integrator dynamics) are common in mechanical systems, and so are of interest to us. For instance, if we are interested in achieving agreement among the positions (rather than velocities) of autonomous vehicles in a network, double-integrator dynamics must be considered since typically the accelerations of the vehicles are controlled. The techniques used to design agreement protocols in this article can readily be adapted for double-integrator networks, by meshing them with the analysis techniques discussed in [12]. We expect to discuss agreement in double-integrator networks in future work.

We have just begun to study agreement for networks in which agents' dynamics are intrinsically interconnected. Power networks and air traffic networks are examples of systems

in which agents' dynamics are dependent on each other. We believe that agreement in such networks can be analyzed by applying the decentralized control analysis of [8], as we have done for integrator networks in [12]. We are also interested in considering interdependent dynamics in the controlled voter model, by meshing uncontrollable interactions (i.e., standard voter model updates) with controllable dynamics.

Generalized Notions of Agreement In this article, we have solely considered agreement of the entire state vector of a network. In some applications, agreement of part of the state vector, or of functions of the state variables, may instead be required. The theory that we have developed here can be extended for analysis of agreement among linear functions of the state.

Chapter 4

A CONTROL-THEORETIC PERSPECTIVE ON DISTRIBUTED DISCRETE-VALUED DECISION-MAKING IN NETWORKS OF SENSING AGENTS

We address the problem of global sensor fusion for the purpose of distributed decision-making, from a control-theoretic perspective. In particular, we introduce a quasi-linear stochastic distributed protocol, using which a network of sensing agents can reach agreement in order to take a collective action. Using control-theoretic methods, we design the parameters of our protocol—which include weights in the local update rules used by the agents and a finite stopping time—to achieve agreement in a fair and rapid manner. We show analytically that the developed protocol achieves fair agreement with certainty in the noise-free case, and achieves fair agreement with high probability even in the presence of communication noise and assuming very little information storage capability for the agents. Our development is illustrated throughout with a canonical example motivated

by autonomous vehicle control.

4.1 Introduction

In many application areas, networks of agents with sensing capabilities are required to integrate their individual observations, and to make decisions or take actions collectively based on this fusion of information (e.g., [23]). The task of information fusion and decision-making in these networks is often complicated by the essential *decentralization* of the network dynamics and control: power/cost constraints and security/reliability concerns dictate that communication is short-range and, in some cases, that decision-making is done by the individual agents. For instance, swarms of military vehicles, such as unmanned aerial vehicles (UAVs), may need to collectively decide on the presence or abscence of a target, and take a decision on whether to destroy the target, in a totally distributed manner.

In this Chapter, we put forth the viewpoint that it is useful to consider the information-fusion and decision-making tasks of networks with sensing agents[1] jointly, as a decentralized *stabilization* or *agreement* problem. In pursuing this control-theoretic viewpoint, we propose a stochastic protocol for decision-making or agreement that is particularly suitable given typical characteristics of networks with sensing agents—i.e., networks with agents that operate at low power and with little memory, are subject to communication failures,

[1]We use the terminology "networks with sensing agents" rather than "sensor networks" because we envision our work as applicable to not only distributed and wireless sensor networks but also to networks the combine sensing with other dynamics (such as networks of UAVs). We also use the terminology to clarify that the networks that we consider do not necessarily contain a large number of agents.

and are seeking to make discrete-valued decisions such as whether or not to attack a target. By formulating the developed protocol in terms of the *influence model* (see [17, 18]), we are able to analytically determine the performance characteristics of our protocol, and to relate the protocol's performance with the network topology and fault characteristics.

Our work draws on, and aims to contribute to, both the literature on sensor networking and the control-theoretic study of agreement-protocol design. The article [23] contains a review of recent developments and challenges in distributed sensor networking. Several recent works have recognized the energy savings and scalability engendered by use of distributed algorithms (e.g., [24, 25, 26]) in sensor networks, providing broad motivation for our study. Of particular interest to us, the article [24] develops a distributed algorithm that uses information fusion for a specific application—event region detection—in a manner that suppresses faults and hence allows commitment to a decision and ensuing action. We also seek to achieve agreement among agents for the purpose of decision-making, but take the perspective that these agents begin with heterogeneous opinions about the issue to be decided on, not only because of faults but because of heterogeneity in their observation capabilities and perhaps intrinsic differences in their motives. Thus, in contrast to [24], we seek distributed algorithms for which the decided-on opinion of each agent takes into account the initial opinions of agents throughout the network rather than of only a set of nearby neighbors.

Agreement protocols are iterative algorithms using which a network of agents with initially-different opinions can reach a collective decision (e.g., [10, 2]) and hence take an action, in

a distributed manner. Agreement has been studied in the computer science literature for several years—see [10] for a thorough introduction. Recently, a control-theoretic viewpoint on agreement has been developed (e.g., [2, 11, 27]); this control-theoretic viewpoint is also deeply connected with the more general graph-theoretic study of stabilization in distributed dynamic systems (e.g., [1, 12]). The control-theoretic viewpoint has the advantage of facilitating development of graphical conditions for agreement, and of allowing characterization of protocol performance using linear system analysis techniques. Our previous work [27] is concerned with designing the dependence of the agreed-upon value on the initial opinions of the agents. This idea of *agreement-law design* in [27] plays a significant role in our current study, since we are interested in protocols that fairly[2] weight the initial opinions of the agents in making a collective decision.

We believe that several extensions of the control-theoretic studies on agreement and stabilization ([2, 11, 1, 27, 12]) are required for broad application to various networks with sensing capabilities:

- Agreement among *discrete-valued opinions* should be considered, since many networks with sensing capabilities perform (discrete-valued) detection or classification tasks, as well as binary decision-making. With the exception of our previous work [27], we do not know of any control-theoretic study of agreement that considers discrete-valued opinions. We focus on agreement upon discrete-valued opinions in

[2]We use the term *fair* loosely, to describe a rule that weights the initial opinions of the agents in coming up with the decided-upon value equitably, in the sense that the accuracy and motivation for each agent's initial opinion is properly incorporated.

this paper.

- A stochastic protocol is needed, in order to achieve agreement in a manner that equitably weights the initial opinions of the agents. As will be discussed further, a stochastic protocol can also be simpler to implement than some of the protocols described in the literature. To this end, we pursue a stochastic protocol in this work.

- In many networks with sensing capabilities, faults in the communication of information may occur (often because of severe limitations in transmitter power and communication bandwidth) and hence communication faults should be considered explicitly in our protocol design. Our analysis of the developed agreement protocol explicitly considers stochastic faults in communication.

- When the occurrence of faults is combined with limitations in the storage capabilities of agents and/or a requirement of fast decision-making, loss of the information contained in the initial opinions of the agents may result if too many iterations of the agreement protocol are used. Thus, a metric for stopping the agreement algorithm should be developed. In contrast to the previous control-theoretic studies of agreement (which define agreement in an asymptotic manner), we define a *stopping time* for our algorithm.

The stochastic protocol for agreement/decision-making that we propose here has a special quasi-linear structure, that permits significant characterization of its performance. In particular, the closed-loop dynamics of our network (i.e., the evolution of the opinions of the agents upon application of the agreement protocol) can be represented using a

specially-structured stochastic automaton known as the *influence model* ([17, 18]). The special tractability of the influence model then allows us to check the fairness of our protocol (and in turn to design the parameters of the protocol for fairness) using our results from [27], and to choose an appropriate stopping time for the protocol. Further, the influence model representation permits justification of the algorithm's success in a limiting case, and makes clear the connection between the the network topology and the performance of the agreement protocol.

The remainder of the Chapter is organized as follows. We conclude this section with a summary of the notation and terminology for graphs used in the article. In Section 4.2, we formulate our model for a distributed network of sensing agents, and introduce our stochastic strategy (protocol) for agreement or decision-making in this network. In Section 4.3, we use the influence model representation of the closed-loop model to evaluate the convergence properties and initial-condition dependence (and hence fairness) of our agreement protocol, and in the process identify a stopping time for our protocol. Finally, Section 4.4 contains discussion and evaluation of our protocol, and pursues a sensor networking-motivated application that combines our strategy with that of [24]. Examples are used throughout to illustrate our development.

4.1.1 Graph-Theoretic Notation

The analyses in this article are concerned with the communication and/or interaction topology of agents in a network, which we often find convenient to represent using a

graph. Thus, we need to briefly introduce the graph-theoretic notation used here. For us, a graph is an illustration of the pattern of non-zero entries in a square matrix. In particular, for an $n \times n$ matrix $G = [g_{ij}]$, the graph $\Gamma(G)$ associated with this matrix comprises n *vertices*, labeled $1, \ldots, n$. A *directed edge* is drawn from vertex i to vertex j if and only if g_{ij} is non-zero. We note that these directed edges include *self-loops*—i.e., an edge is drawn from vertex i back to itself if and only if g_{ii} is non-zero. As an example, the graph $\Gamma(G)$ associated with $G = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$ is shown in Figure 4.1.1

Our terminology for the connectibity characteristics of a graph are standard in the context of Markov chains (see [20]), but we briefly discuss the terminology here for convenience. There is said to be a **path** from vertex i to vertex j in a graph $\Gamma(G)$, if and only if there is a set of directed edges that leads from vertex i to vertex j in the graph. Two vertices i and j **communicate** of there are paths from i to j and from j to i in the graph. A graph is called **recurrent** if all pairs of vertices in the graph communicate. A recurrent graph is **ergodic** if, for any sufficiently large l, there is a path of length l (i.e., a path that traverses l not-necessarily-distinct edges) between any two vertices. In other words, a graph is ergodic if it is recurrent and also *aperiodic*. We note that a recurrent graph is ergodic whenever there is at least one agent with a self-loop.

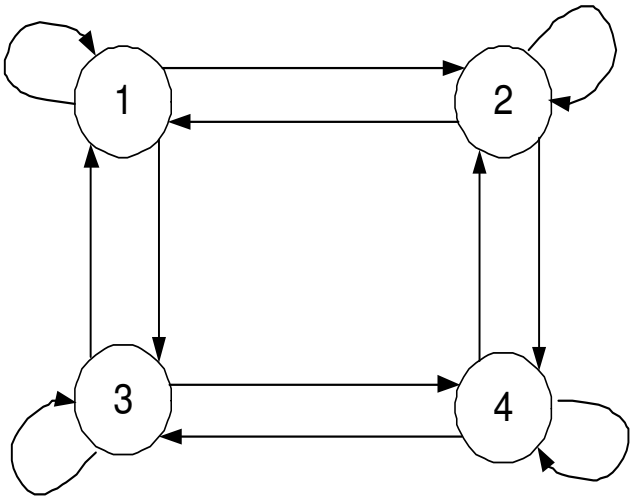

Figure 4.1: Illustration of a graph $\Gamma(G)$ associated with a particular $n \times n$ matrix. This graph is the adjacency graph for the autonomous vehicle control example discussed throughout the paper.

4.2 Formulation

We consider a network of n communicating agents, each of which has a discrete-valued **initial opinion** about a topic of interest (e.g., whether or not there is an enemy vehicle present; whether our next president should be Republican, Democrat, or Independent; what the color of a pomegranate is). It is incumbent upon the agents to agree on a single opinion over an interval of time, perhaps so that they can jointly take an action in response to this common opinion. Agents communicate and/or sense (possibly in a faulty manner) the current opinions of neighboring agents over time, and apply a stochastic protocol to update their opinions based on these dynamic observations. In this article, we assume that the agents update their opinions in discrete time. We use this discrete-time model for the sake of clarity, and because we believe that, in many applications, agents are governed by

a clock or at least can be meaningfully modeled at sampled time instances. Our protocol and its analysis can be readily adapted to continuous-time systems, using models such as those described in [28].

In the remainder of the section, we first formalize the notion of agreement and of an agreement law. We then describe a protocol that we claim can be used to achieve agreement and a desired agreeement law. Finally, we discuss the modeling of communication faults in our framework. This model is needed to appropriately design the parameters of our protocol and to evaluate the protocol, which is done in the subsequent section.

4.2.1 Opinions and Agreement: Definitions

Formally, we define each agent i to have a **opinion** $x_i[k]$ at discrete time k, where $x_i[0]$ denotes the **initial opinion** of agent i. Each agent's opinion at each time k is assumed to be in a set of m opinions, which we shall label $1, \ldots, m$ without loss of generality. We also find it convenient to define an indicator notation for $x_i[k]$; that is, we define $\mathbf{s}_i[k]$ to be an m-component $0-1$ indicator vector for the opinion of agent i at time k, i.e. a vector with entry $x_i[k]$ equal to 1 and all other entries equal to 0.

We are concerned with developing a protocol, using which the opinions of the agents can be brought into agreement in a rapid and equitable manner. Before developing and modeling the protocol, it is worth our while to carefully define the notion of agreement:

The network of n communicating agents is said to be in **agreement** *at time k, if the opinions $x_1[k], \ldots, x_n[k]$ of the n agents are identical at time k. We call the common opinion $a[k] \in 1, \ldots, m$ shared by the n agents the* **agreement value**.

We stress that, in contrast to [2], our definition for agreement considers the opinions of the agents at particular finite times rather than asymptotically.

Because our protocol is stochastic, and because we model communication/sensing in the network as being subject to faults, agreement is achieved in a probabilistic sense using our protocol. Hence, we define the notion of an agreement probability as a measure for the efficacy of our protocol.

The time-k agreement probability for the network is the conditional probability that the agents are in agreement at time k, given the initial opinions
$x_1[0], \ldots, x_n[0]$ *of the agents.*

We note that the agreement probability refers to the total probability that all agents have the same opinion (whatever that opinion might be), rather than the probability that the agents share a particular opinion.

Our aim is to achieve agreement among the agents in an equitable or fair manner. That is, when the agents reach agreement, we intend for the agreement value to appropriately reflect the initial opinions of the agents. Since our protocol is a stochastic one, the agreement value turns out to be stochastic. Thus, it is natural for us to consider the probability

of achieving each possible agreement value, and to characterize how these probabilities depend on the initial opinions of the agents. With this motivation in mind, we consider the following definition for an agreement law:

Consider a network that is in agreement at time k. We define the **agreement law** *as the conditional probability mass function for the agreement value $a[k]$ given the initial opinions of the agents $x_1[0], \ldots, x_n[0]$, i.e. the vector* $\begin{bmatrix} P(a[k] = 1 \,|\, x_1[0], \ldots, x_n[0]) \\ \vdots \\ P(a[k] = m \,|\, x_1[0], \ldots, x_n[0]) \end{bmatrix}$*. For short, we use the notation $P(a[k] \,|\, x_1[0], \ldots, x_n[0])$ for the agreement law.*

Notice that the agreement law is a function that maps the initial opinions of the agents to the probabilities of each possible agreement value, given that the network is in agreement.

These definitions are clarified using an example.

Example

Consider a network comprising $n = 4$ autonomous vehicles that are tasked with identifying whether an enemy vehicle is present and destroying the vehicle through cooperative action if it is present. Each vehicle i forms an initial opinion $x_i[0]$ about whether the target is absent (opinion 1) or present (opinion 2). For instance, $x_1[0] = x_3[0] = 1$, $x_2[0] = x_4[0] = 2$

denotes that vehicles 1 and 3 initially believe that the target is absent, while vehicles 2 and 4 initially believe that the target is present. These initial opinions can equivalently be written in indicator vector notation, e.g., as $\mathbf{s}_1[0] = \mathbf{s}_3[0] = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$ and $\mathbf{s}_2[0] = \mathbf{s}_4[0] = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix}$. Using the protocol described below, the agents update their opinions over time. The network is said to be in agreement at a time k if all the agents have the same opinion (either $a[k] = 1$ or $a[k] = 2$), and the agreement value is that common opinion. Given that the network is in agreement at time k, the agreement law is the conditional probability mass function for the agreement value given the initial opinions. For instance, one possible agreement law is

$$\begin{bmatrix} P(a[k] = 1 \,|\, x_1[0], \ldots, x_4[0]) \\ \\ P(a[k] = 2 \,|\, x_1[0], \ldots, x_4[0]) \end{bmatrix} = 0.25(\mathbf{s}_1[0] + \mathbf{s}_2[0] + \mathbf{s}_3[0] + \mathbf{s}_4[0]). \tag{4.1}$$

If the agents are in agreement and this agreement law holds, notice for instance that the following are true:

- If all the agents initially believe that the enemy is present, then they agree that the enemy is present at time k.
- More generally, if α of the 4 agents initially believe that the enemy is present, then at time k they agree with probability $\frac{\alpha}{4}$ that the enemy is present and agree with probability $1 - \frac{\alpha}{4}$ that the enemy is not present.

Thus, an agreement law of this form holds promise in equitably deciding on a common opinion based on the initial opinions of the agents. Our aim is to develop a protocol that

can achieve an equitable agreement law such as this one.

4.2.2 Agreement Protocol: Formulation

The agents in the network seek to reach agreement, by observing the current opinions of other agents and updating their opinions based on these observations. We assume that the agents are networked (distributed), in that each agent can only observe (through communication or sensing) the opinions of a subset of the other agents in the network. We define the set of agents whose opinions can be observed by agent i as the **neighborhood** of i, and label this set as $\mathcal{N}(i)$; we assume throughout that $\mathcal{N}(i)$ contains i, i.e., each agent can observe its own opinion. In general, observations made by each agent may be faulty; we shall explicitly model these communication/sensing faults in the next section. We find it convenient to associate a graph with the observation topology of the network. In particular, we define the $n \times n$ **adjacency matrix** $D = [d_{ij}]$ of the network to reflect the observation topology, as follows:

$$d_{ij} = 1, \quad j \in \mathcal{N}(i)$$

$$d_{ij} = 0, \quad \text{otherwise}$$

We refer to $\Gamma(D')$ as the adjacency graph of the network. We note that the adjacency graph captures the observation topology of the network, in that there is a directed edge from j to i if and only if agent i can observe the opinion of agent j. We expect that the adjacency graph is recurrent and hence ergodic (since the graph has self-loops) in most applications: we

would expect for a communication pathway to be available between each pair of agents, whenever fair decision-making is desired.

We propose a synchronous discrete-time protocol for reaching agreement[3]. The protocol works as follows. At each time-step, each agent i stochastically updates its opinion using the opinions of its neighbors. The manner in which each agent i updates its opinion between time k and time $k+1$ is as follows:

1. Agent i polls its neighbors for their time-k opinions, and in general observes possibly-faulty versions of these opinions. We use $\mathbf{y}_{ij}[k]$ as a $0-1$ indicator-vector notation for the agent i's observation of agent j's status at time k, for $j \in \mathcal{N}(i)$.

2. Agent i weights and linearly combines its observations. That is, agent i computes the vector $\mathbf{p}_i[k+1] = \sum_{j\in\mathcal{N}(i)} z_{ij}\mathbf{y}_{ij}[k]$, where the weights z_{ij} are non-negative and sum to 1. These weights z_{ij} are design parameters for our protocol. We note that $\mathbf{p}_i[k+1]$ is a probability vector—i.e., its entries are non-negative and sum to 1. For convenience we also use the notation $\mathbf{p}_i[k+1] = \sum z_{ij}\mathbf{y}_{ij}[k]$, where $z_{ij} \triangleq 0$ for $j \neq \mathcal{N}(i)$.

3. We *realize* the time-$(k+1)$ state of agent i according to the probability vector $\mathbf{p}_i[k+1]$. That is, the time-$(k+1)$ opinion of agent i is selected to be $c \in 1,\ldots,m$ with probability listed in component c of $\mathbf{p}_i[k+1]$. Each agent's opinion is updated independently.

4. We stop updating each agent's opinion at a **stopping time** T, which is also a design parameter, once the network is in agreement with high probability and a desired

[3] An analogous asynchronous protocol can be analyzed in a very similar fashion to the synchronous protocol discussed here. Please see [29] for the necessary analysis tools.

agreement law is achieved. The agents' opinions at the stopping time are assumed to be their final ones, based on which an action might be taken.

Thus, we have specified a protocol, which we claim can achieve agreement. Our aim is to show that agreement can indeed be achieved with high probability in this manner, and to design the weights z_{ij} and the stopping time T to achieve a desired agreement law and a high agreement probability—i.e., to achieve agreement in a fair, rapid, and efficacious manner.

A few remarks are in order at this point:

- We stress that our protocol is memoryless (static). The agents in the network do not store, or at least do not make future updates based on, their past opinions. We believe that a static protocol is appealing in many applications because of the limited storage capacity and computational power of sensing agents in these applications. Static protocols are also well-motivated from a humanistic viewpoint, in that individuals tend to seek agreement by arguing for their current viewpoint, rather than basing the argument from their historical sequence of opinions.

- The astute reader will notice that the protocol has a special quasi-linear structure. In particular, the probability vector for the next-opinion of each agent is a weighted *linear* combination of the observations made by each agent. As we will discuss in the next section, this quasi-linear structure (which was studied in detail in the context of the influence model in [17]) permits significant analysis of the network dynamics.

Of course, the cost of restricting ourselves to a quasi-linear protocol is that we risk reduction in performance of the protocol, for example in terms of the agreement laws that we can design. Some evaluation of both the benefits and limitations of our protocol can be found in Section 4.

- We find it convenient to assemble the weights z_{ij} into the **protocol matrix** $Z = [z_{ij}]$. We sometimes use the term **protocol graph** for $\Gamma(Z)$. We note that, in typical cases, we will choose the protocol graph to be ergodic.

Example

Let us return to the autonomous-vehicle example. In our example, we assume that each agent observes two of the other agents. Specifically, we assume that the neighborhoods of the four agents are as follows: $\mathcal{N}(1) = \{4, 1, 2\}$, $\mathcal{N}(2) = \{1, 2, 3\}$, $\mathcal{N}(3) = \{1, 2, 3\}$, and $\mathcal{N}(4) = \{3, 4, 1\}$. The adjacency graph in this case is $D = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$, and so the adjacency graph is the one shown in Figure 4.1.1. The protocol that we propose determines the next-opinion of each agent stochastically, according to a weighted linear combination of the observations available to that agent. For instance, say that, at time k, agent 1 observes that agent 4 and agent 1 have opinion 1 (no target present), and observes that agent 2 has opinion 2 (target present). In indicator vector notation, the observations made by agent 1

are $\mathbf{y}_{14}[k] = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$, $\mathbf{y}_{11}[k] = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}$,and $\mathbf{y}_{12}[k] = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix}$. A particular agreement protocol might determine agent 1's opinion at time $k+1$ according to the probability vector $0.25\mathbf{y}_{14}[k] + 0.5\mathbf{y}_{11}[k] + 0.25\mathbf{y}_{12}[k]$, which would equal $\begin{bmatrix} 0.75 \\ \\ 0.25 \end{bmatrix}$ at this time instant.

4.2.3 A Model for Communication

We have now completely described the agreement problem and suggested a protocol to achieve agreeement. In order to design the parameters of our protocol and to evaluate it, however, we still require a model for the observations that employed in the protocol. In applications of interest, we might expect these observations to be faulty, both because of transmission power constraints on the agents and because of disturbances in the environment. Thus, we seek a model that captures that observations may be faulty versions of the opinions of the other agents.

With this motivation in mind, we consider the following model for observation in the network, or in other words for relating each observation $\mathbf{y}_{ij}[k]$ with the opinion $x_j[k]$, $j \in \mathcal{N}(i)$. If the communication were fault-free, we could assume that $\mathbf{y}_{ij}[k] = \mathbf{s}_j[k]$ (recall that $\mathbf{s}_j[k]$ is the indicator-vector notation for $x_j[k]$. Instead, we model $\mathbf{y}_{ij}[k]$ as being determined based on a probability vector that is parametrized by the current opinion of agent j, $x_j[k]$. That is, given that $x_j[k] = c$, we model $\mathbf{y}_{ij}[k]$ as being realized from the probability vector $A_{ij}(c)$, $c = 1, \ldots, m$. We can describe the model for observations

more concisely, by noting that $\mathbf{y}_{ij}[k]$ is realized from the probability vector $A_{ij}\mathbf{s}_j[k]$, where $A_{ij} = \begin{bmatrix} A_{ij}(1) & \dots & A_{ij}(m) \end{bmatrix}$. Hence, we have postulated a model for the observations. This model for observations together with the specified protocol constitute a complete model for the dynamics of the agents' opinions.

A few notes are in order about our model for observations:

- Notice that the diagonal entries of each A_{ij} represent probabilities that the opinion of agent j is correctly observed by agent i, while the off-diagonal entries represent probabilities of faulty transmission. Based on this, we shall associate a **fault probability** with each edge in the adjacency graph (i.e., with each pair i, j, $j \in \mathcal{N}(i)$. We define this fault probability as the maximum among the sums of the off-diagonal entries in the columns of A_{ij}.

- In designing the parameters of the agreement protocol in Section 3, we shall find it useful to consider both a model in which observation faults do not occur, and one in which faults are possible. Precisely, we refer to a model in which $A_{ij} = I_m$ for all i, $j \in \mathcal{N}(i)$ as a **fault-free model**. We use the term **faulty model** whenever at least one of the A_{ij} is not equal to I_m.

- While the exact model for faults is likely to be specific to the application of interest, it is worthwhile to ruminate on plausible fault models. A simple yet plausible model is one in which each observation is subject to the same fault dynamics, i.e. $A_{ij} = A$ for all i, $j \in \mathcal{N}(i)$. In many applications, we might expect the off-diagonal entries of A to be small but strictly positive. That is, we might expect a small probability of each

opinion being mis-observed as any other opinion. One shortcoming of this model is that the agents' observations of their own opinions are assumed to be faulty. More realistically, we might expect $A_{ii} = I_m$ for each i, while $A_{ij} = A \neq I_m$ for $j \neq i$.

Example

Let us again consider the autonomous-vehicles example, and focus on a single observation made by one of the agents, say $\mathbf{y}_{14}[k]$. In our model, the observation $\mathbf{y}_{14}[k]$ is determined according to a probability vector specified by the current opinion of agent 4. For instance, if agent 4 has opinion 1, the observation specified by $\mathbf{y}_{14}[k]$ is determined according to an arbitrary probability vector, e.g., $\begin{bmatrix} 0.99 \\ \\ 0.01 \end{bmatrix}$. That is, the observation indicates the opinion 1 with probability 0.99, and indicates the opinion 2 with probability 0.01. Similarly, if agent 4 has opinion 2, the observation is realized based on another arbitrary probability vector, say $\begin{bmatrix} 0.04 \\ \\ 0.96 \end{bmatrix}$. A more condensed notation for the observation probability vector in this example is $\begin{bmatrix} 0.99 & 0.04 \\ \\ 0.01 & 0.96 \end{bmatrix} \mathbf{s}_4[k]$.

4.3 Design of the Agreement Protocol

In the previous section, we have proposed a protocol for discrete-valued decision-making (agreement) in a network of sensing agents. What remains to be done is to determine

how the parameters of this stochastic protocol—namely, the weights z_{ij} (equivalently, the protocol matrix Z) and the stopping time T—should be selected, and in turn to decide whether or not our strategy is in fact a useful approach for decision-making or agreement. In this section, we analyze the model for agreement that was introduced with our protocol in the previous section, and use this analysis to design the weights z_{ij} and the stopping time so as to rapidly achieve a desired agreement law.

Our approach for analyzing the model for agreement and designing the agreement protocol is as follows. We first show that our model for agreement can be viewed as an *influence model* (see [17, 18] for an introduction to the influence model), and use this perspective to study the asymptotic behavior of the network when the agreement protocol is applied. This asymptotic analysis shows that, when observation faults may occur, the agreement law loses its dependence on the initial opinions of the agents and hence fair agreement is impossible asymptotically. Thus, we are motivated to seek agreement in a finite time. Specifically, we show how to design the weights z_{ij} to achieve a desired *positive linear* agreement law given that faulty observation does not occur. We then discuss conditions on the network for which these protocols can be used to achieve agreement to desired agreement laws with high probability even when faults may occur. Application of the protocols in these faulty cases is shown to require use of a finite stopping time.

4.3.1 A Tool for Analyzing Asymptotics: the Influence Model

The influence model is a discrete-time and discrete-valued stochastic automaton defined on a network, whose update has a special quasi-linear structure [17]. This quasi-linear update is appealing because it permits characterization of state occupancy probabilities for small groups of agents in the network using low-order linear recursions. The special structure of the influence model also allows for significant characterization of the asymptotics of the automaton, in terms of these low-order recursions (or in terms of matrices or graphs associated with these recursions). An introduction to the influence model can be found in thesis [17] and the article [18], and a few further results can be found in the later thesis [29].

In the influence model, each agent (called a *site* in [17]) can take on one of a finite number of opinions (called *statuses*) at each time step. (In general, the number of statuses taken by each agent, and the interpretation of these statuses, may vary throughout the network.) At each time-step, each agent updates its status independently, as follows:

- The agent polls its neighbors for next-status probability vectors, and then independently realizes its next status based on a weighted linear combination of these probability vectors. The weights are assumed to be non-negative and to sum to 1.
- The probability vector provided by each neighbor is parametrized by the current status of that neighbor; that is, the neighbor provides one of a set of possible probability vectors, depending on its current status.

From this description of the influence model update, it is clear that our model for agreement can be viewed as an instance of an influence model in which each agent can take on the same number of opinions. This interpretation immediately allows us to apply the many analyses of the influence model to our model. For our purposes here, we shall only be concerned with one analysis, namely characterization of the asymptotics of the global dynamics of the network. In particular, we note that the joint opinions of the n agents in our model are governed by a Markov chain with m^n states. Because our model is an influence model, it turns out that characteristics of this **master Markov chain** can be phrased in terms of the protocol matrix $Z = [z_{ij}]$ and the local fault matrices A_{ij}, and hence asymptotic properties of the agreement law can be determined. Unfortunately, for a typical fault model and protocol, these asymptotic properties constitute negative results: they show that the agreement law loses its dependence on the initial condition asymptotically, so that a desired agreement law cannot possibly be designed in an asymptotic sense. Results of this sort are presented for a plausible fault model in the subsequent theorem.

Theorem 4.1 *Consider a communication model where for each i there is $j \in \mathcal{N}(i)$ such that A_{ij} is dense (i.e., all entries are non-zero). For this communication model, the sequence of agreement laws $P(a[k] \,|\, x_1[0], \ldots, x_n[0])$ converges to a function that has no dependence on the initial conditions $x_1[0], \ldots, x_n[0]$, whenever a protocol matrix with strictly positive weights z_{ij} (for $j \in \mathcal{N}(i)$) is used.*

Proof. This theorem can be proved directly using results in [17], but a first-principles proof is simple enough to warrant inclusion. In particular, notice that the transition matrix for the

master Markov chain is dense, since each agent can transition to any other opinion in the course of one time-step and the update for each agent is independent. Since this transition matrix is dense, the master Markov chain comprises a single ergodic class. Hence, from standard Markov chain analysis (see, e.g., [20]), the probability that the master Markov chain is in any state, and so the conditional probability that the network has a particular agreement value given that it is in agreement, converge and do not depend on the inital conditions $x_1[0], \ldots, x_n[0]$. □ □

We remark that the above theorem applies to the typical case in which the graph of the protocol matrix $\Gamma(Z)$ is recurrent and agents have available perfect observations of their own statuses, but communication between different agents is subject to arbitrary faults. The result of the theorem for this typical example makes clear that we cannot in general hope to achieve fair agreement (i.e., to design the agreement law) from the asymptotic dynamics of the network. In particular, in the limit, the agreement value of the network becomes independent of the initial conditions of the agent, and hence agreement law design is impossible. This asymptotic loss of dependence on the initial condition has a conceptual explanation: since our protocols are memoryless, over time the faults that impact the network come to dominate its dynamics in comparison to the initial conditions of the agents, and hence the opinions of the agents become independent of their initial values[4]. Hence, we are motivated to use a finite stopping time for our protocol. More particularly, we are motivated to stop the protocol after enough time has passed that agreement is likely, but

[4]It is worth noting that many other sufficient conditions for independence of the agreement value from the initial opinions can be developed; this loss of information is common when memoryless protocols are used.

before the faults have come to dominate the system dynamics. We pursue this finite-time agreement strategy in the the next sub-section.

Example

Let us again consider the four-agent autonomous-vehicle example introduced in Section 2. From Theorem 4.1, we see that the agreement law asymptotically loses its dependence on the initial statuses of the agents whenever a protocol matrix with a recurrent graph is used and observations are faulty.

Because of the small number of agents in the example, we can easily construct the master Markov chain, and hence verify the result of Theorem 4.1. For instance, when the protocol matrix $Z = \begin{bmatrix} 0.5 & 0.25 & 0 & 0.25 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0.25 & 0 & 0.25 & 0.5 \end{bmatrix}$ is used and the fault model $A_{ij} = A = \begin{bmatrix} 0.99 & 0.04 \\ 0.01 & 0.96 \end{bmatrix}$ is assumed, the asymptotic agreement law becomes $\begin{bmatrix} 0.87 \\ 0.13 \end{bmatrix}$ regardless of the initial opinions of the agents.

4.3.2 Protocol Design for a Model with Faults

In this section, we consider the design of agreement protocols with finite stopping times. In particular, we pursue agreement law design in the context of a fault-free model. We then delineate conditions on the protocol matrix and the fault model given which agreement is achieved with high probability before a fault occurs. Under these conditions, we can apply the protocol designed for the the fault-free model for a finite duration, and guarantee that the desired agreement law is achieved with high probability.

We begin by studying agreement law design for a fault-free model. Again, we find it convenient to view the dynamics of our model as those of an influence model. Since we are concerned with agreement among all the agents in the network, let us again consider the asymptotics of the master Markov chain. In the fault-free case, we note that this master Markov chain has at least m absorbing states: if all the agents have the same opinion, then each agent will retain this opinion since there are no faults in communication. In other words, if the network has reached agreement to any agreement value, it will remain in agreement. In fact, given that the graph of the protocol matrix $\Gamma(Z)$ is ergodic, it can be shown (see [18]) that the network will in fact reach one of these m absorbing states, and hence will reach agreement asymptotically. This result is formalized in the following theorem:

Theorem 4.2 *Consider a fault-free model. If the protocol graph $\Gamma(Z)$ for the model is ergodic, then the network reaches agreement in probability, i.e. the sequence of agreement probabilities converges*[5]

[5] *The network can also be shown to reach agreement with probability* 1. *This stronger notion of stochastic convergence*

to 1*, for any set of initial opinions* $x_1[0], \ldots, x_n[0]$.

Proof. We refer the reader to [18] for the proof in the case that $m = 2$. The generalization of the proof to arbitrary m is trivial, so we omit it. □ □

In the fault-free case, we not only can guarantee agreement asymptotically, but can design the asymptotic agreement law by intelligently choosing the protocol matrix Z. More precisely, we can design the asymptotic agreement law to be any linear function of the form $\alpha_1 \mathbf{s}_1[0] + \ldots + \alpha_n \mathbf{s}_n[0]$, where $\alpha_1, \ldots, \alpha_n$ are positive and sum to 1. This ability to design the agreement law is a primary advantage of our strategy for decision-making (agreement), since it provides flexibility in the dependence of the agreed-upon value on the initial values of the agents. The result is formalized in the following theorem:

Theorem 4.3 *Consider a network with a recurrent adjacency graph, and say that we seek to achieve the* **positive linear** *asymptotic agreement law*

$$\lim_{k \to \infty} \begin{bmatrix} P(a[k] = 1 \,|\, x_1[0], \ldots, x_n[0]) \\ \vdots \\ P(a[k] = m \,|\, x_1[0], \ldots, x_n[0]) \end{bmatrix} = \alpha_1 \mathbf{s}_1[0] + \ldots + \alpha_n \mathbf{s}_n[0], \tag{4.2}$$

where $\alpha_1, \ldots, \alpha_n$ *are strictly positive and sum to* 1*. We can use the following three-step procedure to construct a protocol matrix that achieves this agreement law:*

is not needed for our purposes here.

1. We construct the matrix $\widehat{Z} = \begin{bmatrix} \frac{1}{|\mathcal{N}(1)|} & & \\ & \ddots & \\ & & \frac{1}{|\mathcal{N}(n)|} \end{bmatrix} D$ (where D is the adjacency matrix for the network), and note that $\widehat{Z}$ is a stochastic matrix whose graph is ergodic.

2. We find the left eigenvector $\widehat{\mathbf{w}}$ of $\widehat{Z}$ corresponding to the dominant unity eigenvalue.

3. We choose the protocol matrix to be
$$Z = \begin{bmatrix} \frac{\widehat{w}_1/\alpha_1}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} & & \\ & \ddots & \\ & & \frac{\widehat{w}_n/\alpha_n}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} \end{bmatrix} (\widehat{Z} - I) + I.$$

Furthermore, the probability of agreement converges to 1 *when this protocol is used.*

Proof. This theorem takes advantage of one of the essential tractabilities of the influence model, namely that opinion probabilities of single agents can be found using a low-order linear recursion. Specifically, as long as the designed protocol graph is ergodic (which we shall show to be always true when the procedure above is used), we know that the network reaches agreement asymptotically, and hence that the agreement law can be determined by finding the asymptotics for the opinion-probability vector for any single agent. These opinion probabilities for individual agents satisfy a linear recursion [17, 18]. In particular, for any agent i, we find that $P(x_i[k+1] = j \,|\, x_1[0], \ldots, x_n[0]) = \sum_{l=1}^n z_{jl} P(x_l[k] = j \,|\, x_1[0], \ldots, x_n[0])$ for $j \in 1, \ldots, m$. That is, the probability that an agent i has opinion j at a particular time is a linear combination of the probabilities that the agent's neighbors have opinion j at that time, and hence these probabilities can be found recursively. By stacking

the opinion probabilities into a single vector and applying the probability recursion for k time-steps (see [18] for details), we obtain that

$$\begin{bmatrix} P(x_1[k]=1\,|\,x_1[0],\ldots,x_n[0]) \\ \vdots \\ P(x_1[k]=m\,|\,x_1[0],\ldots,x_n[0]) \\ ---- \\ \vdots \\ ---- \\ P(x_n[k]=1\,|\,x_1[0],\ldots,x_n[0]) \\ \vdots \\ P(x_n[k]=m\,|\,x_1[0],\ldots,x_n[0] \end{bmatrix} = (Z \otimes I_m)^k \begin{bmatrix} \mathbf{s}_1[0] \\ \vdots \\ \mathbf{s}_n[0]. \end{bmatrix} \tag{4.3}$$

From standard linear systems results and and linear-algebraic manipulation, we can find the asymptotic probability vectors for the opinions of the agents, which are equal (since the network is in agreement asymptotically). We thus can find agreement law for the network. Omitting the details (see [18] for these), we find that the asymptotic agreement law is

$$\lim_{k\to\infty} \begin{bmatrix} P(a[k]=1\,|\,x_1[0],\ldots,x_n[0]) \\ \vdots \\ P(a[k]=m\,|\,x_1[0],\ldots,x_n[0]) \end{bmatrix} = w_1\mathbf{s}_1[0] + \ldots + w_n\mathbf{s}_n[0], \tag{4.4}$$

where $\mathbf{w}$ is the left eigenvector of Z corresponding to the dominant unity eigenvalue.

Now let us check that the left eigenvector of Z is in fact the vector $\alpha = \begin{bmatrix} \alpha_1 & \dots & \alpha_n \end{bmatrix}$, to check that the desired agreement law is achieved. To do so, note that

$$\alpha Z = \alpha(\begin{bmatrix} \frac{\widehat{w}_1/\alpha_1}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} & & \\ & \ddots & \\ & & \frac{\widehat{w}_n/\alpha_n}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} \end{bmatrix} (\widehat{Z} - I) + I) = \frac{1}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} \widehat{\mathbf{w}}(\widehat{Z} - I) + \alpha = \alpha. \qquad (4.5)$$

Hence, the desired agreement law is achieved.

It remains to be shown that Z is a stochastic matrix with an ergodic graph. To do so, notice first the $\widehat{Z}$ is a stochastic matrix since its entries are non-negative and each row sums to 1. In fact, the graph $\Gamma(\widehat{Z})$ is ergodic since the graph is recurrent by assumption, and further the diagonal entries of $\widehat{Z}$ are non-zero. Since $\widehat{Z}$ is stochastic and has an ergodic graph, it has a single dominant unity eigenvalue and the corresponding left eigenvector $\widehat{\mathbf{w}}$ is strictly positive (see, e.g., [20]). Now consider $Z = \begin{bmatrix} \frac{\widehat{w}_1/\alpha_1}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} & & \\ & \ddots & \\ & & \frac{\widehat{w}_n/\alpha_n}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} \end{bmatrix} (\widehat{Z} - I) + I$. Note that $\widehat{Z} - I$ has row sums equal to zero, negative diagonal entries, non-negative off-diagonal entries, absolute row sums less than 1, and has a graph that is recurrent. Since $\widehat{\mathbf{w}}$ and $\widehat{\alpha}$ are strictly positive, $\begin{bmatrix} \frac{\widehat{w}_1/\alpha_1}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} & & \\ & \ddots & \\ & & \frac{\widehat{w}_n/\alpha_n}{\sum_{i=1}^n \widehat{w}_i/\alpha_i} \end{bmatrix} (\widehat{Z} - I)$ also has row sums equal to zero, negative diagonal entries, absolute row sums less strictly than 1, and has a recurrent graph. Finally, we find that Z has row sums equal to 1 and non-negative entries. The diagonal entries are strictly positive and the graph remains connected, so Z is a stochastic matrix

with an ergodic graph.

Since Z is ergodic, we also see from Theorem 4.2 that the the probability of agreement approaches 1 when this protocol is used. Hence, the proof is complete. □ □

For the fault-free model, we have so far designed a protocol to achieve a desired agreement law, and also shown that the probability of agreement approaches 1 asymptotically when this protocol is applied. In fact, we can also lower-bound the rates at which agreement, and a desired agreement law, are achieved; such results on the speed of agreement are valuable to gauge whether the developed protocol is effective even when faults may occur. Bounds on the rate of agreement and the rate of convergence to the asymptotic agreement law are given in the following theorem:

Theorem 4.4 *Consider a fault-free model, and assume that the graph of the protocol matrix is ergodic. Then the probability that the network is not in agreement by time k is upper-bounded for any set of initial opinions by a function of the form $C\lambda^k$, where C is a positive constant and $\lambda = \max(|\lambda_s(D)|, |\lambda_d(D_2)|)$. Here, $\lambda_s(D)$ refers to the subdominant eigenvalue of D. Also, D_2 is the matrix formed by taking the Kronecker product of D with itself and then removing rows and columns corresponding to self-Kronecker products of rows and columns of D, and $\lambda_d(D_2)$ is the dominant eigenvalue of D_2. Furthermore, the distance between the agreement law at time k and the asymptotic agreement law (in a two-norm sense) is upper bounded by a function of the form $C_2\lambda^k$, where C_2 is a positive constant.*

Proof. The bound on the agreement probability can be proved through a clever formulation

of the master Markov chain of an influence model, which is pursued in the thesis [29]. The details of this proof are unimportant to our current development and so are omitted. The bound on the distance of the agreement law from the asymptotic law also can be obtained by considering the settling properties of the master Markov chain. Again, we feel that the details are unimportant. □ □

What is important is to note that we now have an exponential bound (with respect to time) on the probability of disagreement of the agents. Hence, for a given protocol matrix, we can lower-bound the probability of agreement within a given finite time interval. If, further, the probability of a fault occurring within this time interval can be upper-bounded, we can lower-bound the probability of agreement even when faults are permitted in our model.

The final aspect to our agreement protocol design study is to upper-bound the probability of a fault occurring within a number of time-steps in terms of the probability of a single fault, so that we can guarantee agreement with high probability when the probability of a fault is sufficiently small. The following theorem provides a bound on the probability that a fault occurs within a number of time steps.

Theorem 4.5 *Let* f_{max} *be the* maximum fault probability *among the edges in the adjacency graph, and let* L *be the total number of edges in the adjacency graph. Then the probability that no faulty transmissions have occurred by time-step* k *is greater than or equal to* $(1 - f_{max}L)^k$.

Proof. By viewing the event that a fault occurs somewhere in the network at a given time-

step as the union of the events that a fault occurs on each particular link, we immediately see that the probability of a fault at time k is less than or equal to $f_{max}L$. Hence, the probability that a fault has not occurred by time k is lower-bounded by $(1 - f_{max}L)^k$.

□ □

We have thus developed a lower bound on the probability that no faulty transmissions have occurred by time-step k. For each k, this bound approaches 1 in the limit of small f_{max}. Hence, when f_{max} is sufficiently small, we can use the protocol developed in Theorem 4.3 to achieve agreement with high probability before a fault occurs. In particular, by stopping the algorithm at a time when agreement has been achieved with high probability but a fault has likely not occurred, we can guarantee a high agreement probability while achieving a desired agreement law. This ability to achieve agreement with high probability at a finite stopping time is captured in the following theorem:

Theorem 4.6 *Assume that the adjacency graph of the network is recurrent, and say that we design a protocol matrix Z to achieve a desired asymptotic agreement law assuming fault-free communication, according to Theorem 4.3. When this protocol matrix is used together with a stopping time T, the agreement probability at the stopping time is lower-bounded in the faulty model (for any set of initial opinions) by $(1 - C\lambda^T)((1 - f_{max}L)^T)$ (where C, λ, f_{max}, and L are defined in Theorems 4.4 and 4.5. In the limit as f_{max} approaches 0, the agreement probability approaches 1 when the stopping time is chosen as $\frac{1}{\sqrt{f_{max}L}}$. Furthermore, the agreement law approaches the asymptotic law for the fault-free model in this limiting case.*

Proof. The agreement probability at time T is greater than or equal to the joint probabil-

ity that agreement is achieved at time T and no faults occur before time T. Hence, the agreement probability is upper bounded by the product of the conditional probability of agreement given that no faults have occurred and the probability that no faults have occurred. Thus, invoking Theorems 4.4 and 4.5, we find that the agreement probability at time T is lower-bounded by $(1-C\lambda^T)((1-f_{max}L)^T)$. We can straightforwardly check that this lower bound on the agreement probability approaches 1 in the limit of small f_{max}, when the stopping time $T = \frac{1}{\sqrt{f_{max}L}}$ is used. Further, since the probability of having a fault by this stopping time approaches 0 as f_{max} approaches 0 while the stopping time itself increases unboundedly with decreasing f_{max}, we recover that the agreement law approaches the asymptotic one for the fault-free case. □ □

Example

Let us again consider the four-agent example. Say that we wish to design the agreement law $0.4\mathbf{s}_1[k] + 0.3\mathbf{s}_2[k] + 0.2\mathbf{s}_3[k] + 0.1\mathbf{s}_4[k]$. Using Theorem 4.3, we find that the following protocol matrix achieves the desired agreement law asymptotically, in the fault-free case:

$$Z = \begin{bmatrix} 0.92 & 0.040 & 0 & 0.040 \\ 0.053 & 0.89 & 0.053 & 0 \\ 0 & 0.080 & 0.84 & 0.080 \\ 0.16 & 0 & 0.16 & 0.68 \end{bmatrix}. \tag{4.6}$$

Applying Theorem 4.4, we also obtain that the parameter λ that governs the rate of agreement is $\lambda = 0.92$.

From Theorem 4.6, we know that the agreement protocol designed for the fault-free case can also be used in the faulty case, whenever the probability of failure f_{max} is sufficiently small. Application to this faulty case requires use of a finite stopping time, as discussed in Theorem 4.6.

Because the number of agents in this example is small, we can explore the performance of the protocol further by constructing the master Markov chain. We have done so, assuming a probability of one fault per one-thousand transmissions ($A_{ij} = A = \begin{bmatrix} 0.999 & 0.001 \\ 0.001 & 0.999 \end{bmatrix}$, $f_{max} = 0.001$). From the master Markov chain, we can find the agreement probability at each time-step, given the initial opinions of the agents. These agreement probabilities are plotted in Figure 4.3.2, for the case where agents 1 and 2 initially share opinion 2 while the others have opinion 1. This plot shows that agreement is indeed achieved with high probability before a fault occurs. We also illustrate the agreement law, in Figure 4.3.2. This figure shows that the desired agreement law $\begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix}$ is achieved at a finite time but lost asymptotocally. Finally, it is instructive to simulate the operation of the agreement protocol, to provide a clearer understanding of why a finite stopping time is needed. In Figure 4.3.2, we plot the number of agents that agree that a target is present (opinion 2) at each time-step, We see that the network reaches agreement quickly (in this case to opinion 2), but eventually the network is bumped out of this agreement value and reaches agreement on

opinion 1. Thus, we see that eventually the faults come to dominate the behavior of the protocol, and the dependence of the agreement law on the initial opinions of the agents is lost.

Although in this simple example we can verify the results of our analysis by constructing the master Markov chain, we note that the power of our analysis derives from the fact that in general we do not need to construct the master Markov chain. Design of the protocol parameters and characterization of the protocol's performance can be achieved systematically, without requiring consideration of the joint behavior of all the agents' opinions.

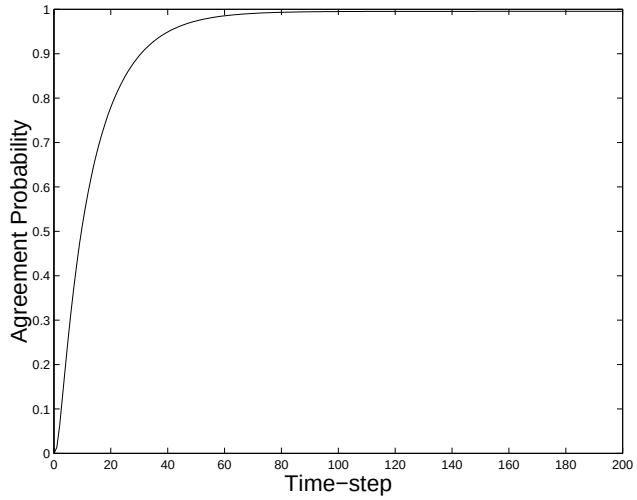

Figure 4.2: The agreement probability at each time-step is shown. We see that the network is in agreement with high probability within 50 time-steps.

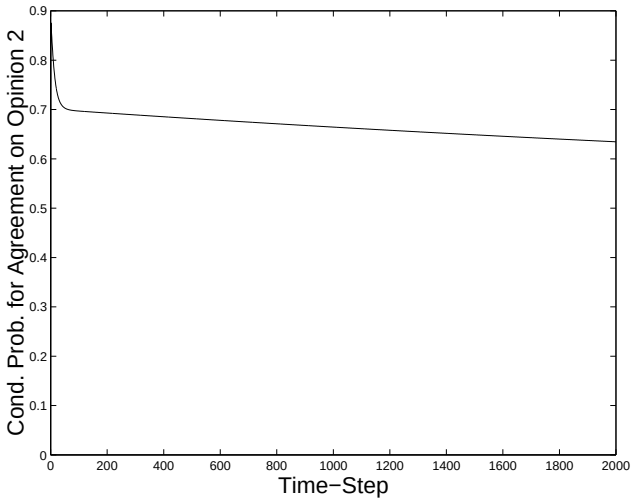

Figure 4.3: The agreement law is illustrated. In particular, the conditional probability of the agreement value 2 given that the network is in agreement is shown. We see that the desired conditional probability of 0.7 is achieved around time-step 30. Slowly, this desired agreement law is lost, as the conditional probability asymptotically approaches the initial condition-independent value of 0.5.

4.4 Discussion

We have developed a protocol for fair agreement or decision-making in a network of sensing agents that may be subject to faults in communication. As is made clear by the autonomous vehicle control example, our protocol holds promise as a tool for distributed decision-making. In particular, it builds on the current control-theoretic studies of agreement protocol design in several respects, including by permitting agreement among discrete-valued opinions, explicitly modeling faults in observation, and introducing the notion of a stopping time.

From a broader perspective, we view our work as a first step toward developing a control

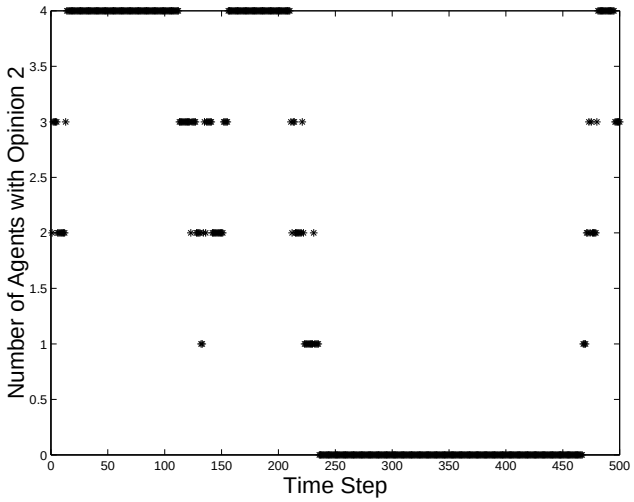

Figure 4.4: The number of agents with opinion 2 is shown for the four-agent autonomous vehicle example. The agents quickly agree on opinion 2, but over time the network is bumped out of agreement by faults, and consequently the dependence of the agreement law on the initial opinions of the agents is also lost.

theory for decision-making in networks of communicating or sensing agents. Our control-theoretic viewpoint allows for such advances as the design of the agreement law, by bringing to bear linear systems and controls notions. We believe that our work (along with other recent control-theoretic studies of agreement, e.g., [2]) also has the complementary benefit of introducing the problem of fair, distributed decision-making to the control community. While this article puts forth some advantages of a control-theoretic viewpoint on agreement protocol design, we believe much remains to be done in evaluating the developed protocol, and in tying our results by those given in the computer science community (e.g., [10]).

The following are a couple specific directions that we believe should be pursued to better evaluate the application of our protocol, and to further improve the protocol.

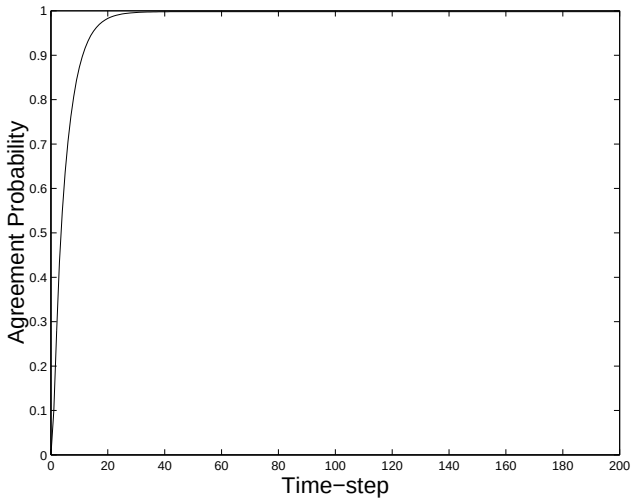

Figure 4.5: The agreement probability at each time-step is shown, when the optimized (scaled) protocol is used. We see that the network is in agreement with high probability within 20 time-steps.

Protocol Optimization: In this article, we have primarily been concerned with deciding whether or not agreement can be achieved, rather than optimizing the rate at which agreement is achieved. Optimization of the rate of agreement is worthwhile, both because rapid agreement may be required in the application of interest, and because improving the rate of agreement may reduce the probability of a fault occurring before the stopping time. One simple strategy for improving the agreement rate is through scaling of the protocol matrix developed in Section 4.3.2. For instance, let us again consider the autonomous vehicle control example. It can easily be checked that the protocol matrix $Z = \begin{bmatrix} 0.75 & 0.13 & 0 & 0.12 \\ 0.17 & 0.66 & 0.16 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ 0.5 & 0 & 0.5 & 0 \end{bmatrix}$ achieves the same desired agreement law as the protocol

developed in Section 4.3.2, but achieves agreement more quickly[6] (according to $\lambda = .81$ rather than $\lambda = 0.92$), as shown in Figure 4.4. We believe that more involved strategies for optimizing the agreement rate could yield significant improvement of the protocol. We refer the reader to [2] for a linear matrix inequality-based strategy for optimizing agreement rates in a particular deterministic model; possibly a similar approach could be used for our models.

Evaluation of Error Probabilities: In this paper, we have only been concerned with the relationship between the agents' initial opinions and a final decision taken by the agents. In reality, we might expect these initial opinions to be noisy and possibly biased of an underlying phenomenon. It would be interesting to enforce a probabilistic model on the initial opinions of the agents, and—using this model—to characterize the success of our decision-making strategy (e.g., in terms of the probability of making the correct decision or the expected cost of the decision taken). One particularly compelling reason for pursuing this Bayesian approach is to better evaluate the constraint of a linear agreement law. It is easy to check that the minimum probability of error decision laws are not linear ones, except in the special case where the optimal strategy is to distribute the opinion of one agent to all the others. Thus, for applications where such minimum error agreement laws are desired, a comparison between our laws and the minimum error laws would be valuable. It is important to stress, however, that there are no known protocols that achieve minimum error agreement among distributed agents, and so linear agreement laws remain

[6]This scaling of the transition matrix can be developed more generally; we chose not to do so in Section 4.3.2 because it frustrates the notation without providing much further insight into the network dynamics.

compelling even for these applications; the flexibility and tractability our protocol may be well worth a degree of suboptimality in some applications. Further, we believe that it is naive to assume accurate knowledge of priors in many applications, and also believe that decision-making may often be impacted by agents' selfish motivations rather than solely a minimum error probability requirement. For these reasons, we have chosen to develop our protocol without consideration of prior probabilities; we leave it to future work to consider decision-making from both a minimum probability of error and a game-theoretic viewpoint.

In considering our study of agreement from this Bayesian viewpoint, we also believe that meshing our strategy with that of [24] is a valuable direction of research. In particular, we believe that the sensor-network error reduction scheme of [24], which suggests using measurements from neighboring sensors to reduce errors in a locally maximum likelihood manner, could be used as a first step in our protocol. Thus, both an immediate reduction in errors and an eventual fair agreement would be achieved.

Bibliography

[1] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations", submitted to *IEEE Transactions on Automatic Control*, April 2003.

[2] R. Olfati Saber and R. M. Murray, "Agreement problems in networks with directed graphs and switching topologies", *IEEE Conference on Decision and Control*, 2003.

[3] D. J. Stilwell and B. E. Bishop, "Platoons of underwater vehicles: communication, feedback, and decentralized control", *IEEE Control Systems Magazine*, Dec. 2000.

[4] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks", submitted to *Automatica*, July 2003.

[5] C. Reynolds, "Flocks, herds, and schools: a distributed behavioral model", *SIGGRAPH*, 1987.

[6] A. Saberi, A. A. Stoorvogel and P. Sannuti, "Decentralized Control with Input Saturation", submitted to the *American Control Conference*, July 2004.

[7] C. Godsil and G. Royle, *Algebraic Graph Theory*, Springer-Verlag: New York, 2000.

[8] S. Wang and E. J. Davison, "On the stabilization of decentralized control systems", *IEEE Transactions on Automatic Control*, vol. AC-18, pp. 473-478, Oct. 1973.

[9] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii USA, Dec. 2003.

[10] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann: San Mateo, CA, 1996.

[11] L. Xiao and S. Boyd, 'Fast Linear Iterations for Distributed Averaging', accepted for publication in *Systems and Control Letters*, 2004.

[12] S. Roy, A. Saberi, and K. Herlugson, 'Formation and Alignment of Distributed Sensing Agents with Double-Integrator Dynamics', accepted for publication in the IEEE Press Monograph *Sensor Network Operations*, 2004.

[13] H. K. Khalil, 'On the existence of positive diagonal P such that $PA + AP < 0$, *IEEE Transactions on Automatic Control*, Vol. AC-27, pp. 181-184, 1982.

[14] D. Hershkowitz, 'Recent Directions in Matrix Stability', *Linear Algebra and its Applications*, Vol. 171, pp. 161-186, Jul., 1992.

[15] R. A. Holley and T. M. Liggett, 'Ergodic theorems for weakly interacting infinite systems and the voter models', *Annals of Probability*, Vol. 3, pp. 643-663, 1975.

[16] P. Clifford and A. Sudbury, 'A model for spatial conflict', *Biometrika*, Vol. 60, No. 3, pp. 581-588, 1973.

[17] C. Asavathiratham and S. Roy and B. C. Lesieutre and G. C. Verghese, 'The Influence Model', *IEEE Control Systems Magazine*, Vo. 21, No. 6, pp. 52-64, Dec. 2001.

[18] C. Asavathiratham, *The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains*, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Oct. 2000.

[19] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes, 3^{rd} ed.*, Oxford University Press, Aug. 2001.

[20] R. G. Gallager, *Discrete Stochastic Processes*, Kluwer Academic Publishers: Boston, 1996.

[21] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison Wesley: Menlo Park, CA, 1973.

[22] P. Seiler and R. Sengupta, "Analysis of Communication Losses in Vehicle Control Problems", *Proceedings of the American Control Conference*, Arlington, Virginia, June 25-27, 2001.

[23] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Distributed sensor networks – a review of recent research", *Journal of the Franklin Institute*, vol. 338, pp. 655-668, 2001.

[24] B. Krisnamachari and S. S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks", *IEEE Transactions on Computers*, Vol. 53, No. 3, Mar. 1, 2004.

[25] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks", *Proceedings of the the ACM/IEEE International Conference on Mobile Computing and Networks (MobiCom 1999)*, Aug. 1999.

[26] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks,", *Proceedings of the the International Journal of High Performance Computing Applications*, 2002.

[27] S. Roy, A. Saberi, and K. Herlugson, "A control-theoretic perspective on the design of distributed agreement protocols" submitted to *Automatica*.

[28] R. Durrett, "An introduction to infinite particle systems," *Stochastic Porcesses and Their Applications*, pp. 109-150, 1981.

[29] S. Roy, *Moment-linear stochastic systems and their applications*, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Jun. 2003.