FORMATION AND TRACKING IN SENSING AGENT NETWORKS: CONTROLLER DESIGN AND SECURITY

By

LIANG CHEN

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

MASTERS OF SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering & Computer Science

December 2006

To the Faculty of Washington State University:

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

_____
Co-Chair

_____
Co-Chair

_____

Acknowledgments

I am grateful to a number of people who have made the succesful culmination of my degree and this thesis possible. First, I thank my advisors, Sandip Roy and Ali Saberi, for their brilliant ideas which guided this research and advanced my knowledge, and for the help they have given me throughout the research process. Thank you to Professor Thomas Fischer for the illuminating conversation on research topics. The research that led to this dissertation could have not been accomplished without the help of my advisors and committee members. I thank my friends and colleagues—Zheng Wen, Carolina Parada, James Slade and Robin Jacob—for their help in my study process. I also thank my parents for their immense love and support throughout my graduate studies.

This work was supported in part by NSF Grant ECS-0528882 (Sensors), and by the Office of Naval Research under Grant N000140310848.

FORMATION AND TRACKING IN SENSING AGENT NETWORKS: CONTROLLER DESIGN AND SECURITY

Abstract
by Liang Chen, M.S.
Washington State University
December 2006

Co-Chair: Sandip Roy

Co-Chair: Ali Saberi

Distributed control systems have been receiving more and more attention these years because of some of their advantages over centralized control systems.

One significant advantage is that distributed control systems enable communicating agents to complete complex dynamic tasks cooperatively. One powerful control strategy to achieve such complex tasks is to have each agent track a specified command signal. This control problem is known as tracking control. These tracking problems require communications within the network, in particular, we consider *double-integrator networks*. Noting that the communication capabilities of simple agents are highly limited, we develop the desirable low-information-flow tracking algorithms for double-integrator networks. In the process, we also study the role played by sensing topology of the network, and identify the trajectory information that must be distributed to the agents.

Another important advantage of distributed control systems is their potential to achieve security. In this thesis, we shall motivate and define the notions of security for *single-integrator networks*. Based on our definition, we then study how the parameters of the controllers/algorithms can be chosen to achieve algorithmic goals while maintaining security.

CONTENTS

LIST OF FIGURES

To my dear parents and Xiaoyan

1. INTRODUCTION

A sensing-agent network consists of communicating agents, each of which has some measurement capabilities. In several modern applications, a sensing-agent network is required to complete a complex task (see, e.g., [1,2,3]). In some of these applications, the teams must complete *tracking tasks*—ones in which each agent's state (e.g., position) must follow a specified command signal (e.g. [3,4,5]). For instance, teams of autonomous vehicles which sense relative positions may need to follow a "lawn-mower" pattern, to search a minefield. Similarly, a bank of antennas may need to follow a path in a coordinated fashion. Very broadly, these tracking problems require two sorts of *information flow*: 1) local sensing for *control*, i.e. using which each agent can correctly actuate its dynamics (change its state) so as to follow a specified path; and 2) higher-level communication for *trajectory-distribution*, i.e. for disseminating each agent's desired path to it. Often, in tracking applications for modern communicating-agent teams, the information flow must be highly limited, in that it must be both local in space and sparse in time (see, e.g., [7] for motivation regarding AUVs). In chapter 2, we marry well-known techniques for servo control with techniques for decentralized control and distributed-algorithm-development, to develop low-information-flow tracking algorithms for networks of sensing/communicating agents. In doing so, we delineate the role played by sensing topology of the network in our ability to achieve tracking. We also study what trajectory information must be distributed to the agents, and explore means for distributing this information with sparse or no communication.

In using distributed or decentralized algorithms for the communicating-agent networks, one oft-advanced motivation is the need for *security*. That is, distributed algorithms seemingly have the potential to be secure, in the sense that an adversary observing internal dynamics or communications in one part of the network may nevertheless be unable to characterize the whole behavior of the network. While this notion is an important motivation for using distributed algorithms and controllers, however, we do not know of any systematic attempt to formalize this concept in the control-theoretic algorithm-development paradigm.

In chapter 3 of this thesis, we aim to motivate and define the notions of security and fault tolerance for a class of simple yet applicable algorithms and controllers. To be more precise, we consider security for models with *single-integrator* agent dynamics, general linear observation (communication/sensing) topologies, and linear static feedback; such a model has been shown to appropriately capture and permit control of interesting mechanical systems (e.g. autonomous-vehicle teams, see [13], and also can be used as protocol for achieving algorithmic tasks in networks (e.g., formation and alignment or *agreement* among a network of sensors, see [13]).

Based on the definition for security, we then study how the parameters of the controllers/algorithms can be chosen to achieve algorithmic goals while maintaining security.

This thesis is organized in the following way: Chapter 2 addresses the low-information-flow tracking controller design for double-integrator networks; Chapter 3 motivates and defines the notions of security, and present some results on the secure tracking and formation controller design, for single-integrator networks.

2. INFORMATION FLOW REQUIRED FOR TRACKING CONTROL IN NETWORKS OF MOBILE SENSING AGENTS

We take the following approach to studying tracking control in communicating-agent networks. Section 2.1 develops the double-integrator-network model, and pose the decentralized tracking problem. In Section 2.2, we show how decentralized tracking can be achieved in a double-integrator network with a given communication/sensing topology when the appropriate trajectory signals are distributed to each agent. In turn, we identify the function of trajectory signals (from neighbors) that must be distributed to each agent in order to complete the tracking task. In Section 2.3, we introduce a set of motions—specifically, translation, rotation, and expansion—that are typical of multi-agent *formations*. We show that, given a set of agents that only need move in formation (i.e., along these trajectories), the path-distribution task can be achieved through very sparse communication. Finally, Section 2.4 explores how agents can *infer* trajectory information rather than being sent this information.

2.1 Model and Problem Formulation

In this section, we motivate and describe a model comprising a network of communicating/sensing agents with double-integrator internal dynamics, and then introduce the *tracking problem* in the context of this *double-integrator network*.

2.1.1 The Double Integrator Network

We consider a network of n mobile sensors or vehicles or **agents**, labeled $1, \ldots, n$. These agents aim to cooperatively complete a dynamic task, by actuating their internal dynamics using sensed observations. Let us motivate and introduce our model for the internal dynamics for the agents, and then consider the sensing architecture.

Increasingly, modern *communicating-agent networks* (networks of mobile sensors, autonomous vehicle teams) are made up of agents with simple but highly-constrained dynamics. For many such networks, a plausible canonical model for an agent's dynamics is a saturating double integrator. We adopt this saturating double-integrator model for our agents' internal dynamics, i.e., we assume agent i is governed by

$$\ddot{r}_i = \sigma(u_i), \tag{2.1}$$

where r_i is the **position**[*] of agent i, u_i is agent i's input or actuation, and $\sigma()$ is the standard saturation function. We also find it convenient to refer to the derivative of agent i's position with respect to time, (i.e. $v_i = \dot{r}$) as agent i's velocity. We note that for the above dynamics, each agent is assumed to have a scalar position. However, there is no loss of generality since it turns out that the internal dynamics, observations and controller of a multiple-dimensional agent can be equivalently represented using multiple scalar agents (see [2]).

For some applications (e.g., computational ones, where agents' states represent opinions/estimates rather than actual positions), actuator saturation may not be a significant constraint. Motivated by such applications, and so as to make the presentation of results clearer, we often first

[*] We use the term **position** for an agent's state with the vehicle-control application in mind (in which Newton's law yields the double-integrator model), but in fact double-integrator models can capture various dynamics of interest in communicating-agent networks (see [2] for some examples)

consider the case where agents are governed by pure double integrators

$$\ddot{r}_i = u_i \tag{2.2}$$

rather than saturating double integrators.

Each agent in the network has certain sensing capabilities. Our model for sensing is quite general: each agent is assumed to make multiple observations, each of which is a linear combination of (in general multiple) agents' current positions or velocities. We note that such a model permits representation of absolute and relative position observations, among others. Specifically, we suppose agent i has $2m_i$ observations (m_i observations on positions, and m_i observations on velocities) that can be written in the form

$$\mathbf{a}_{p_i} = G_i \mathbf{r} \tag{2.3}$$

$$\mathbf{a}_{v_i} = G_i \mathbf{v},$$

where $\mathbf{r} \triangleq \begin{bmatrix} r_1 & \dots & r_n \end{bmatrix}^T$, $\mathbf{v} \triangleq \begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}^T$, and the **graph matrix** G_i has dimension $m_i \times n$. It should be noted (as can be seen from Equation 2.3) that each agent is assumed to have identically-structured observations on positions and velocities. This restriction is plausible in many applications (see [2] for motivation) and also can be eliminated by considering more general controllers than pursued here; the methods developed here can be generalized to this case. For convenience, we also define the **full graph matrix** as $G = \begin{bmatrix} G_1^T & \dots & G_n^T \end{bmatrix}^T$.

We assume that agent i has available its observations for computing its actuation (input). That is, the observations $\mathbf{a}_{p_i}$ and $\mathbf{a}_{v_i}$ of agent i are considered as information that is available to the local controller. Our goal is to design a *static linear controller* (a controller without memory, specifically one that sets the input u_i to a linear combination of the current observations) for each agent i, so as to globally achieve a tracking task[†].

[†] Our motivation for considering static controllers is that they are easily implementable even in devices with limited

We refer to the internal dynamics, communication topology, and decentralized control paradigm described above together as a **saturating double-integrator network** (respectively, **double-integrator network** in the case where actuator saturation is ignored).

2.1.2 Tracking Problem Formulation

Motivated by a range of applications, we aim to design controllers for the double-integrator network for **tracking**, i.e. controllers using which each agent i's position can follow a **desired trajectory** $\bar{r}_i(t)$, $t \geq 0$. We find it convenient to refer to a set of desired trajectories $\bar{r}_1(t), \ldots, \bar{r}_n(t)$ together as the **tracking task** $(\bar{r}_1, \ldots, \bar{r}_n)$.

Typically, in the controls literature, desired trajectories are assumed to be ones that can be generated by an autonomous linear system (which is termed an *exosystem*): almost all trajectories of interest (for instance, ramp, step, or sinusoidal signals) can be represented in this way (see e.g. [10]). For us, signals of this form are compelling with respect to their distribution to agents, since they can be specified by the parameters (more specifically, *eigenvalues* or *modes*) of the autonomous systems and the initial condition of the autonomous system's state. Thus, whenever convenient, we assume (without meaningful loss of generality) that the each desired trajectory is generated by the exosystem

$$\dot{\mathbf{w}}_i = S_i \mathbf{w}_i \tag{2.4}$$

$$\bar{r}_i = \mathbf{d}_i^T \mathbf{w}_i \tag{2.5}$$

where $\mathbf{w}_i \in R^{q_i}$, the system matrix S_i is assumed without loss of generality to have eigenvalues in the CRHP (since tracking is an asymptotic task, see [10]), and $\mathbf{d}_i$ is a constant vector of dimension

complexity. Consideration of static control also clarifies the exposition of information flow for tracking; many results readily generalize to settings where more complicated controllers are used.

$q_i \times 1$. The initial conditions $\mathbf{w}_i(0)$ are set so that the desired trajectory is generated. The following example illustrates the trajectory-generation from an exosystem.

Suppose the desired trajectory for agent i is of the form $\bar{r}_i(t) = t + sin(2t)$, let us construct the exosystem for agent i.

Notice that the trajectory consists a ramp and a sinusoidal signal. The ramp signal has a initial value of 0 and a slope of 1, and the sinusoidal signal has an amplitude of 1, a frequency of 2 rad/s and an initial phase of 0.

Then the trajectory signal for agent i is given by

$$\bar{r}_i = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \mathbf{w}_i, \text{ where } \dot{\mathbf{w}}_i = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & -2 & 0 \end{bmatrix} \mathbf{w}_i \text{ with } \mathbf{w}_i(0) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Our aim is to design controllers for each agent, so that their positions follow the desired trajectories in an asymptotic sense. Let us define the achievement of the tracking task formally:

Definition 1: A double-integrator network is said to achieve or complete the tracking task $(\bar{r}_1, \ldots, \bar{r}_n)$, if the error signals $e_i(t) \triangleq r_i(t) - \bar{r}_i(t)$, $i = 1, \ldots, n$, approach 0 as $t \to \infty$.

We note that tracking is an *asymptotic* task; a settling time describing how quickly tracking is achieved can be obtained, see [10] for details.

2.2 Tracking and the Required Information Flow

In this section, we first give conditions under which a double-integrator network (with and without actuation saturation) can achieve tracking and show how a controller can be designed to

do so. Throughout this section, we assume full information flow regarding the tracking task, i.e., we assume that the agents are given all required information about the signals in the tracking task $(\bar{r}_1, \ldots, \bar{r}_n)$. We also delineate carefully what information about the tracking task must be given to each agent.

First, we consider a double-integrator network that is not subject to actuation saturation, and show how to develop a controller for the tracking task, in the process giving broad conditions under which tracking is possible.

Theorem 1: A double-integrator network can complete any tracking task $(\bar{r}_1, \ldots, \bar{r}_n)$ if there exists a block diagonal matrix $K = \begin{bmatrix} \mathbf{k}_1^T & & \\ & \ddots & \\ & & \mathbf{k}_n^T \end{bmatrix}$ (where each $\mathbf{k}_i$ is a m_i-component vector) such that all eigenvalues of KG are in the OLHP.

In this case, the tracking task can be achieved by driving each agent i with the input

$$u_i = \begin{bmatrix} \mathbf{k}_i^T & \alpha \mathbf{k}_i^T \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_{p_i} \\ \mathbf{a}_{v_i} \end{bmatrix} - \begin{bmatrix} G_i D \mathbf{w} \\ G_i D S \mathbf{w} \end{bmatrix} \right) + \mathbf{d}_i^T S_i^2 \mathbf{w}_i \tag{2.6}$$

where α is a sufficiently large positive number, $D = \begin{bmatrix} \mathbf{d}_1^T & & \\ & \ddots & \\ & & \mathbf{d}_n^T \end{bmatrix}$, $S = \begin{bmatrix} S_1 & & \\ & \ddots & \\ & & S_n \end{bmatrix}$, and $\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$.

Proof: We prove our theorem by verifying that each agent i actually follows the desired trajectory $\mathbf{d}_i^T \mathbf{w}_i$ with the above input signal.

Noting that $\mathbf{a}_{p_i} = G_i\mathbf{r}$ and $\mathbf{a}_{v_i} = G_i\mathbf{v}$, when the above input signal is applied, the dynamics of agent i is given by

$$\begin{cases} \dot{r}_i = v_i \\ \dot{v}_i = u_i = \mathbf{k}_i^T G_i\left(\mathbf{r} - D\mathbf{w}\right) + \alpha\mathbf{k}_i^T G_i\left(\mathbf{v} - DS\mathbf{w}\right) + \mathbf{d}_i^T S_i^2\mathbf{w}_i \end{cases} \tag{2.7}$$

Making the state transform of $z_{p_i} = r_i - \mathbf{d}_i^T\mathbf{w}_i$ and $z_{v_i} = v_i - \mathbf{d}_i^T S_i\mathbf{w}_i$, the dynamics for agent i becomes

$$\begin{cases} \dot{z}_{p_i} = z_{v_i} \\ \dot{z}_{v_i} = \mathbf{k}_i^T G_i\mathbf{z}_p + \alpha\mathbf{k}_i^T G_i\mathbf{z}_v \end{cases} \tag{2.8}$$

where $\mathbf{z}_p = \begin{bmatrix} z_{p_1} & \dots & z_{p_n} \end{bmatrix}^T$, and $\mathbf{z}_v = \begin{bmatrix} z_{v_1} & \dots & z_{v_n} \end{bmatrix}^T$. Assembling the dynamics of n agents in the double-integrator network, we get the following state equation:

$$\begin{bmatrix} \dot{\mathbf{z}}_p \\ \dot{\mathbf{z}}_v \end{bmatrix} = \begin{bmatrix} 0 & I \\ KG & \alpha KG \end{bmatrix} \begin{bmatrix} \mathbf{z}_p \\ \mathbf{z}_v \end{bmatrix} \tag{2.9}$$

In our previous work (see [2]), it has been proved that if the eigenvalues of KG are in the OLHP, the eigenvalues of $\begin{bmatrix} 0 & I \\ KG & \alpha KG \end{bmatrix}$ are guaranteed to be in OLHP for sufficiently large positive α. Hence $\mathbf{z}_p, \mathbf{z}_v \to 0$ as $t \to \infty$, and then it follows that for each agent i, $e_i = z_{p_i} \to 0$ as $t \to \infty$. $\square$

A couple notes about this theorem are worthwhile:

1) We stress that the controllers being used to achieve the tracking task are *static* (memoryless) linear ones: the input $u_i(t)$ at each time t is a linear function of the current observations.

2) We have chosen to present the control inputs in terms of the exosystem parameters, with the motivation that this form is often the most easily implemented one. We can easily phrase the input in terms of the desired trajectories. In this notation, we find that $u_i = \begin{bmatrix} \mathbf{k}_i^T & \alpha\mathbf{k}_i^T \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_{p_i} \\ \mathbf{a}_{v_i} \end{bmatrix} - \begin{bmatrix} G_i\bar{\mathbf{r}} \\ G_i\dot{\bar{\mathbf{r}}} \end{bmatrix} \right) + \ddot{\bar{r}}_i$, where $\bar{\mathbf{r}}^T = \begin{bmatrix} \bar{r}_1 & \dots & \bar{r}_n \end{bmatrix}$.

According to Theorem 1, tracking is possible whenever there exists an appropriately-structured (block-diagonal) K such that KG has all eigenvalues in the OLHP. Essentially, whenever this condition holds, the controller can be chosen to make the trajectory stable[‡]. Thus, we refer a double-integrator network for which there is K such that KG has all eigenvalues in the OLHP as a **stabilizable double-integrator network**. We note that the linear-algebra problem of whether there exists block-diagonal K such that KG has eigenvalues in the OLHP is well-studied (e.g., [11,12]), and there are several broad classes of full graph matrices for which such stabilizing K exists. We refer the reader to [2,12] for details.

So far, we have developed conditions under which an unsaturating double-integrator network can achieve tracking; we note that the amplitude of the input signal u_i may be arbitrarily large when the controller in Theorem 1 is used, and hence the result does not necessarily carry through to the case where actuators may saturate. We next develop a condition for tracking in a saturating double-integrator network. Conceptually, tracking under saturation requires the further condition that the actuator can provide enough acceleration at all times to move each agent along its desired trajectory, plus an arbitrarily small amount of further acceleration for convergence to the trajectory (stabilization). Under these conditions, by making the convergence to the trajectory sufficiently slow, we can achieve tracking for an arbitrarily large set of initial conditions. That is, tracking is achieved in a *semi-global* sense, i.e. given any closed and bounded ball of initial conditions for the saturating double-integrator network, there exists a controller that achieves tracking for any initial condition in this ball. This notion is formalized in the following theorem:

Theorem 2: Consider the tracking problem of a saturating double-integrator network with n agents. A tracking task $(\bar{r}_1, \ldots, \bar{r}_n)$ can be achieved for any given closed and bounded set of initial condi-

[‡] That is, the desired trajectories are attractive and also stable in the sense of Lyapunov.

tions, say $\mathbf{X}_0$, for (2.1), if the two following conditions hold:

(I) There exists a block diagonal matrix $K = \begin{bmatrix} \mathbf{k}_1^T & & \\ & \ddots & \\ & & \mathbf{k}_n^T \end{bmatrix}$ (where $\mathbf{k}_i$ is an m_i-component vector) such that all eigenvalues of KG are in the OLHP.

(II) There exists a $\delta > 0$ and a $T \geq 0$ such that $\|\mathbf{d}_i^T S_i^2 \mathbf{w}_i\|_{\infty,T} \leq 1 - \delta$, (or equivalently, $\|\frac{d^2 \bar{r}_i}{dt^2}\|_{\infty,T} \leq 1 - \delta$) for all $i = 1, \ldots, n$.

In this case, the tracking task can be achieved by driving each agent i with the following family of input signals, parameterized in ε:

$$u_i = \begin{bmatrix} \varepsilon^2 \mathbf{k}_i^T & \alpha\varepsilon \mathbf{k}_i^T \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_{p_i} \\ \mathbf{a}_{v_i} \end{bmatrix} - \begin{bmatrix} G_i D \mathbf{w} \\ G_i D S \mathbf{w} \end{bmatrix} \right) + \mathbf{d}_i^T S_i^2 \mathbf{w}_i \tag{2.10}$$

where $\varepsilon \in (0, \varepsilon^*]$, and ε^* is a function of the radius of the initial condition ball for Equation (2.1) and the double-integrator network parameters, α is a sufficiently large positive number, $D = \begin{bmatrix} \mathbf{d}_1^T & & \\ & \ddots & \\ & & \mathbf{d}_n^T \end{bmatrix}$, $S = \begin{bmatrix} S_1 & & \\ & \ddots & \\ & & S_n \end{bmatrix}$, and $\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$.

Proof: We prove this theorem by showing that for each given closed and bounded set of initial condtions $\mathbf{X}_0$, there exists an $\varepsilon^* > 0$ such that for all $\varepsilon \in (0, \varepsilon^*]$, the tracking task can be achieved using the proposed control law when the two conditions hold.

The family of inputs for agent i are of the form

$$u_i = \begin{bmatrix} \varepsilon^2 \mathbf{k}_i^T & \alpha\varepsilon \mathbf{k}_i^T \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_{p_i} \\ \mathbf{a}_{v_i} \end{bmatrix} - \begin{bmatrix} G_i D \mathbf{w} \\ G_i D S \mathbf{w} \end{bmatrix} \right) + \mathbf{d}_i^T S_i^2 \mathbf{w}_i.$$

Note that the constructed input signal takes the same form as that given in Theorem 1, except the gain matrix is parameterized in ε. With this input signal, the closed-loop dynamics of each agent

i is

$$\dot{r}_i = v_i \tag{2.11}$$

$$\dot{v}_i = \sigma(u_i) \tag{2.12}$$

Making a state transform (identical to that in the proof of Theorem 1) $z_{p_i} = r_i - \mathbf{d}_i^T \mathbf{w}_i$ and $z_{v_i} = v_i - \mathbf{d}_i^T S_i \mathbf{w}_i$, the dynamics for agent i becomes

$$\begin{cases} \dot{z}_{p_i} = z_{v_i} \\ \dot{z}_{v_i} = \sigma(\varepsilon^2 \mathbf{k}_i^T G_i \mathbf{z}_p + \alpha\varepsilon \mathbf{k}_i^T G_i \mathbf{z}_v + \mathbf{d}_i^T S_i^2 \mathbf{w}_i) - \mathbf{d}_i^T S_i^2 \mathbf{w}_i \end{cases} \tag{2.13}$$

where $\mathbf{z}_p = \begin{bmatrix} z_{p_1} & \dots & z_{p_n} \end{bmatrix}^T$, and $\mathbf{z}_v = \begin{bmatrix} z_{v_1} & \dots & z_{v_n} \end{bmatrix}^T$.

We then show that by choosing the parameter ε sufficiently small, the input of each agent i remains in the linear region $\forall\ t \geq T$ and the tracking task is achieved.

Let us first consider the transformed dynamics of agent i if the actuation were not subject to saturation

$$\begin{cases} \dot{z}_{p_i} = z_{v_i} \\ \dot{z}_{v_i} = \varepsilon^2 \mathbf{k}_i^T G_i \mathbf{z}_p + \alpha\varepsilon \mathbf{k}_i^T G_i \mathbf{z}_v \end{cases} \tag{2.14}$$

Assembling the dynamics without saturation of n agents, we have the state equation

$$\begin{bmatrix} \dot{\mathbf{z}}_p \\ \dot{\mathbf{z}}_v \end{bmatrix} = \begin{bmatrix} 0 & I \\ \varepsilon^2 KG & \alpha\varepsilon KG \end{bmatrix} \begin{bmatrix} \mathbf{z}_p \\ \mathbf{z}_v \end{bmatrix} \tag{2.15}$$

We make an observation that if the eigenvalues of $\begin{bmatrix} 0 & I \\ KG & \alpha KG \end{bmatrix}$ are denoted as $\lambda_1, \dots, \lambda_{2n}$, then the eigenvalues of $\begin{bmatrix} 0 & I \\ \varepsilon^2 KG & \alpha\varepsilon KG \end{bmatrix}$ are $\varepsilon\lambda_1, \dots, \varepsilon\lambda_{2n}$. Therefore, for any $\varepsilon > 0$ the state trajectory of system (2.15) is identical to that of system (2.9), except that the time axis is scaled by ε. Since

when condition (I) is satisfied, the system (2.9) is asymptotically stable by choosing proper α and $\hat{K}$, it follows that the system (2.15) is asymptotically stable. Moreover, for any initial conditions in a ball, the state trajectory stays in a ball (probably larger) and is independent of ε.

We then consider the original system in which agents are subject to actuation saturation. We observe that at time T, the transformed state $\begin{bmatrix} \mathbf{z}_p(T) \\ \mathbf{z}_v(T) \end{bmatrix}$ belongs to a bounded set, since $\begin{bmatrix} \mathbf{z}_p(0) \\ \mathbf{z}_v(0) \end{bmatrix}$ is bounded and $\begin{bmatrix} \mathbf{z}_p(T) \\ \mathbf{z}_v(T) \end{bmatrix}$ is determined by a linear differential equation with bounded inputs.

Then if we consider the dynamics of agent i from time T onwards, there exists an ε_i^* such that for all $t \geq T$, $\varepsilon^2 \mathbf{k}_i^T G_i \mathbf{z}_p + \alpha\varepsilon \mathbf{k}_i^T G_i \mathbf{z}_v < \delta$ if $\varepsilon \in (0, \varepsilon_i^*]$. Hence the input of each agent remains in linear region for all $t \geq T$, when $\varepsilon \in (0, \varepsilon^*]$, where $\varepsilon^* = min\{\varepsilon_1^*, \ldots, \varepsilon_n^*\}$.

Finally, since $z_{p_i} \to 0$ as $t \to \infty$, we have $e_i \to 0$ as $t \to \infty$, and the tracking task is achieved. □

Tracking of three typical trajectories (without actuator saturation)—circular, sinusoidal and lawn-mower trajectories—are illustrated using the control law proposed in Theorem 1. The simulation results are shown in Figure 2.1.

From the form of the inputs in Theorem 1, we notice that each agent i's input depends only on its own observations, as required; however, computation of the input in general requires knowledge of the desired trajectories of multiple agents. In the subsequent sections, we will develop methods for providing the agents with this information using minimal communication when the agents are known to move in formation, and will even explore whether agents can deduce the required trajectories when only a leader has been told the tracking task. Before considering these special cases, we first identify the trajectory information that in general must be provided to agent i to achieve the

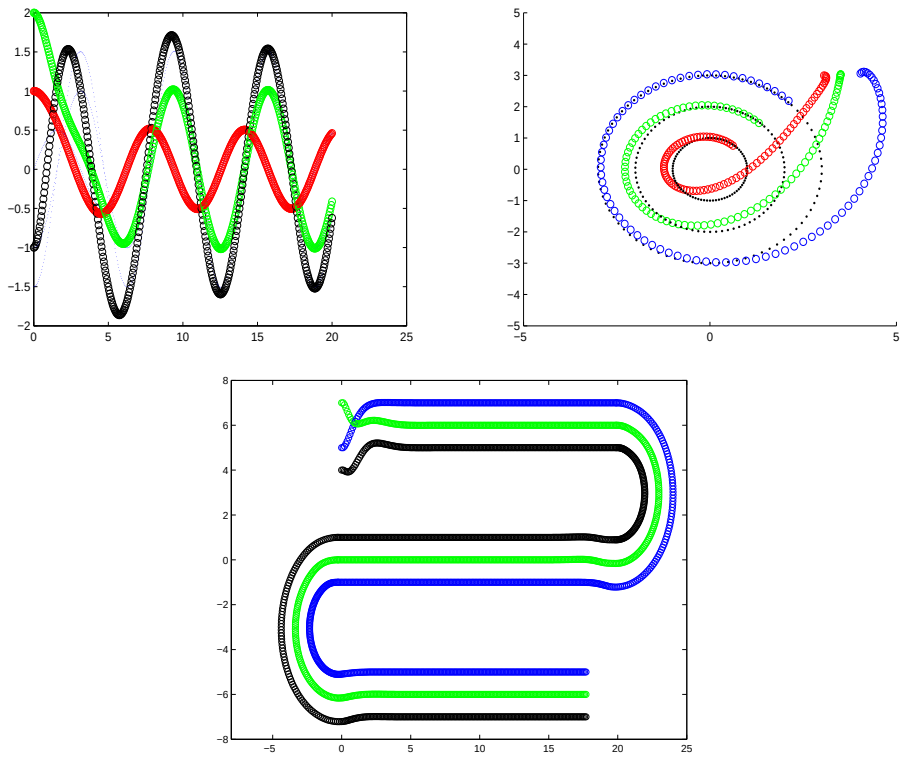

Fig. 2.1: Tracking of circular, sinusoidal and lawn-mower trajectories is illustrated. In all three examples, the agents' sensing capabilities are distributed, i.e. each agent makes different observations that are combinations of multiple agents' states.

tracking task. We find that the required information is deeply connected to the structure of the communication/sensing topology, i.e., of the graph matrices $G_1, \ldots, G_n$. To present this result, let us formulate the notion of a **graph** for the network communication/sensing:

Definition 2: The **network graph** is a directed graph with n nodes labeled $1, \ldots, n$, which corresponds to the n agents. The network graph has an edge from vertex j to vertex i, if and only if $G_{ij} \neq \vec{0}$, where $i \neq j$ and G_{ij} is the jth column of the graph matrix G_i.

We note that a directed edge from vertex j to vertex i indicates that the observations made by agent i depend on the position/velocity of agent j.

We also find it convenient to define the notion of neighbors from the network graph. In particular, if the network graph has an edge from vertex j to vertex i, we refer to vertex (equivalently, agent) j as an **upstream neighbor** of vertex (equivalently, agent) i. We use the notation U(i) for the set of upstream neighbors of agent i, and use the term **upstream neighbor set** for this set.

Let us now give the general result on the trajectory information required by each agent:

Theorem 3: When the control laws proposed in Theorems 1 and 2 are used to achieve a tracking task, each agent i requires a signal $\bar{z}_i$, which is a function of the trajectories $\bar{r}_j$, $j \in U(i)$, as well as the trajectory $\bar{r}_i$, in order to achieve the tracking task. Specifically, the agent i requires the signal

$$\bar{z}_i(\bar{r}_{U(i)}, \bar{r}_i) = -\mathbf{k}_i^T \Sigma_{j \in U(i)} G_{ij}(\bar{r}_j + \alpha \frac{d\bar{r}_j}{dt}) + \frac{d^2\bar{r}_i}{dt^2}, \tag{2.16}$$

where we have used the notation $\bar{r}_{U(i)}$ for the set of trajectories $\bar{r}_j$, $j \in U(i)$.

Proof: Equation (2.16) follows directly from the control laws (2.6) and (2.10). □

Theorem 3 makes clear that each agent i requires a signal $\bar{z}_i$, which is a function of the trajectories that the agent and its upstream neighbors must follow. Since the statistic $\bar{z}_i(\bar{r}_{U(i)}, \bar{r}_i)$ specifies the trajectory information that must be distributed to or computed by agent i, we refer to the function $\bar{z}_i()$ as the **trajectory information distribution function (TIDF)** for agent i, and refer to a signal $\bar{z}_i$ generated by this function for a particular set of trajectories as a **trajectory information signal (TIS)** for agent i.

We stress here that the tracking task CANNOT be achieved by simply sending each agent its own desired trajectory: the TIS for agent i depends on the desired trajectories not only of agent i but of its upstream neighbors. More specifically, agent i's input (actuation) can be viewed as comprising two components, one ($\frac{d^2\bar{r}_i}{dt^2}$) that provides the agent with the power needed to follow its desired trajectory and only depends on its own trajectory, and one ($-\mathbf{k}_i^T \Sigma_{j \in U(i)} G_{ij}(\bar{r}_j + \alpha \frac{d\bar{r}_j}{dt})$)

that is needed for agents to stabilize about the trajectory and depends on the desired trajectories of upstream neighbors.

Let us put forth a couple perspectives that contextualize the notion of **trajectory-distribution**, i.e. of providing each agent with appropriate information about the desired trajectories:

A Decentralized-Control Perspective From one perspective, our development of a trajectory-distribution paradigm advances the literature on *decentralized tracking*. While there is a wide literature in this area (e.g. [6]), our work differs from the existing literature in the following way: in contrast to the literature, we do not assume that each agent's observations include the variable(s) that are required to follow the desired trajectory. That is, although we expect for the position of agent i to follow a desired trajectory, we do not assume that the agent i necessarily has an observation of its position. The assumption in the literature is greatly simplifying, in the sense that each agent only needs its own desired trajectory signal to compute the control input; however, the assumption is too restrictive for our purposes, since we require that agents be able to complete tasks only through network measurements/communication.

The comparison of our work with the literature leads us to consider the important notion of decentralization in tracking problems. For the strategies given in the literature, the trajectory-distribution aspect of the task is totally decentralized, in that each agent only needs its own desired trajectory to compute its actuation. In contrast, in our setting the trajectory-distribution aspect of the task is not completely distributed, in that each agent's actuation depends not only on its own desired trajectory but on the desired trajectories of other agents. However, this dependence on other trajectories is sparse, in the sense that each agent's actuation only depends on the trajectories of upstream neighbors in addition to its own trajectory. Hence, we can view trajectory distribution as being partially decentralized in our problem.

We note here that *decentralization* in trajectory distribution is a conceptual notion: it describes the dependence of an agent's actuation signal on the *desired* trajectories of the n agents, rather than on actual observations made by the agents. The structure of this dependence does not immediately specify a communication scheme for trajectory distribution since the required communication also depends on the specifics of where the desired trajectories are generated/stored, and where the actuation signals are computed. However, understanding the structural dependence of the actuation signal on desired trajectories is important, in that it clarifies the content/complexity of computations and communications needed for trajectory distribution.

An Information-Communication Perspective Let us now consider trajectory distribution from the perspective of how the required information can be communicated to the agents. In general, trajectory-distribution can be achieved by sending each agent's TIS (or the desired trajectories from which the TIS can be computed) to it before a tracking task is to be completed, or during the commission of the task. Alternately, in some special cases, trajectory distribution can be achieved without any special communication if the agents infer the necessary desired trajectory information through their sensed observations. Such detection of or *adaptation* to the correct trajectory may be feasible when the possible desired trajectories of the agents are limited to a small set; we address this case in Section 2.4.

In the typical case that trajectory information must be sent to the agents, a trajectory-communication scheme which overlays the sensing network (observation graph) may well be needed. The TIDF for each agent makes clear the minimal statistic that must be communicated to that agent to achieve the tracking task.

In many applications, the desired trajectories may be decided on by a central authority. For instance, a set of robots may be tasked to sweep a minefield in a specified pattern. In such cases,

each TIDF indicates the sparsity of the computation required of the authority to generate the corresponding TIS: specifically, the authority must combine the desired trajectories of upstream neighbors (and their time-derivatives) to obtain the TIS for a particular agent, which is then sent to the agent using some overlayed communication scheme.

In other applications, the desired trajectories may themselves be chosen in a distributed fashion: either each agent may choose its own trajectory, or the trajectories may be set by a group of leaders in the network. (Such distributed trajectory-generation is sensible, for instance, in networks with a large number of agents that are concurrently participating in several different tasks.) In such fully-distributed settings, the TIDF indicates that communication of the trajectory signals from upstream neighbors is sufficient for each agent to compute its TIS. Thus, in this case, the TIDFs explicitly illustrate the sparsity and required topology for trajectory communication.

We have thus given conditions on the communication/sensing topology for tracking in double-integrator networks and saturating double-integrator networks. In the process, we have exposed the need for trajectory-distribution, identified the trajectory information signals needed by each agent, and indicated the dependence of these signals on the sensing/communication topology.

2.3 Information Distribution in Formation

Our development in Section 2.2 suggests that, in general, substantial effort is needed to distribute information about desired trajectories to the agents in the network. However, for many autonomous vehicle control and mobile sensing applications, the agents in a network are required to move in *formation*, i.e. they maintain a geometric pattern as they move through space. Further, in some cases, this formation moves through space according to some simple rules (e.g., the

formation has to be at certain locations at given times rather than following an arbitrarily-set trajectory). We can take advantage of these structures to simplify the trajectory distribution aspect of the trajectory task. More specifically, we show how evaluation of each agent's TIDF is simplified when the agents are known to move in formation, and in turn discuss how trajectory-ditribution can be achieved with simpler or less communication.

To consider tracking in formation, we must impose a geometric (spatial) interpretation for the agents' states. Specifically, motivated by typical autonomous-vehicle-control applications, we consider a network of n agents moving in the plane[§]. Each agent's x-direction (horizontal) and y-direction (vertical) motions are governed by double integrators, i.e. the agent's controller sets accelerations in each direction with the goal of controlling its position in the plane. Each agent makes a set of position observations, each of which is a linear combination of agents' x and y-direction positions. For each position observation, the agent is assumed to make a corresponding velocity observation, which is the same combination of the agents' velocities. We refer to this model as a **planar double integrator network (PDIN)**. In the case where actuators may saturate, we use the term **saturating planar double integrator network**.

We are interested in achieving tracking in the PDIN. As with the double-integrator network, we specify a tracking task with a set of trajectories for the desired motions of each agent. For a PDIN, the agents move in the plane, so we specify a **desired x-trajectory** (desired trajectory in the x-direction) $\overline{r}_{ix}(t)$ and **desired y-trajectory** $\overline{r}_{iy}(t)$ for each agent i. We refer to the desired trajectories together as the **tracking task** for the double-integrator network.

As briefly discussed in Section 2.1 and in [2], we can straightforwardly reformulate a PDIN (with associated tracking task) as a double-integrator network with $2n$ agents (and an associated

[§] The generalization to higher-dimensional motion is straightforward.

tracking task). From the graph matrices of the equivalent double-integrator network, we can decide whether the double-integrator network is stabilizable, and hence whether the PDIN can achieve the tracking task. In this case (i.e., when the double-integrator network is stabilizable), we also refer to the PDIN as **stabilizable**. For a stabilizable PDIN, the tracking task can be completed, i.e. each agent can asymptotically follow its desired x-trajectory and y-trajectory. In this case, we can again find a minimum statistic about the desired trajectories that must be provided to each agent to permit complete the tracking task. In keeping with the general case, we refer to the (two) signals that must be given to agent i (for x-direction and y-direction tracking) to permit tracking as the **trajectory information signals** (TIS).

We are interested in trajectory distribution for *formation-tracking* tasks in stabilizable PDINs, i.e. tracking tasks in which the desired trajectories maintain a fixed pattern in the plane at all times. In particular, we claim that trajectory-distribution for formation tracking can be achieved with simpler/less communication than for general tracking tasks. To expose this simplification, let us begin by carefully defining the notion of a formation and of formation tracking. To define formation, we first define the notion of a nominal formation, i.e. a set of points that describe a pattern in the plane.

Definition 3: A nominal formation F_0 is an ordered set of n pairs $(\widehat{r}_{1x}, \widehat{r}_{1y}), \ldots, (\widehat{r}_{nx}, \widehat{r}_{ny})$, along with a reference $\widehat{r}_{0x}$ and $\widehat{r}_{0y}$. The nominal formation describes a pattern of points in the plane, together with a reference point for this pattern.

We refer to a set of points as being in the formation F_0, if these points form the same pattern in space as the nominal formation F_0:

Definition 4: An ordered set of n points in the plane (i.e., n pairs) $(r_{1x}, r_{1y}), \ldots, (r_{nx}, r_{ny})$ is said

to be in the formation F_0, if all points $(\widehat{r}_{ix}, \widehat{r}_{iy})$ in the nominal formation can be placed on the corresponding points (r_{ix}, r_{iy}) through expansion around the reference point, rotation around the reference point, and translation in the x- and y-directions. That is, $(r_{1x}, r_{1y}), \ldots, (r_{nx}, r_{ny})$ is in formation F_0 if there are parameters a, θ, p_x, and p_y such that $r_{ix} = a[(\widehat{r}_{ix} - \widehat{r}_{0x})cos(\theta) + (\widehat{r}_{iy} - \widehat{r}_{0y})sin(\theta)] + p_x$ and $r_{iy} = a[-(\widehat{r}_{ix} - \widehat{r}_{0x})sin(\theta) + (\widehat{r}_{iy} - \widehat{r}_{0y})cos(\theta)] + p_y$, for all i.

The three typical trajectories–translation in the $x-$ and $y-$directions, rotation around the reference point and expansion around the reference point–are illustrated in Figure 2.2.

We are interested in having a network of agents complete a tracking task while in formation. Let us thus formally define the notion of a formation-tracking task:

Definition 5: A formation-F_0 tracking task is one in which the set of desired trajectories $(\overline{r}_{1x}(t), \overline{r}_{1y}(t)), \ldots, (\overline{r}_{nx}(t), \overline{r}_{ny}(t))$ is in the formation F_0, at each time t.

That is, a formation-F_0 tracking task is one in which the desired or nominal trajectories of the agents are in the formation F_0 at all times t. We stress here that formation is enforced on the desired trajectories, not on the agents themselves. If the agents are able to complete the tracking task, however, they enter and remain in formation after some time passes. This is a sensible assumption for many applications, in that vehicles/agents must move into formation at the commencement of a task and then remain in formation. We note that our notion of formation tracking is identical to the notion of *convergence to formation* developed in [3], except in that our notion permits rotation and expansion in addition to translation.

Since the desired trajectories at each time are in the formation F_0 at each time t, it is automatic

that the desired trajectories for each agent i, $1 \leq i \leq n$, can be written in the form

$$\overline{r}_{ix}(t) = a(t)[(\widehat{r}_{ix} - \widehat{r}_{0x})cos(\theta(t)) + (\widehat{r}_{iy} - \widehat{r}_{0y})sin(\theta(t))] + p_x(t)$$

$$\overline{r}_{iy}(t) = a(t)[-(\widehat{r}_{ix} - \widehat{r}_{0x})sin(\theta(t)) + (\widehat{r}_{iy} - \widehat{r}_{0y})cos(\theta(t))] + p_y(t) \tag{2.17}$$

for four signals $a(t)$, $\theta(t)$, $p_x(t)$, and $p_y(t)$. We refer to the signals $a(t)$,$\theta(t)$, $p_x(t)$, and $p_y(t)$ as the **expansion**, **rotation**, **x-translation**, and **y-translation** parameters, respectively. We notice that these **formation-tracking signals** together with the nominal formation specify completely the tracking task.

Since the tracking task is specified by formation-tracking signals and nominal formation, only this information or some subset thereof need be communicated to each agent for trajectory distribution. Let us formalize this notion:

Theorem 4: Consider a stabilizable PDIN. The TIS for each agent i's x-trajectory or y-trajectory is in general a function of the formation-tracking signals $(a(t), \theta(t), p_x(t), p_y(t))$ as well as the nominal-formation parameters $\widehat{r}_{ix}$, $\widehat{r}_{iy}$, $\widehat{r}_{jx}$, $\widehat{r}_{jy}$, and $\widehat{r}_{0x}$, $\widehat{r}_{0y}$, $j \in U(i)$.

Theorem 4 makes clear the information distribution required for formation tracking. In many applications, we envision that agents know in advance (or can obtain with minimal communication) their required information about the nominal formation, i.e. the location of the the reference and the locations of upstream agents (and the agent itself) in formation. In this case, all agents only need be given, or be able to obtain, the four formation-tracking signals. That is, information distribution for formation-tracking tasks is simpler in that all agents only need to be sent or to be able to compute a common set of signals. From these four common signals together with observations, they can each compute their required input.

While formation-tracking permits a significant simplification in information distribution, the

distribution task is still taxing in the sense that (in general) the signals $a(t)$, $\theta(t)$, $p_x(t)$, and $p_y(t)$ must be distributed to the agents or computed by them. In most settings, the formation may only need to follow a few trajectories with simple structure to achieve desired tasks. In such cases, we can develop a paradigm for providing agents with parametric information about the desired trajectories, from which the agents can compute their inputs. Precisely, as is very common in tracking applications (see e.g. [10]), we can limit the number of *modes* (signal frequencies) contained in the desired trajectories, and hence simply communicate the modes and trajectory-initial conditions. There are of course several plausible paradigms for limiting the motions of the agent. The following is one such paradigm:

Definition 6: A **standard formation-F_0 tracking task** is a formation-F_0 tracking task in which the expansion, rotation, and translation signals $a(t)$, $\theta(t)$, $p_x(t)$ and $p_y(t)$ are set to desired values at specified times $t = 0, t_1, t_2, \ldots$, and are interpolated linearly between these desired values at intermediate times.

Consider a standard formation-F_0 tracking task in a stabilizable PDIN. We notice that we can specify the formation-tracking signals for this task simply by specifying the signals at time $0, t_1, t_2, \ldots$. Thus, it is automatic that we can distribute these formation-tracking signals by distributing the values of the signals only at the specified times. Let us formalize this notion:

Theorem 5: Consider a stabilizable PDIN that must complete a standard formation-F_0 tracking task. If each agent i is provided with the formation-tracking signals $(a(t), \theta(t), p_x(t), p_y(t))$ at times $0, t_1, t_2, \ldots$ as well as the nominal-formation parameters $\widehat{r}_{ix}$, $\widehat{r}_{iy}$, $\widehat{r}_{jx}$, $\widehat{r}_{jy}$, and $\widehat{r}_{0x}$, $\widehat{r}_{0y}$, $j \in U(i)$, then the agent can compute its TIS and hence the tracking task can be achieved.

In the interest of space, we omit detailed discussion of formation tracking in saturating PDINs.

Briefly, it is easy to show that tracking in saturating PDINs is possible, if an (unsaturated) PDIN with the same observation topology is stabilizable, and further the formation trajectory signals change slowly enough. In the case where standard formation tracking is required, we can always meet this additional condition by making the times at which the formation-tracking signals are specified sufficiently far apart.

2.4 Information Flow through Adaptation

So far, we have considered distribution of desired-trajectory information through explicit (though hopefully sparse) overlayed communication. In some applications, such additional communication may be infeasible, or tracking of an agent with hidden or unknown trajectory information may be desired. In this final section, we explore, in the case of formation-tracking, how to achieve trajectory information flow distribution through *adaptation*, i.e. how agents can identify and follow trajectories based on signatures in the actuation signals rather than through explicit overlayed communication.

Our contention is that adaptation is most feasible for formation-tracking tasks where agents can only follow trajectories having a small number of unknown parameters. In such cases, agents without knowledge of desired trajectory information can infer this information from transients in their actuation signals and hence adapt their TIS appropriately.

Let us explore information flow through adaptation in the context of an example. Specifically, we consider the case that agents in a double-integrator network aim to adapt their TIS to follow a turn action while maintaining formation. In particular, we assume that the agents are initially moving with constant velocity. A leader-agent is given a command to turn left or right (in both cases, the new desired trajectory is a straight-line path, and is perpendicular to the original path before the

turn), and the other agents must infer their TIS and hence follow this new path. In addition, we make two assumptions on the communication/sensing topology of the double-integrator network: 1) the leader-agent measures its own state in an absolute frame (and so can follow its new TIS), and 2) in the corresponding *network graph* (as defined in Section 2.2), there exists at least one path from the leader-agent to each other agents in the network. The network graph otherwise may be arbitrary. With such possible trajectories and sensing topology, we shall show that the TIS can be adaptively-determined by each agent and hence the formation-tracking can be achieved.

We stress that we are considering a special case with only one type of trajectory—translation with constant velocity—and two possible new trajectories. It should also be noted that such a limited set of trajectories means that there are a small number of unknown parameters corresponding to that set. We note, however, that if these trajectories are generated by exosystems as defined in section 2.1, the set of initial conditions for the new exosystem may be infinite because the turn (left or right) can happen at any time.

Before presenting our method for TIS adaptation, it is useful to define *the shortest path from the leader-agent to another agent* (i.e. an agent that has no measurement on its own state in an absolute frame), *the weight of an edge*, and *the weight of a path.*

Definition 7: **A shortest path from the leader-agent to another agent i** is a directed path from the leader-agent to agent i in the network graph, which has no more edges than any other path from the leader-agent to agent i.

We note that there may be more than one shortest path from the leader-agent to a non-leading agent.

Definition 8: **The weight of a directed edge** from vertex j to vertex i is defined to be $\mathbf{k}_i^T G_{ij}$,

where $\mathbf{k}_i$ is agent i's position control gain, and G_{ij} is the jth column of the graph matrix G_i.

We note that edge weights are well-defined even when agents have multiple observations.

Definition 9: **The weight of a directed path** is the product of edge weights along that path.

We are ready to show how the non-leading agents can adapt their TIS appropriately. It turns out that, after the leader-agent makes a turn, a transient appears in the actuation signal of each non-leading agent. The sign of this transient is closely related to the acceleration of the leader-agent and the weights of the shortest paths from the leader-agent to that particular non-leading agent. Formally, we have the following theorem:

Theorem 6: Suppose that a group of n agents in a PDIN are moving in a straight line with constant velocity, and a leader-agent is given a command to make a left or right turn. When the leader-agent makes the turn, each of the non-leading agents can infer the direction and time of the turn from their actuation-signal transients, and hence the appropriate TIS can be adaptively determined by each non-leading agent. Specifically, immediately after the leader-agent turns, we have that $sgn(u_i) = sgn(p_i u_l)$, where u_i is the acceleration of agent i (in x- or y-direction), u_l is the acceleration of the leader-agent (in x- or y-direction), and $p_i \neq 0$ (which can always be done by adjusting the control gain) is the sum of weights of the shortest paths from the leader-agent to agent i.

Proof: We prove this theorem by first explicitly computing the derivatives of non-leading agent i's actuation signal (in the $x-$direction) with respect to time, and then arguing that the sign of non-leading agent i's acceleration (in the $x-$direction) immediately after the leader-agent turns is determined by the ξth-order derivative, where ξ is the length of the shortest path from the leader-agent to agent i. Since the proof for the $y-$direction is exactly the same, we only need to consider the $x-$direction.

From the control law described in Theorem 1, the actuation signal of a non-leading agent i (in the $x-$direction) is:

$u_i = \mathbf{k}_i G_{ii}(r_i - \bar{r}_i) + \alpha \mathbf{k}_i G_{ii}(v_i - \bar{v}_i) + \mathbf{k}_i \Sigma_{j \in U(i)} G_{ij}(r_j - \bar{r}_j) + \alpha \mathbf{k}_i \Sigma_{j \in U(i)} G_{ij}(v_j - \bar{v}_j)$

For convenience, let $u_i = f_1(i) + f_2(i) + f_3(i) + f_4(i)$, where

$f_1(i) = \mathbf{k}_i G_{ii}(r_i - \bar{r}_i)$

$f_2(i) = \alpha \mathbf{k}_i G_{ii}(v_i - \bar{v}_i)$

$f_3(i) = \mathbf{k}_i \Sigma_{j \in U(i)} G_{ij}(r_j - \bar{r}_j)$

and $f_4(i) = \alpha \mathbf{k}_i \Sigma_{j \in U(i)} G_{ij}(v_j - \bar{v}_j)$.

Therefore, we have the following expressions:

$u_i^{(m)} \mid_{t=t_T} = (f_1^{(m)}(i) + f_2^{(m)}(i) + f_3^{(m)}(i) + f_4^{(m)}(i)) \mid_{t=t_T}$

$u_i^{(m+1)} \mid_{t=t_T} = (\frac{1}{\alpha} f_2^{(m)}(i) + f_2^{(m+1)}(i) + \frac{1}{\alpha} f_4^{(m)}(i) + f_4^{(m+1)}(i)) \mid_{t=t_T}$

From the above expressions, together with the fact that $f_1^{(0)}(i) \mid_{t=t_T} = f_2^{(0)}(i) \mid_{t=t_T} = f_3^{(0)}(i) \mid_{t=t_T} = f_4^{(0)}(i) \mid_{t=t_T} = 0$ and $f_2^{(m+1)}(i) \mid_{t=t_T} = \alpha \mathbf{k}_i G_{ii} u_i^{(m)} \mid_{t=t_T}$, we can recursively show that $u_i^{(\beta)} \mid_{t=t_T} = f_4^{(\beta)}(i) \mid_{t=t_T}$, if $u_i^{(m)} \mid_{t=t_T} = 0$ for $m = 0, \ldots, \beta - 1$.

Hence, we can recursively express the βth-order derivative of u_i with respect to time as:

$u_i^{(\beta)} \mid_{t=t_T} = \alpha \Sigma_{j_1 \in U(i)} \mathbf{k}_i G_{i j_1} u_{j_1}^{(\beta-1)} \mid_{t=t_T}$

$\ldots$

$u_{j_{\beta-1}}^{(1)} \mid_{t=t_T} = \alpha \Sigma_{j_\beta \in U(j_{\beta-1})} \mathbf{k}_{j_{\beta-1}} G_{j_{\beta-1} j_\beta} u_{j_\beta} \mid_{t=t_T}$

If $\beta < \xi$, then the leader-agent is not in the set $U(j_{\beta-1})$. Therefore, $u_i^{(\beta)} \mid_{t=t_T} = 0$

If $\beta = \xi$, then the leader-agent is in the set $U(j_{\xi-1})$. Therefore, $u_i^{(\xi)} \mid_{t=t_T} = \alpha^\xi p_i u_l \mid_{t=t_T} \neq 0$, where p_i is the sum of weights of the shortest paths from the leader-agent to agent i, and u_l is the actuation signal of the leader-agent.

Since $u_i^{(\beta)} \mid_{t=t_T} = 0$ for all $\beta < \xi$ and $u_i^{(\xi)} \mid_{t=t_T} \neq 0$, the sign of agent i's acceleration u_i (in

the $x-$direction) immediately after the leader-agent turns is the same as the sign of $u_i^{(\xi)} \mid_{t=t_T}$, i.e. $sgn(u_i) = sgn(p_i u_l)$. $\square$

In many applications, the non-leading agents have observations on their positions relative to other agents. In such cases, a **grounded Laplacian** sensing architecture (see [2] for definition, and note that in our case, only the leader-agent has absolute position measurement) is used, and we have the following corollary:

Corollary 1: Suppose that n agents in a PDIN with grounded Laplacian sensing topology are moving in a straight line with constant velocity, and a leader-agent is given a command to make a left or right turn. Then, immediately after the leader-agent turns, the sign of each non-leading agent i's acceleration (in the x- and y-direction) is the same as that of the leader-agent (in the x- and y-direction).

Proof: From the definition of grounded Laplacian sensing architecture, all non-zero off-diagonal entries of the full graph matrix G are negative. In addition, it is known that the eigenvalues of KG (where K is the block-diagonal control gain matrix) are in the OLHP if and only if the non-zero entries of K are negative. Therefore, the weight of path from the leader-agent to each non-leading agent i is always positive, i.e. $p_i > 0$. According to Theorem 6, we have $sgn(u_i) = sgn(u_l)$, and the corollary results. $\square$

We thus notice that the sign of the leader-agent's acceleration, in the $x-$ and $y-$direction, can be inferred from the non-leading agents' local actuation signals. Such information is sufficient for non-leading agents to determine the trajectory of the leader-agent, and hence the appropriate TIS can be adapted by each non-leading agent. Specifically, we use the following mechanics to adapt the TIS for each non-leading agent i:

First, we set a reasonable threshold for the actuation signal of each non-leading agent i. When agent i is moving with constant velocity as desired, the amplitude of its actuation signal is close to zero and should be smaller than the threshold. In addition, if a turn is made by the leader-agent, each threshold should be exceeded by the amplitude of the transient in each non-leading agent i's actuation signal.

Then, from the sign of u_i, agent i is able to determine the acceleration, and hence the new trajectory, of the leader-agent. Specifically, agent i can adaptively determine its own new TIS by using its knowledge of the new velocity (speed and direction) of motion. In doing so, we use the old TIS value when the threshold is exceeded as the initial condition for the new TIS. It should be noted that the TIS of agent i is not switched to the newly-adapted TIS until all non-leading agents' threshold have been exceeded. That is, after agent i's threshold is exceeded and its new TIS is determined, a reasonable time delay is desired before the new TIS is used by agent i. If such a time delay is absent, the new behaviors of some non-leading agent i may affect the TIS adaptation of some non-leading agent j whose actuation signal transient has not exceeded its threshold. In this case, agent j may not detect the new trajectory, or even determine an incorrect new trajectory. It should also be mentioned that, since we simply use the old TIS values when threshold is exceeded as initial conditions for the new TIS, some error may be introduced to the formation after the turn is made (since there is a delay between the time of the turn and the time when the threshold is exceeded). However, we also note that this error can be made arbitrarily small by making the threshold small.

We then present an example that uses the above mechanics.

A double-integrator network containing four agents has the full graph matrix $G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -1 & 0 & 0 & 1 \end{bmatrix}$.

Before a turn is made, agents move along the $x-$axis with constant velocity as desired. When the x-coordinate of the leader-agent (agent 1) reaches $x = 11$, agent 1 receives a command signal to turn right. By setting the thresholds of non-leading agents to be 0.02, they are able to detect the new trajectory of the leader-agent, and change their TIS to follow the new trajectory. The simulation result is shown in Figure 2.3.

Connection to [14] In [14], a platoon of vehicles are performance-regulated (i.e. a performance statistic of the platoon, e.g. variance, tracks a desired statistic trajectory), while individual agents' trajectories are not explicitly set and are unknown until the closed-loop system is simulated. In order to realize performance-regulation, an exosystem is used to generate the desired platoon performance and an external device measures the actual platoon performance. This information is broadcast to all the vehicles. This method involves relatively little communication.

Our method of tracking through adaptation is different from [14] in that each agent's trajectory and the nominal formation are explicitly specified, and hence the performances of both individual agents and the group are known. Also, our method of tracking through adaptation does not require an external device as in [14]. Instead, each agent can obtain needed information locally (and hence no explicit communication is needed). However, our method has the disadvantage that it only may work for a limited set of possible trajectories.

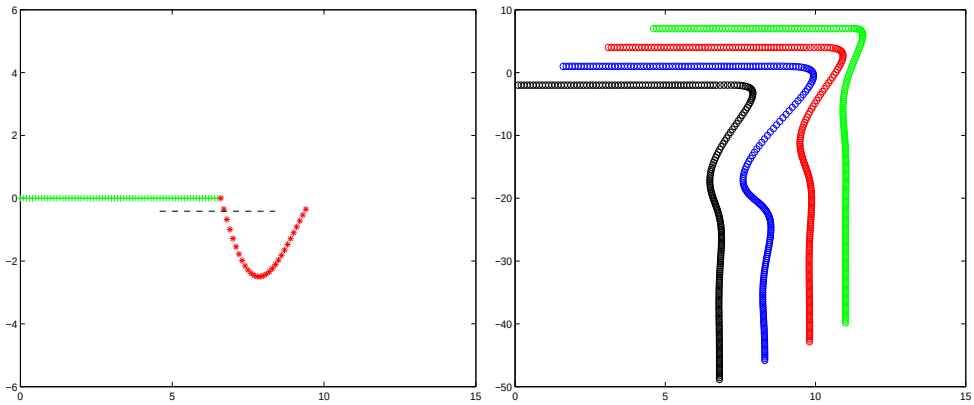

Fig. 2.3: Information flow through adaptation is demonstrated. The left plot shows the transient signature in non-leading agent 2's $y-$direction acceleration in response to leader-agent 1's change in trajectory. The right plot shows tracking through adaptation.

3. SECURITY OF DISTRIBUTED ALGORITHMS FOR COMMUNICATING-AGENT NETWORKS

3.1 Problem Formulation

In this chapter, we study how a communicating agent network's graph and decentralized controller/algorithm should be designed, in order that it can simultaneously achieve an asymptotic performance task and a security task. We begin with a careful formulation of the system model, two types of dynamic tasks, and the notions of security.

3.1.1 System Model Formulation

Let us consider a network of n agents, labeled $1, \ldots, n$. We assume that each agent has internal dynamics described by a single integrator, i.e. $\dot{x}_i = u_i$ where x_i is the state of agent i and u_i is its control input. For convenience, we define a **full state vector** or **state vector** $\mathbf{x}^T \triangleq \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}$ and **input vector** $\mathbf{u}^T \triangleq \begin{bmatrix} u_1 & \ldots & u_n \end{bmatrix}$. In this notation, $\dot{\mathbf{x}} = \mathbf{u}$.

Each agent makes m_i observations, each of which is a linear combination of the agents' states. That is, each agent has available a length-m_i **observation vector** $\mathbf{y}_i = G_i\mathbf{x}$, where the $m_i \times n$ matrix G_i is termed the topology matrix for agent i. We also find it convenient to assemble the observations made by the agents into a single **full observation vector** $\mathbf{y}^T = \begin{bmatrix} \mathbf{y}_1^T & \ldots & \mathbf{y}_n^T \end{bmatrix}$. We

note that $\mathbf{y} = G\mathbf{x}$, where $G = \begin{bmatrix} G_1 \\ \vdots \\ G_n \end{bmatrix}$ is termed the full topology matrix. We envision that this full topology matrix G can be designed by the user in some applications but is pre-set in others; we shall consider both cases in our development.

We refer to the internal dynamics, observation paradigm, and decentralized feedback control law together as a **single-integrator network**.

3.1.2 Dynamic Tasks Formulation

Formation Tasks In some cases, a group of agents desire to complete a dynamic task cooperatively. And hence the agent's state may be required to settle at a target value. This type of dynamic task is known as a *formation task*. For the purpose of this chapter, we formally define a formation task as the following:

Definition 10: A single-integrator network can be formation stabilized to $\mathbf{x}_0$ if a proper linear time-invariant static controller can be designed for it, so that the state x_i of each agent is globally asymptotically convergent to x_{i0} for each agent i.

Our previous work (see [2]) has shown that a single-integrator network can be formation stabilized to $\mathbf{x}_0$ if and only if it can be formation stabilized to $(0, 0)$. Therefore, the achievement of a formation task is equivalent to the achievement of system stabilization.

Hence our aim here is to design a decentralized control law $\mathbf{u} = KG\mathbf{x}$ such that the eigenvalues of KG are in the OLHP, where K is a $n \times \sum_i m_i$ matrix and has the form $\begin{bmatrix} \mathbf{k}_1^T & & \\ & \ddots & \\ & & \mathbf{k}_n^T \end{bmatrix}$, where $\mathbf{k}_i$ is a vector of length m_i.

In our previous work, we have identified quite-general conditions on G for which a (block-diagonal) gain matrix K can place the eigenvalues of KG in the OLHP (see [12]). We have also identified condtions on G such that all feedback matrices within a set place the eigenvalues of KG in the OLHP [12], by invoking the notion of *D-stability* (see [17]). We shall use these conditions as needed in our ensuing development.

Tracking Tasks In a formation task, as each agent's state settles to the target value, there is no requirement on the specific trajectory of the agent's state, and the dynamic task can be completed only through cooperation between agents. However, in some other dynamic tasks, each agent desires to complete a dynamic task. In this case, the agent's state must follow a specified trajectory. This type of dynamic task is referred to as a *tracking task*. We assume that the trajectory to be followed by agent i, denoted $\overline{x}_i$ and called the **reference signal** for agent i, is one that can be generated by a *linear exosystem*: $\overline{x}_i = \mathbf{d}_i^T \mathbf{w}_i$, where $\mathbf{w}_i$ satisfies a linear differential equation $\dot{\mathbf{w}}_i = Q_i \mathbf{w}_i$. It is well known (see e.g. saberi) that almost any trajectory can be represented in this way, and hence this assumption is not in any way restrictive to our development. Also, since our goal is to track the reference signal in an asymptotic sense (see below), we are not interested in reference signal components with trivial steady-state and hence assume w.l.o.g. that the eigenvalues of Q_i are in the CRHP (see e.g. [10]).

We find it convenient to assemble the exosystems of the multiple agents into a single exosystem. In particular, let us define the **exosystem state vector** as $\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$. The exosystem state

vector satisfies the differential equation $\dot{\mathbf{w}} = Q\mathbf{w}$, where $Q \triangleq \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_n \end{bmatrix}$. Also, let us define a tracking vector $\overline{\mathbf{x}} = \begin{bmatrix} \overline{\mathbf{x}}_1 \\ \vdots \\ \overline{\mathbf{x}}_n \end{bmatrix}$. Notice that $\overline{\mathbf{x}} = D\mathbf{w}$, where $D \triangleq \begin{bmatrix} \mathbf{d}_1^T & & \\ & \ddots & \\ & & \mathbf{d}_n^T \end{bmatrix}$. We also find it convenient to define an **error vector** $\mathbf{e}$ as $\mathbf{e} = \mathbf{x} - \overline{\mathbf{x}}$.

Our goal is to build a decentralized controller for the network so that each agent's state x_i asymptotically tracks the reference signal $\overline{x}_i$, or in other words so that $\lim_{t\to\infty} \mathbf{e} = 0$. In completing the tracking task, we assume that each agent can generate or obtain any required *excitation signals* (signals that actuate the agent so that its state can follow the tracking vector*). We also enforce that the controller is decentralized, so that each agent i can only use the measurements $\mathbf{y}_i$ to achieve control. In this work, we limit ourselves to static linear controllers, i.e. ones in which the the control input at a given time is an affine function of the observations at that time. Such controllers are often very easy to implement, and hence are applicable to a wide range of control tasks. The following theorem gives conditions under which a decentralized static linear controller can be used to achieve the tracking task, and specifies that controller that does so.

Theorem 7: The decentralized control law

$$\mathbf{u} = K\mathbf{y} - KG\overline{\mathbf{x}} + DQ\mathbf{w} \tag{3.1}$$

achieves tracking—i.e. $\lim_{t\to\infty} e(t) = 0$ for all initial conditions $\mathbf{x}(0)$ when this controller is used—if and only if the eigenvalues of KG are in the OLHP.

* Please see Chapter 2 for a discussion of how the exosystem dynamics can be propagated to the agents with minimal information flow, hence permitting agents to generate their required excitation signals

Proof:

Sufficiency:

When the above control law is used, $\dot{\mathbf{x}} = KG\mathbf{x} - KG\bar{\mathbf{x}} + DQ\mathbf{w}$. Let us make a change of state variable $\mathbf{z} = \mathbf{x} - \bar{\mathbf{x}}$. Noting that $\dot{\bar{\mathbf{x}}} = D\dot{\mathbf{w}} = DQ\mathbf{w}$, we recover $\dot{\mathbf{z}} = KG\mathbf{z}$. It is classical that if the eigenvalues of KG are in the $OLHP$, $\mathbf{z} \to 0$ as $t \to \infty$, which implies that $\mathbf{x} \to \bar{\mathbf{x}}$ as $t \to \infty$. Hence the tracking task is achieved.

Necessity:

If the eigenvalues of KG are not in the $OLHP$, then at least one mode of $\dot{\mathbf{z}} = KG\mathbf{z}$ are persistent. Hence for initial conditions $\mathbf{z}(0)$ that are not perpendicular to all the left eigenvectors corresponding to the sustaining modes, $\mathbf{z}(t)$ does not converge to zero as $t \to \infty$. Therefore, $\mathbf{x}$ does not converge to $\bar{\mathbf{x}}$ as $t \to \infty$, which implies that the controller of this form cannot achieve tracking task when the eigenvalues of KG are not in $OLHP$, so the necessity is proved. □

We thus propose to use controllers of the form given in Equation 3.1 to achieve the tracking task. We note that a single-integrator network is parametrized by the topology matrix G, the **feedback matrix** or **gain matrix** K, and the exosystem parameters Q and D.

3.1.3 Security Formulation

In order to define the notions of security, it is helpful to define some terminology for groups of agents and their observations:

- We use the term **agent set** for an unordered set of distinct agents (i.e., a set with distinct elements chosen from $1, \ldots, n$).
- For a given agent set $\mathcal{S}$, we use the notation $\mathbf{y}_{\mathcal{S}}$ for a vector comprising the observations of the agents in $\mathcal{S}$. WLOG, we place the observations in increasing order with respect to the

agent labels. That is, for an agent set $\mathcal{S} = \{i_1, \ldots, i_m\}$, where $i_1 < \ldots < i_m$, $\mathbf{y}_{\mathcal{S}} = \begin{bmatrix} \mathbf{y}_{i_1} \\ \vdots \\ \mathbf{y}_{i_m} \end{bmatrix}$.

We refer to the vector $\mathbf{y}_{\mathcal{S}}$ as the **observation vector for the agent set** $\mathcal{S}$. We also define the **topology matrix for S** as $G_{\mathcal{S}} = \begin{bmatrix} G_{i_1} \\ \vdots \\ G_{i_m} \end{bmatrix}$. We note that $\mathbf{y}_{\mathcal{S}} = G_{\mathcal{S}}\mathbf{x}$.

We are now ready to define the notions of a security requirement: the (S, M) *security* and *overall security*.

The Notion of **(S,M) Security** Conceptually, we envision a security requirement as being achieved if some important aspect of the closed-loop state dynamics (e.g., the state of a particular important agent, the starting point or home base of the agents, or perhaps some aggregate statistic of the network) cannot be determined by an adversary that obtains the observations of a particular agent or group of agents. We envision a security task as being completed, or security as being achieved, if an appropriate Boolean function of security requirements is achieved. Throughout our development, we make the (worst-case) assumption that the adversary has knowledge of the single-integrator-network parameters ((K, G) for formation tasks, and (K, G, D, Q) for tracking tasks), and can use this information to try to estimate the state from the obtained observations.

Formally, we define the notion of (S,M) security as follows:

Definition 11: Consider the situation that an agent set S has been violated, and so an adversary has observed $\mathbf{y}_{\mathcal{S}}$ over a time interval $[t_1, t_2]$. If the adversary cannot uniquely compute any statistic of the form $z(t) = \mathbf{m}^T\mathbf{x}(t)$, for all initial conditions, for any $\mathbf{m}$ in the range space of a matrix M and any $t \in [t_1, \infty)$, then we say that the single-integrator network achieves the security requirement

(S, M). Otherwise (i.e., if even a single such statistic can be computed at a particular time), the (S,M) security requirement is not met, or in other words the system is (S,M) insecure. We shall often use an indicator variable $Z(S, M)$ to describe whether or not the (S,M) security requirement has been met (i.e., $Z(S, M) = 1$ if the (S,M) security requirement is met, and $Z(S, M) = 0$ otherwise).

We define an (S,M) security task based on a set of security requirements:

Definition 12: Consider a set of p security requirements $(\mathcal{S}_1, M_1)$, ..., $(\mathcal{S}_p, M_p)$, and let $Z_1, \ldots, Z_p$ indicate whether or not each (S,M) security requirement is achieved by the single-integrator network. Then a (S,M) security task T is said to be achieved if the Boolean function $f_T(Z_1, \ldots, Z_p)$ is unity.

The Notion of Overall Security In addition to the notion of (S,M) security defined above, in some cases, we are also interested in another level of security—*overall security*, which is formally defined as follows:

Definition 13: Consider the situation that an agent set S has been violated, and so an adversary has observed $\mathbf{y}_S$ over an interval $[t_1, t_2]$. If the adversary cannot uniquely determine $\mathbf{x}(t)$ at any time $t \in [t_1, \infty)$, then we say that the single-integrator network is **overall secure** with respect to set S.

Therefore, a system is overall secure as long as the adversary is not able to uniquely determine the values for all the state variables. We note that the overall security is *weaker* than any particular (S,M) security requirement. As we will see through examples in the following subsection, both (S,M) security and overall security are useful in practice.

It is also worth noting that overall security equivalent to an (S,M) security task that is an union of n (S,M) security requirements: $f_T(Z_1, \ldots, Z_n) = Z_1 \cup Z_2 \cup \ldots \cup Z_n$, where Z_i indicates whether

the security requirement (S, e_i) is achieved, and where e_i is an $n \times 1$ vector with the ith entry equal to 1 and all other entries equal to 0.

It turns out that the notions of security defined above are closely tied to the control-theoretic notion of *observability*, as we shall make explicit in the subsequent analysis. The notion of observability is naturally tied to the notions of security because it formalizes whether or not the initial value and hence trajectory of part of a system's dynamics can be determined from a set of observations.

3.1.4 Many Examples

The following examples illustrate the usefulness of our notions of security.

Consider a single-integrator network with 4 agents, divided into two groups: group A consists of agent 1 and 2, and group B consists of agent 3 and 4. Group A needs to accomplish task A in a relatively safe environment, and this task is to be accomplished with agents in group A staying close to each other. And group B needs to finish task B in a relatively hostile environment. In addition, task A is considered more important than task B.

In this case, agents in group B are more likely to be violated by the adversary, and one needs to protect group A even when agents in group B are violated. Therefore, the observations available to the adversary are the outputs of agent 3 and agent 4. the corresponding (S,M) security requirement formulation should have $S = \{3, 4\}$ and $M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$. Then it is desired that the parameters of the network (K, G, D, Q) are designed such that the security requirement

(S,M) is achieved.

Consider the situation that 3 unmanned aerial vehicles, which cannot be found by radar, are detecting the ground situation of a hostile area. Once a vehicle is violated by the adversary, it quits the detection task. The mission is considered as successful if at least one vehicle completed the detection task.

Assume that the third vehicle detects the most dangerous area, and hence it is relatively vulnerable. In this case, one would wish to design the system such that the adversary cannot violate both agent 1 and agent 2, if agent 3 is violated. Hence we are considering the overall security of the UAV group with $S = \{3\}$.

3.2 Tests of (S,M) Security

Our aim in this section is to develop tests for whether a particular single-integrator network achieves the desired dynamic task and a particular (S,M) security task T. In the problem-formulation section, we have already given conditions under which the tracking task is achieved, so our primary aim in this section is to develop a test for (S,M) security, i.e. to identify conditions under which an (S,M) security task is achieved. Note, however, that if we can test whether particular (S,M) security requirements (S, M) are met, then we can also test whether an (S,M) security task T is achieved since we need only evaluate the Boolean function $f_T()$ to decide. Thus, we focus on developing a test for whether a particular (S,M) security requirement is met along with the tracking task.

3.2.1 Tests for (S,M) Security of Tracking Tasks

Our tests are based on the control-theoretic notion of observability. We kindly request that readers who are unfamiliar with these notions consult a graduate text on linear systems or control theory, e.g. [15].

Before presenting the tests, we find it convenient to define some further notation regarding linear-algebraic constructs associated with the single-integrator network. In particular, for a given single-integrator network and security requirement (S, M), we consider the following:

- We define the **closed-loop system matrix** as $A = \begin{bmatrix} KG & DQ - KGD \\ 0 & Q \end{bmatrix}$. We note that A describes the entire closed-loop dynamics of the single-integrator network, i.e. $\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{w}} \end{bmatrix} = A \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix}$.
- We use the notation V_{uo} for any basis of the unobservable subspace corresponding to the pair $(\begin{bmatrix} G_S & 0 \end{bmatrix}, A)$. We note that such a basis can be constructed in a systematic way from A and G_S, see e.g. [15].

We are now ready to present a test for whether a single integrator network achieves a tracking task and an (S,M) security requirement:

Theorem 8: A single-integrator network achieves the desired tracking task as well as the security requirement (S, M) if and only if 1) the eigenvalues of KG are in the OLHP, and 2) $\begin{bmatrix} M^T & 0 \end{bmatrix} V_{uo}$ has full row rank.

Proof: It has been shown that the tracking task can be achieved if and only if there exists a gain matrix K such that the eigenvalues of KG are in the OLHP. Therefore, it only remains to show

that the security requirement (S, M) is achieved if and only if $\begin{bmatrix} M^T & 0 \end{bmatrix} V_{uo}$ has full row rank. We assume that $M \in R^{n \times r}$, where $r \leq n$.

Let $V_1, V_2, \ldots, V_j$ be the eigenvectors corresponding to unobservable modes of the system, and $V_{j+1}, \ldots, V_{j+l}$ be the eigenvectors corresponding to the observable modes. Then $\begin{bmatrix} \mathbf{m}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(0) \\ \mathbf{w}(0) \end{bmatrix} =$

$\begin{bmatrix} \mathbf{q}^T M^T & 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 & \ldots & V_j & V_{j+1} & \ldots & V_{j+l} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{j+l} \end{bmatrix}$. $z(t) = \mathbf{m}^T \mathbf{x}(t)$ cannot be computed if and only if $\mathbf{m}^T \mathbf{x}(0)$ has non-zero projection in the unobservable subspace for some initial conditions.

If $\begin{bmatrix} M^T & 0 \end{bmatrix} \begin{bmatrix} V_1 & \ldots & V_j \end{bmatrix}$ has full row rank, then $\begin{bmatrix} \mathbf{q}^T M^T & 0 \end{bmatrix} \begin{bmatrix} V_1 & \ldots & V_j \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_j \end{bmatrix}$ is non-zero for some initial conditions, and hence the security requirement has been achieved.

If $\begin{bmatrix} M^T & 0 \end{bmatrix} \begin{bmatrix} V_1 & \ldots & V_j \end{bmatrix}$ does not have full row rank, then there exists a vector $\mathbf{q}$ such that $\begin{bmatrix} \mathbf{q}^T M^T & 0 \end{bmatrix} \begin{bmatrix} V_1 & \ldots & V_j \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_j \end{bmatrix} = 0$ for all initial conditions. Therefore, the security requirement is not achieved.

In the case that (S, M) security is not achieved, if the adversary has observed $y_{\mathcal{S}}$ over the time interval $[0, t_1]$, then by using the observer canonical form of a linear system, we find that the statistic $z(t)$ can be computed as:

$$z(t) = \begin{bmatrix} \mathbf{m}^T & 0 \end{bmatrix} P\Phi(t,0)[\int_0^{t_1} \Phi^T(t,0) P^T \begin{bmatrix} G_S & 0 \end{bmatrix}^T \begin{bmatrix} G_S & 0 \end{bmatrix} P\Phi(t,0)dt]^{-1} \cdot [\int_0^{t_1} \Phi^T(t,0) P^T \begin{bmatrix} G_S & 0 \end{bmatrix}^T \mathbf{y}_S(t)dt] \tag{3.2}$$

where $P = \begin{bmatrix} \mathbf{p}_1^T & \dots & \mathbf{p}_l^T \end{bmatrix}$, $\mathbf{p}_1, \dots, \mathbf{p}_l$ are the l independent rows of the observability matrix (which has a rank of l), and where $\Phi(t,0) = e^{A_{11}t}$, in which A_{11} is the system matrix of the observable subsystem in observer canonical form (see [15] for details), i.e, $\begin{bmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{bmatrix}$. $\square$

Consider a single-integrator network with two agents, with full graph matrix $G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. These agents are tracking two sinusoids. Suppose that agent 2 has been violated, so $S = \{2\}$ and $G_{\mathcal{S}} = \begin{bmatrix} 0 & 1 \end{bmatrix}$. Let $D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $Q = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $K = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$, and $\begin{bmatrix} \mathbf{x}(0) \\ \mathbf{w}(0) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$.

If we choose $M = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, we find that the matrix $\begin{bmatrix} M^T & 0 \end{bmatrix} V_{uo}$ has full row rank, i.e., has a rank of 1. And hence the (S,M) security requirement is achieved.

If we choose $M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, we find that the matrix $\begin{bmatrix} M^T & 0 \end{bmatrix} V_{uo}$ has a rank of 1, i.e., it does not have full rank. And hence the (S,M) security requirement is not achieved.

Theorem 8 gives us a way of testing whether a designed system is (S,M) secure or not, and makes it easier for the designer to design an (S,M) secure system.

3.2.2 Tests for (S,M) Security of Formation Tasks

In a formation task, the state of the network converges to a target value. It has been shown that a formation task can be achieved if and only if the network is stabilizable (see [2]). Therefore, we need only consider the case that the target value is the origin.

We note that the (S,M) security for formation tasks is a simplified version of the (S,M) security for tracking tasks. Therefore, analogous to theorem 8, we have the following theorem for (S,M) secure formation test. We shall use the notation V'_{uo} to denote a basis of the unobservable subspace corresponding to the pair (G_s, KG).

Theorem 9: A single-integrator network achieves the desired formation task as well as the security requirement (S,M) if and only if 1) the eigenvalues of KG are in the OLHP, and 2) $M^T V'_{uo}$ has full row rank.

Proof: It is clear that a formation task is achieved if and only if condition 1) holds.

Analogous to the proof of theorem 8, $\mathbf{m}^T\mathbf{x}(t)$ cannot be computed if and only if condition 2) holds. □

3.3 Tests of Overall security

In this section, we identify the conditions under which a single-integrator network performing a tracking or formation task is overall secure. We note that an overall security requirement is weaker than an (S,M) security requirement. In fact, by making a comparison between (S,M) security and overall security, we have the following straightforward theorem:

Theorem 10: A single-integrator network performing a tracking or formation task achieves overall security with respect to S, if it achieves security requirement (S,M) for any M.

In the subsequent discussion, we shall consider overall security for both tracking and formation tasks.

3.3.1 Tests for Overall Security of Tracking Tasks

Overall security of a tracking task is achieved when at least one state variable of the network cannot be computed by the adversary. This can be tested by considering multiple security requirements. Formally, we have the following theorem:

Theorem 11: A single-integrator network achieves the desired tracking task as well as the overall security requirement if and only if 1) the eigenvalues of KG are in the OLHP, and 2) there exists at least one vector I_i such that $\begin{bmatrix} I_i^T & 0 \end{bmatrix} V_{uo} \neq 0$, where I_i is an n-component indicator vector with the unity entry at the ith position, and V_{uo} is a basis of the unobservable subspace of the pair $(\begin{bmatrix} G_S & 0 \end{bmatrix}, A)$.

Proof: The tracking task is achieved if and only if condition 1) holds. It remains to show that overall security is achieved if and only if condition 2) holds.

Sufficiency: If condition 2) holds, then it follows that there exists at least one state variable of the system that cannot be computed by the adversary, and hence the network is overall secure.

Necessity: If condition 2) does not hold, i.e., none of $\begin{bmatrix} I_i^T & 0 \end{bmatrix} V_{uo}$, for $i = 1, \ldots, n$, has full row rank, then it is clear that all the state variables can be constructed by the adversary, and it follows that the system is not overall secure. □

3.3.2 Tests for Overall Security of Formation Tasks

From the definition of overall security, we note that a single-integrator network completing a formation task is overall secure with respect to S as long as the system is not observable to the

adversary when agent set S is violated. Formally, we have the following straightforward theorem:

Theorem 12: A single-integrator network is overall secure with respect to S if and only if the pair ($G_{\mathcal{S}}$,KG) is unobservable.

In general, an overall secure formation task is more easily achieved than any (S,M) secure formation requirement. In fact, we can show that the overall security of a formation task can be preserved even when the adversary has some control over the controller. Formally, we have the following theorem:

Theorem 13: For a single-integrator network performing a formation task, if the output feedback gain matrix K is designed such that the overall security with respect to S is achieved, then the system remains overall secure with respect to S even when the feedback gain of the violated agents can be changed by the adversary.

Proof: If a system is not observable, then there exists an unobservable subspace. In addition, this subspace is an invariant subspace of the system matrix KG, where G is the full graph matrix of the network. Furthermore, it is well-known that this invariant subspace contains at least one eigenvector of the system matrix KG, denoted by V. Therefore, there is at least one eigenvector of KG that is in the unobservable subspace of the system. And we accordingly have $KGV = \lambda V$, where λ is the corresponding eigenvalue of V.

Since the eigenvector V is in the unobservable subspace, we have $G_S V = 0$, where G_S is the observation vector for the agent set S. Together with the fact that $KGV = \lambda V$, we can see that the $i_1th, i_2th, \ldots, i_jth$ entries of V are zeros, where $i_1, i_2, \ldots, i_j$ are the labels of agents in set S.

Suppose the $i_1th, i_2th, \ldots, i_jth$ rows of K are changed by the adversary, and the new gain matrix is K'. Since K' is block diagonal, it follows that $K'GV = \lambda V$, i.e., V remains the eigenvector of

$K'G$. Then it is clear that V is still in the unobservable subspace of the system.

Hence we can conclude that the system stays overall secure when the feedback gain of the violated agents can be changed by the adversary. □

It is worth noting that the overall security of a formation task is *robust* in that the adversary is not able to change the overall security status by changing the controllers of the violated agents.

3.4 Estimation of $z(t)$ through observer

When (S,M) security requirement or overall security with respect to S is achieved, the adversary is not able to compute $z(t)$ exactly due to the presence of unobservable modes. In this case, the adversary may try to estimate the statistic $z(t)$ using an observer. From the observer canonical form, one can obtain an observable subsystem. The state of this observable subsystem can be estimated sufficiently fast using an observer. This is illustrated by the following example.

We consider the system described in example 3.2.1. Suppose an observer is constructed by the adversary, and the (S,M) security requirement is $(2, 1)$, i.e., agent 1 should be secure when agent 2 is violated. The system matrix of the original system is $\mathbf{A} = \begin{bmatrix} -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & -2 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$, the input is zero, and the output is $y_{\mathcal{S}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix}$. Using the linear transformation

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & -2 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -5 & 0 & 0 & 0 \end{bmatrix} \mathbf{z},$$ one can obtain the observer canonical form. The system becomes

$$\begin{bmatrix} \dot{\mathbf{z}}_o \\ \dot{\mathbf{z}}_{uo} \end{bmatrix} = \begin{bmatrix} -2 & 9 & -16 & 0 & 0 & 0 \\ 0 & -2 & 5 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_o \\ \mathbf{z}_{uo} \end{bmatrix} \tag{3.3}$$

$$\mathbf{y}_{\mathcal{S}} = \begin{bmatrix} 1 & -2 & 4 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_o \\ \mathbf{z}_{uo} \end{bmatrix} \tag{3.4}$$

The pair (C_o, A_o) is observable, where $C_o = \begin{bmatrix} 1 & -2 & 4 \end{bmatrix}$, and $A_o = \begin{bmatrix} -2 & 9 & -16 \\ 0 & -2 & 5 \\ 0 & -1 & 2 \end{bmatrix}$. An observer can therefore be constructed for this subsystem.

The dynamics of the observer is known to have the form $\dot{\hat{\mathbf{z}}}_o = \mathbf{A}_o\hat{\mathbf{z}}_o + \mathbf{L}(\mathbf{y}_{\mathcal{S}} - C_o\hat{\mathbf{z}}_o)$, where $\mathbf{L}$ is a proportional gain defined as $\mathbf{L} = \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix}^T$. We shall look at the corresponding error equation

$$\dot{\tilde{\mathbf{z}}}_o = (A_o - \mathbf{L}C_o)\tilde{\mathbf{z}}_o \tag{3.5}$$

It is known that one can obtain arbitrary eigenvalues of $(A_o - \mathbf{L}C_o)$ by choosing the proportional gain $\mathbf{L}$. This implies that if one chooses the eigenvalues to be in the OLHP and far away from the imaginary axis, the observer can estimate $\mathbf{z}_o$ sufficiently fast. For instance, if one wants to place all the three eigenvalues at -10, then $\mathbf{L} = \begin{bmatrix} 147.6 & 447.8 & 194 \end{bmatrix}^T$. And the simulation result is shown in Figure 3.1. It can be seen that the estimation has an overshoot at the beginning, and converges to the real value very fast.

Therefore, the adversary is able to obtain satisfactory estimations by simply constructing a linear device, saving a lot of computation efforts. Besides, if $z(t)$ contains only observable and detectable modes, then it can be estimated by the adversary asymptotically.

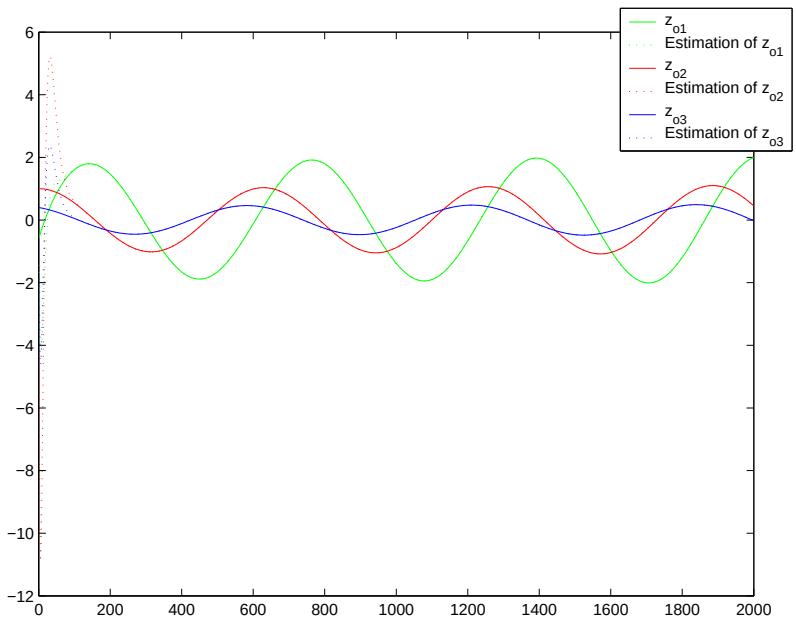

Fig. 3.1: The actual values and estimations of $\mathbf{z}_o$. (Noting that $\mathbf{z}_{uo}$ cannot be estimated)

3.5 Connection to [16]

There has been other papers that deal with the state estimation of a decentralized system. It is interesting to look into [16], which shows that an agent i can transmit the initial condition information available to agent i to agent j by some signaling strategy, if there exists an edge from agent i to agent j in the *graph* (see [16]).

When this result is applied to our work, we find that agent i can transmit the initial condition information available to itself to agent j, if there is an edge from agent i to agent j in the network graph, and the adversary has access to the inputs of the agents. In this case, the unobservable subspace of the system is reduced (see [16]). We note that this is an extreme situation, since the adversary has control over the inputs of the agents, which indicates that the agents are not functioning properly.

To illustrate the impact of signaling on security situation, we consider the following two cases:

Case 1: the adversary is able to specify external input signals for each channel. In this case, the initial condition information available to agent j may be transmitted to agent i by using some signaling strategy, provided that agent j is an upstream agent of agent i. Furthermore, if there is no loop in the network graph, and agent i is violated, then all initial conditions of upstream agents of agent i can be computed by the adversary.

Case 2: the adversary has available the observations of violated agents, but has no control over the inputs of agents in the network, i.e. the adversary cannot apply any signaling strategies. In this case, the initial condition information available to agent i cannot be transmitted to its downstream agents, and the unobservable subspace cannot be reduced.

With the following example, we illustrate security situation difference when signaling can or cannot be used.

Suppose a single-integrator network has the full graph matrix $G = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & -1 & 1 \end{bmatrix}$, and the output feedback gain matrix is chosen to be $K = \begin{bmatrix} -1 & & \\ & -2 & \\ & & -1/2 \end{bmatrix}$. When agent 3 is violated, we have the following two cases:

1. If the adversary can use signaling strategies, the unobservable subspace is reduced to $K_1 \cap K_2 \cap K_3$, where K_i is the unobservable subspace of agent i. Therefore, the adversary can determine initial conditions of all three agents.

2. If no signaling is allowed, the rank of the observability matrix is 2, which means that the adversary cannot determine all initial conditions of the three agents.

4. CONCLUSION

In this thesis, we have extended the controller/algorithm for formation in double-integrator networks (see [2]) to permit tracking of temporal signals. We have delineated the role played by the sensing topology of the network, and also identified the trajectory information that needs to be distributed. A low-information-flow tracking controller has also been designed.

We have motivated and defined the notions of security, i.e., (S,M) security and overall security. Conditions have been developed to test the two notions of security in both tracking and formation tasks. The two cases–the statistic $z(t)$ does and does not contain sustaining unobservable modes–are differentiated by considering the situation that the adversary uses a Luenberger estimator. Finally, we connected our work to [16].

BIBLIOGRAPHY

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

[2] S. Roy, A. Saberi, and K. Herlugson, 'Formation and Alignment of Distributed Sensing Agents with Double-Integrator Dynamics', IEEE Press Monograph on *Sensor Network Operations*, 2004 (in press).

[3] A. Williams, G. Lafferriere, and J. J. P. Veerman, "Stable motions of vehicle formations," in *Proceedings of the 2005 Conference on Decision and Control.*

[4] P. Ogren, E. Fiorelli, and N. Leonard, "Cooperative control of mobile sensor networks," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, 2004.

[5] D. J. Stilwell, B. E. Bishop, and C. A. Sylvester, "Redundant manipulator techniques for partially decentralized path planning and control of a platoon of autonomous vehicles," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 4, Aug. 2005.

[6] E. Davison, "The robust decentralized control of a general servomechanism problem," *IEEE Transactions on Automatic Control*, vol. 21, no. 1, pp. 14-24, Feb. 1976.

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

[8] S. Roy, K. Herlugson, and A. Saberi, "A control-theoretic perspective on distributed discrete-valued decision-making in networks of sensing agents," to appear in the *IEEE Transactions on Mobile Computing.*

[9] R. O. Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions in Automatic Control*, vol. 49, pp. 1520-1533, Sep. 2004.

[10] A. Saberi, A. Stoorvogel, and P. Sannuti, *Output Regulation and Control Problems with Regulation Constraints*, Springer-Verlag, 1999

[11] M. E. Fisher and A. T. Fuller, "On the stabilization of matrices and the convergence of linear iterative processes," *Proceedings of the Cambridge Philosophical Society*, pp. 417–425, 1956.

[12] S. Roy, J. Minteer, and A. Saberi, "Some new results on stabilization by scaling," accepted to the *American Control Conference*, 2006.

[13] S. Roy, A. Saberi, and K. Herlugson, "A control-theoretic perspective on the design of distributed agreement protocols,", Submitted to the *Journal of the Association for Computing Machinery.*

[14] D. J. Stilwell, and B. E. Bishop, "Redundant manipulator techniques for path planning and control of a platoon of autonomous vehicles," *Proceedings of the 41st IEEE Conference on Decision and Control*, pp. 2093–2098, 2002.

[15] W. J. Rugh, "*Linear System Theory*, Prentice Hall, 1996

[16] H. Kobayashi, and T. Y. Yoshikawa, "Graph-Theoretic Approach to Controllability and Local-

izability of Decentralized Control Systems," *IEEE Transactions on Automatic Control*, vol. 27, No. 5, Oct. 1982.

[17] D. Hershkowitz, "Recent direction in matrix stability," *Linear Algebra Appl.* 171 (1992), 161-186.