Formation and Alignment of Distributed Sensing Agents with Double-Integrator Dynamics and Actuator Saturation

Sandip Roy* Ali Saberi*† Kristin Herlugson*†

June 11, 2004

Abstract

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

*School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, U.S.A.{sroy,saberi,kherlugs}@eecs.wsu.edu

†Support for this work was provided by the Office of Naval Research under grant number N000140310848.

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.

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., [1, 2, 3, 4, 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

[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

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.

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

use the term sensing architecture, though our formulation may quite possibly provide good representation for certain communication architectures also.

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 [1], 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 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.

2.1 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 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

$$\mathbf{a}_{ij} = g_{j1}(i) \begin{bmatrix} r_1 \\ v_1 \end{bmatrix} + \ldots + g_{jn}(i) \begin{bmatrix} r_n \\ v_n \end{bmatrix}, \tag{1}$$

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

$$\mathbf{a}_i = C_i \mathbf{x}, \tag{2}$$

where

$$C_i = \begin{bmatrix} G_i & 0 \\ 0 & G_i \end{bmatrix}. \tag{3}$$

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., [2]. 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:

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

where $\mathcal{N}_i$ are the neighbors of agent i (see [2] 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., [2], 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:

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

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

$$G_{3x} = G_{3y} = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}. \tag{4}$$

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:

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

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

$$G_{3x} = G_{3y} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}. \tag{5}$$

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:

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

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 [1]. 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:

$$G_{1x} = G_{1y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$G_{2x} = G_{2y} = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$G_{3x} = G_{3y} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

2.3 Vector Representation

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

$$\begin{bmatrix} \dot{r}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} r_i \\ v_i \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \sigma(u_i), \tag{7}$$

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:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} \sigma(\mathbf{u}), \tag{8}$$

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 (8) and observations given by (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**.

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 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 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

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

by applying decentralized stabilization results for linear systems ([8]) and for systems with saturating actuators ([6]).

Theorem 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 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

$$det\left(\begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} K \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix}\right) = 0, \tag{9}$$

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 (9) 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 (9) can be rewritten as follows:

$$det\left(\begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I_n \end{bmatrix} K \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} \right) = det\left(\begin{bmatrix} 0 & I_n \\ Q_p & Q_v \end{bmatrix} \right), \tag{10}$$

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

$$\sum_{i=1}^{n} \alpha_i \tilde{\mathbf{k}}_p(i) G_i \tag{11}$$

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:

$$Q_p = \begin{bmatrix} k_1 G_1 \\ \vdots \\ k_n G_n \end{bmatrix} = \begin{bmatrix} b_1^T \\ \vdots \\ b_n^T \end{bmatrix}. \tag{12}$$

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 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 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 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 [2] Earlier, we discussed that the Laplacian sensing architecture of [2] is a special case of our sensing architecture. Now we are ready to compare the stabilizability results of [2] 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 [2] (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 [2], 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.

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 [1], 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:

$$\begin{aligned}
\mathbf{b}_{1x}^T &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\mathbf{b}_{1y}^T &= \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{b}_{2x}^T &= \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \end{bmatrix} \\
\mathbf{b}_{2y}^T &= \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \\
\mathbf{b}_{3x}^T &= \begin{bmatrix} -1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \\
\mathbf{b}_{3y}^T &= \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{13}$$

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.

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 intance, flocking behaviors may involve only convergence of differences between agents' positions or velocities (e.g., [3]). 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) stabilization 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 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 $V \triangleq \begin{bmatrix} \mathbf{b}_1^T \\ \ldots \\ \mathbf{b}_n^T \end{bmatrix}$ 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, deferring a formal proof to a future article.

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 $A_c = \begin{bmatrix} 0 & I \\ K_1G & K_2G \end{bmatrix}$ 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 nullace 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.

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 [2] (within the context of double-integrator networks), and brings our work in alignment (no pun intended) with the studies of [3]. 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 [3]. The graph matrix[3] in our example is

$$G = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 1 \end{bmatrix} \tag{14}$$

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

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

[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.

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.

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.

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 6 *Consider a linear double-integrator network with graph matrix G. Let $\mathcal{K}$ be*

[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).

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).*

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

$$A_c = \begin{bmatrix} 0 & I \\ K_1 G & K_2 G \end{bmatrix}, \tag{15}$$

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

$$A_c = \begin{bmatrix} 0 & I \\ KG & aKG \end{bmatrix}. \tag{16}$$

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

$$\mathbf{w}' A_c = \begin{bmatrix} \alpha\rho \mathbf{v}' & \mathbf{v}'(1 + \alpha a \rho), \end{bmatrix} \tag{17}$$

where ρ—the eigenvalue of KG corresponding to the eigenvector $\mathbf{v}'$—lies in the OLHP. This vector is a left eigenvector of A_c with eigenvalue λ if and only if $\alpha\rho = \lambda$ and $1 + \alpha a \rho = \alpha\lambda$.

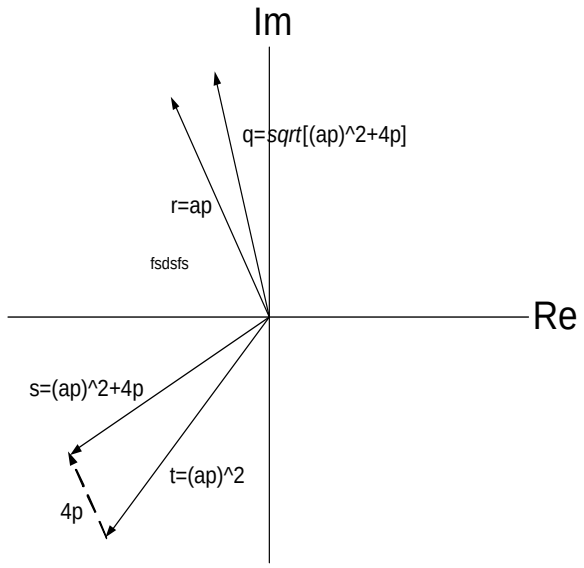

Figure 1: We introduce the notation used for the geometric proof that the eigenvalues of A_c are in the OLHP.

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 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

$$a > \sqrt{\frac{2}{|\rho| cos(90 - tan^{-1}\frac{-Re(\rho)}{Im(\rho)})}}. \tag{18}$$

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

$$a > \max_{\rho} \sqrt{\frac{2}{|\rho| cos(90 - tan^{-1}\frac{-Re(\rho)}{Im(\rho)})}}, \tag{19}$$

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 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

$$A_c = \begin{bmatrix} 0 & I \\ KG & aKG \end{bmatrix}. \tag{20}$$

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

$$\widehat{A}_c = \begin{bmatrix} 0 & I \\ \frac{KG}{\zeta^2} & \frac{aKG}{\zeta} \end{bmatrix}. \tag{21}$$

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.

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 controllers 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.

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.

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 8 *If there exists an agent list* $\mathbf{i}$ *such that the kth 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 (please see appendix for the technical details). 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.

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.

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:

$$K = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{22}$$

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, 3, and 4 show the trajectories of the vehicles, and demonstrate the effect of the static control gains on the performance of the closed-loop system.

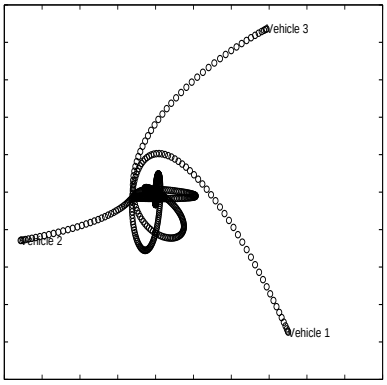

Figure 2: Vehicles converging to a target: coordination with an intermediary.

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.

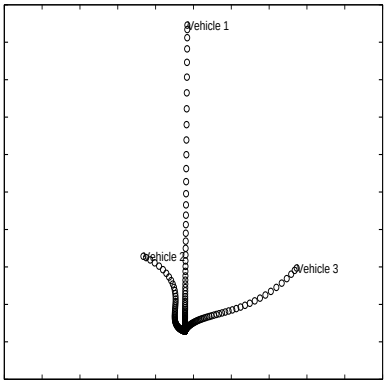

Figure 3: Vehicles converging to the target. The gain matrix is scaled up by a factor of 5 as compared to Figure 2.

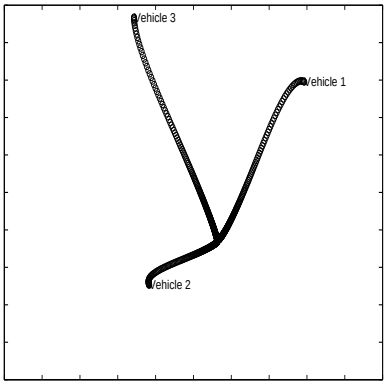

Figure 4: Vehicles converging to the target when the HVG controller parameter a is increased from 1 to 3.

5.3.2 Example: Vehicle Velocity Alignment, Flocking

Finally, we simulate an example, with some similarity to the examples of [3], 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 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.

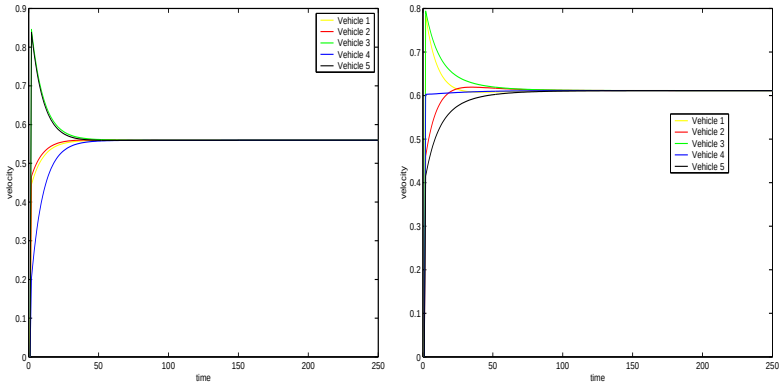

Figure 5: We show alignment of x-direction velocities in two simulations. Notice that the final velocity is different for the two simulations.

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 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 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., [3], 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 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.

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 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 $C \triangleq \begin{bmatrix} C_1 & 0 & \ldots \\ & \ddots & 0 \\ \ldots & 0 & C_n \end{bmatrix}$.

[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.

[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.

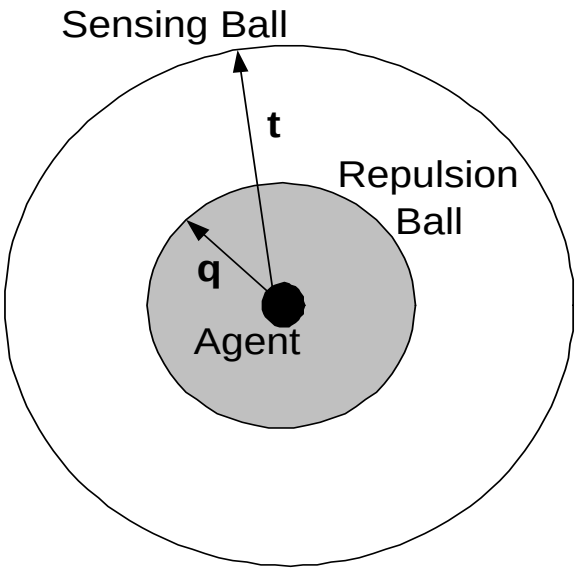

Figure 6: The repulsion ball and local sensing ball are illustrated.

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. 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 [3]. 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. 6). That is, agent i has available the 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 $PDIN(G, C, q, t)$.

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, 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. 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 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

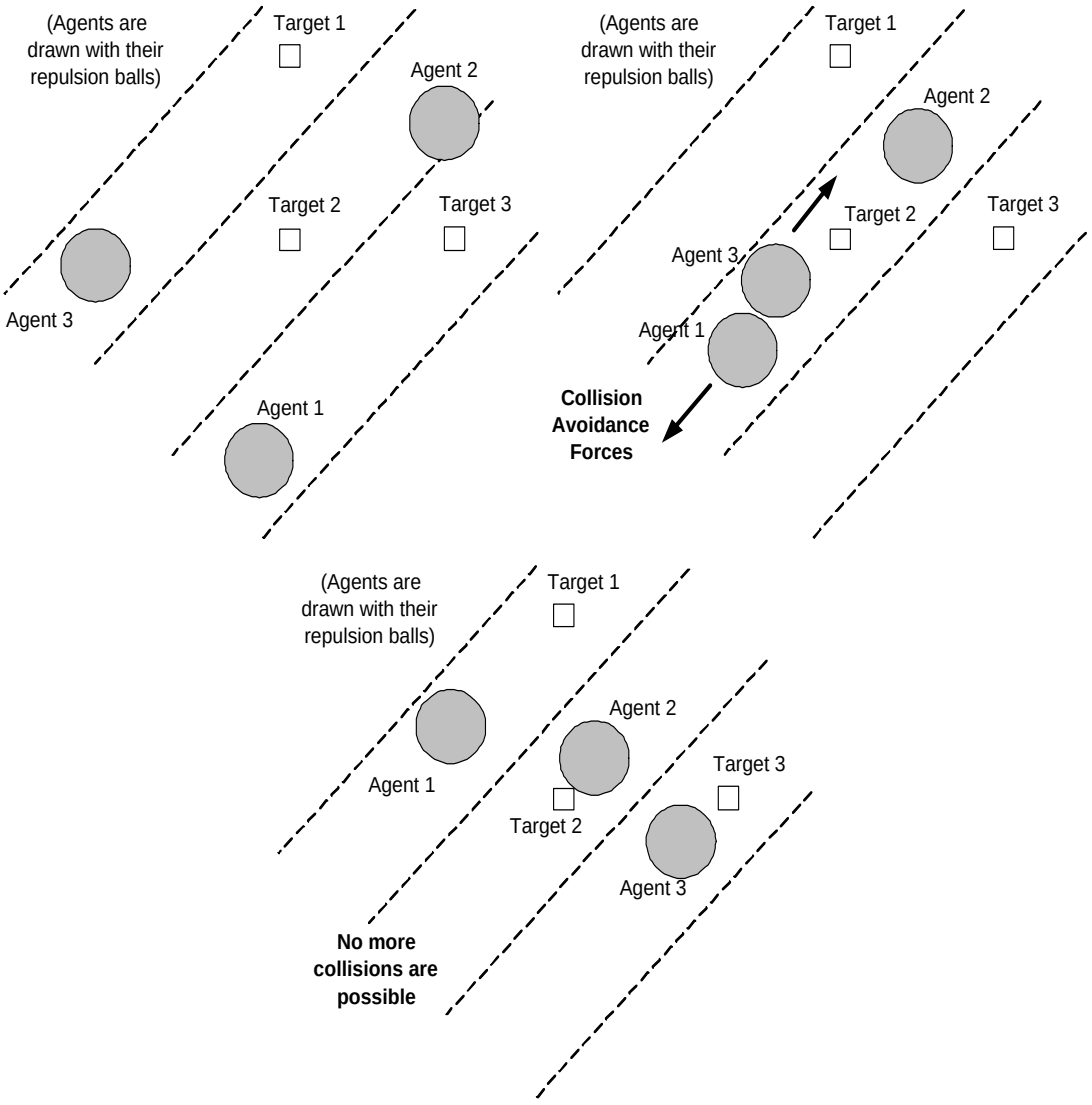

Figure 7: Our approach for formation stabilization with collision avoidance is illustrated, using snapshots at three time points.

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 1 *Consider a $PDIN(G, C, q, t)$ whose agents seek to converge to the formation $\overline{\mathbf{r}}$, 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. 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 $\overline{\mathbf{r}}$, that formation is called* **valid**.

Theorem 9 *$PDIN(G, C, q, t)$ has a dynamic time-invariant controller that achieves both formation stabilization to $\overline{\mathbf{r}}$ (i.e., convergence of the agents' positions to $\overline{\mathbf{r}}$) and collision avoidance if G and C have full rank, and $\overline{\mathbf{r}}$ 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 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

$$\begin{aligned} \dot{\mathbf{z}} &= A_x \mathbf{z} + B_x \mathbf{y}_x, \\ \mathbf{u}_x &= C_x \mathbf{z} + D_x \mathbf{y}_x, \end{aligned} \tag{23}$$

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:

$$\begin{aligned} \dot{\mathbf{z}} &= A_x \mathbf{z} + B_x \mathbf{y}_x, \\ \mathbf{u}_x &= C_x \mathbf{z} + D_x \mathbf{y}_x + \begin{bmatrix} \sum_{j=1}^{n} g(\mathbf{r}_1 - \mathbf{r}_j) \\ \vdots \\ \sum_{j=1}^{n} g(\mathbf{r}_n - \mathbf{r}_j) \end{bmatrix}, \end{aligned} \tag{24}$$

where the non-linear input term $g()$, which uses the warning measurements, is described in

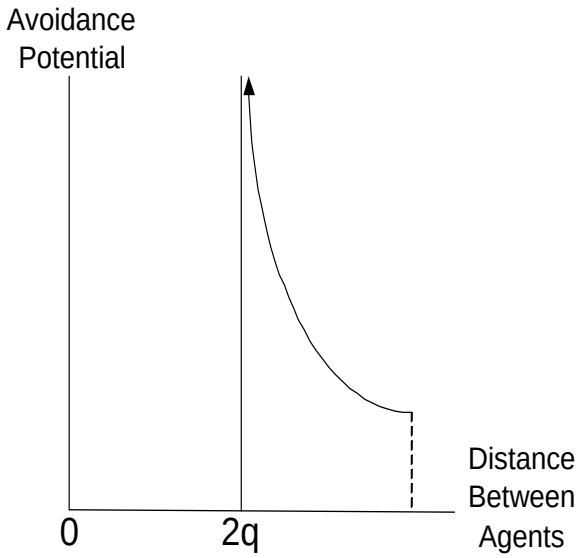

Figure 8: An example potential function for collision avoidance is shown.

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 $2q \leq ||\mathbf{r}_i - \mathbf{r}_j||_2 \leq \epsilon$ 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. 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

[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.

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 $\overline{\mathbf{r}}$, 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 $\overline{\mathbf{s}}_i = \begin{bmatrix} a & b \\ -b & a \end{bmatrix} \overline{\mathbf{r}}_i$, so the network seeks to converge to the formation $\overline{\mathbf{s}} \triangleq (I_n \otimes \begin{bmatrix} a & b \\ -b & a \end{bmatrix}) \overline{\mathbf{r}}$. 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 $\overline{\mathbf{s}}$ 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.

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 1). 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. 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. 10 shows a simulation of collision avoidance, 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. 11 shows a more advanced approach to collision avoidance. In this example, the

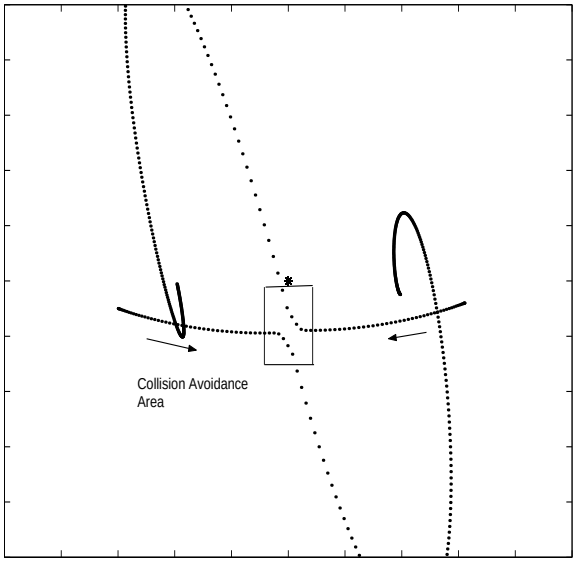

Figure 9: Formation stabilization with collision avoidance is shown.

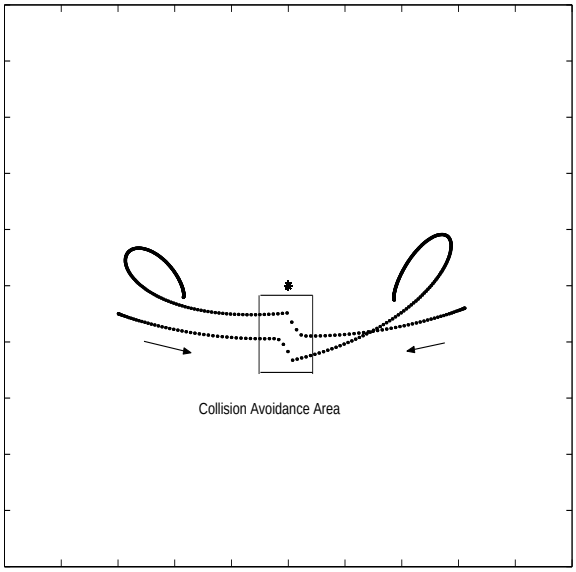

Figure 10: Another protocol for collision avoidance is simulated. Here, we have eliminated the overshoot after collision avoidance using a braking acceleration.

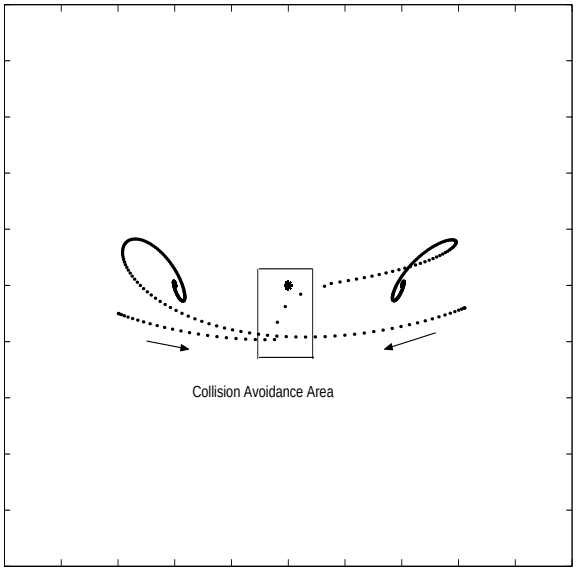

Figure 11: In this simulation, we achieve collision avoidance by guiding the agents along curves in space, once a potential collision is detected.

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.

References

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

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

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

[4] R. Olfati Saber and R. M. Murray, "Agreement problems in networks with directed graphs and switching topologies", *IEEE Conference on Decision and Control*, 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* 42*nd IEEE Conference on Decision and Control*, Maui, Hawaii USA, Dec. 2003.