SYNCHRONIZATION OF MULTIPLE ROTATING SYSTEMS

By

JAMES SLADE

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

Master of Science in Electrical Engineering

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering & Computer Science

August 2007

To the Faculty of Washington State University:

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

_____

Co-Chair

_____

Co-Chair

_____

_____

ACKNKOWLEDGEMENTS

First, I would like to acknowledge my advisors, Sandip Roy and Ali Saberi. They guided me in my research and advanced my knowledge in the classroom. Their help has been greatly appreciated and very beneficial.

I would also like to thank the other members of my thesis committee, Anton A. Stoorvogel and Bernard Lesieutre, for their time and insightful feedback.

Also, I would like to thank Dr. John Morrison at Montana Tech of The University of Montana. Without his support and daily reminder of graduate school, I wouldn't be where I am today.

Lastly, I would like to thank my parents and the rest of my family and friends for their support not only throughout my college carreer, but in everything I do. For their unlimited love and encouragement, I am forever grateful.

SYNCHRONIZATION OF MULTIPLE ROTATING SYSTEMS

Abstract

by James Slade, M.S.
Washington State University
August 2007

Co-Chair: Sandip Roy

Co-Chair: Ali Saberi

In recent years, distributed control systems have been the topic of a vast amount of research and literature due to their advantages over their centralized counterparts. A significant advantage is that a network of cummunicating agents can work in a cooperative manner to achieve complex dynamic tasks. Tasks that have garnered much attention are those of *agreement*—those in which agents with differing initial opinions/states must reach a common opinion. This thesis examines agreement tasks for networks whose agents have states that are cyclic entities. Since the states are cyclic entities, even the simplest control laws can result in *mode lock*, an undesirable stable equilibrium where the states of the agents are not alligned. There have been methods proposed in the past for designing controllers that eliminate mode lock. One such method is a nonlinear control law, which we deconstruct in detail, and use our findings to influence a linear static control design. We carefully explain mode lock and propose new heuristic algorithms for linear static control design to eliminate mode lock.

Contents

List of Figures

To my amazing parents

Chapter 1

Introduction

A variety of algorithms and controllers are needed for *communicating-agent networks*, i.e. networks of autonomous agents with distributed communication/sensing capabilities that coordinate to complete a global task (e.g., [16,18,21–24]). Within this broad domain, algorithms/controllers for *agreement* tasks—those in which agents with differing initial opinions/states must reach a common opinion, that depends in a prescribed manner on the initial values—have been of particular interest ([16, 18, 21, 22, 24]). The agreement problem originated in the computer science community [19], but has been approached by control theorists in the communicating-agent network arena in recent years, with motivation from e.g. sensor fusion and autonomous vehicle coordination applications. The control-theoretic approach suggests using local *averaging*, or weighted averaging rules so as to achieve a global agreement. This formulation permits development of fully distributed algorithms, and asymptotic/transient analysis of these algorithms using linear system theory together with graph theory.

An interesting subset of agreement problems are those in which the states of the systems are cyclic, or rotational quantities. Nominally, this topic of rotational agreement seems closely related to agreement in a linear frame; however, the differences between the two are significant. The cyclic nature of the states makes the system nonlinear, and hence introduces additional equilibrium points

as well as complex dynamic phenomena. In this article, we pursue the problem of decentralized controller design for cyclic agreement. More broadly, our study of cyclic agreement serves as a context for us to study graph-design ideas for networks of communicating agents, an important component of the on-going effort to design high-performance controllers for networks.

Let us briefly review the literature in cyclical agreement. Agreement tasks in which agents' states or observations are cyclical quantities have been motivated in several fields. For instance, the references [4] and [26] describe rotational dynamics in such application areas as environmental biology (specifically, regarding coordination of e.g. fireflies) and physics. Similar problems also arise in distributed clocking of multi-processor computers [11, 20], and in aligning the headings of multiple autonomous vehicles [16].

Based on these motivations, a control-theoretic methodology for agreement in a rotational frame has been advanced in the articles [11,13,16,20,27]. Specifically, the article [20] notes the possibility for mode-locking when measurements are phase differences (rather than differences on a linear scale), and propose a non-linear control scheme which is essentially based on destabilizing the offending (mode-locked) equilibria. While their control scheme is effective for regular meshes, it cannot easily be generalized to arbitrary networks because controller synthesis requires computation of the smallest phase difference between connected agents in all mode-locked states. We also note that this non-linear methodology in general only *destabilizes* the mode-locked configurations (equilibria) and does not eliminate them. The recent efforts [16] have instead taken the perspective that, by measuring phase differences appropriately, the rotational problem can be rephrased as a translational one. However, their approach is restricted to only Laplacian topologies, and also suffers from the difficulty that a small change in an agent's phase can necessitate an extensive response by multiple agents (specifically, they must rotate 180^o to reach equilibrium again).

In this work, we propose a new control scheme for agreement in a rotational frame, and explicitly pursue network topology design for fast agreement. In Chapter 2, we give a brief overview of agreement tasks and some of the application areas for rotational agreement. Also, we state some graph theoretic results that were used in analyzing our model. In Chapter 3, we describe our network model and introduce the rotational agreement problem. Before describing our new control methodology, we first make precise that only tree networks are not subject to mode-locking, and report on a strategy for finding all the mode-locked states for a general network topology. These results motivate us to select observations so as to form a tree network, and in turn use a linear control law or algorithm (Chapter 4). Thus, we address the problem of designing a tree (or selecting a spanning tree in a graph) to maximize the rate of settling to agreement of the algorithm.

Chapter 2

Background Information

In this chapter, we give a brief explanation of agreement tasks and some of the past work done on the topic. Historically, most of this work has been done for systems with translational states, as opposed to rotational states. Also, we explain a couple of the areas where our work can be applied. Finally, we state some graph theoretic results, since network structure and dynamics are of great importance for our results.

2.1 Agreement Tasks

As embedded systems become more sophisticated, decentralized control and agreement tasks are becoming more popular. Decentralized control has many advantages over its centralized counterpart. A network of systems working together cooperatively offers more efficiency and operational capability than a single system working solo. Applications of these decentralized systems are unlimited, as they are already employed in a number of fields (see [21]).

The goal for these tasks is to create an update rule so that each system in the network, which start with different initial states/opinions, has a common state. These states can have physical meaning, such as in systems governed by Newton's Law. In contrast, the states can have only a

computational meaning, such as the opinion of a sensor. The update rules range from simple linear static rules to complex dynamic rules. Each has its advantages for certain applications, and will remain a topic of research for some time.

2.2 Application Areas

This section describe a couple of networks whose systems have states that are cyclic entities. Our work was heavily focused on [20], which applied the topic to networks of distributed clocks.

2.2.1 Distributed Clock Networks

A major application for the title topic is in the area of distributed clock networks, such as a multiprocessor. In the past, clock phases have been alligned by dispersing the output of a central oscillator over a tree like network. This method is troublesome in that it requires repeaters at certain intervals, and this method is not robust, has poor reliability, and high clock skew.

Recently, distributed clocking has been a common way of alligning clocks. Multiple local clocks replace the central oscillator, and an algorithm that uses phase differences between neighboring clocks alligns the all of the clock phases. Distributed clocking is much more reliable than a tree like network. The tree like network requires repeaters at necessary intervals, which add to clock skew. Also, if anything fails in a tree like network, everything downstream of the fault is affected. In a distributed system, faults are isolated.

Distributed clocking has many advantages over past methods. Since clock signals are highly succeptible to noise, a tree like network can be problematic. In order to reduce the noise, all clock lines must be properly terminated and shielded. For large networks, the expense of this can be

extremely high. In distributed clocking, each clock is generated locally so the need for a tree like network is eliminated.

Another advantage of distributed clocking is that clock skew is kept to a minimum. High clock skew can cause unpredictable and undesirable behavior. Since clocks are generated locally in distributed clocking, skew is low.

Distributed clocking is also a robust mehtod. The tree like network is fixed after manufacture so if it is found to be inadequate, there is nothing to be done. However, in distributed clocking additional processors are just added to the existing network.

The design of controllers for aligning the phase of each of these local clocks has been of great interest. In [20], they propose a method to allign the phase of multiple clocks. The articles [7, 15] detail a physical implementation of the method proposed in [20]. In [11], a least squares approach is taken, and in this case clocks are alligned for a wireless network.

2.3 Analyzing Network Structure And Network Dynamics

A network is a system with many distinct but interacting components. We wish to study properties of the components or of the whole network that change with time (network dynamics). Graph theory and algebraic graph theory are the tools used to analyze networks. Graph theory is used to analyze the network structure, while algebraic graph theory connects network dynamics to network structure.

2.3.1 Graph Theory

A graph $G = (V, E)$ consists of a set of nodes (vertices) $V(G) = \{1, 2, \cdots, n\}$, where n is the number of nodes, and an edge set E(G). The edge set E(G) contains pairs of distinct nodes, with each pair of nodes representing an edge in the graph. An edge represents a physical connection from one node, say node A, to another node, say node B, in which node B is able to receive information from node A. If there is no order associated with pairs of nodes, i.e. an edge from node i to node j is the same as an edge from node j to node i, the graph is undirected. Otherwise, the graph is a directed graph. Also, the graph is a weighted graph, one in which each edge has a scalar weight associated with it. This weight can be any real number. In the special case where each edge weight is one, the graph is an unweighted graph.

Often, it is beneficial to illustrate graphs. We draw nodes as dots and edges as lines between nodes. In a network, the individual components are represented as nodes and interactions between the components are represented as edges. A graph is planar if it we can draw its nodes and edges in a two dimensional space so that no two edges cross.

The two vertices associated with an edge are called the ends of the edge. If there is an edge from node i to node$_J$, node i is the tail and node j is the head. A node that is an end of an edge is said to be incident to the edge. Two nodes are said to be adjacent (or neighbors) if there is an edge between them, or equivalently if there is an edge whose ends are the two nodes. A group of nodes with no edges in common (none of which are neighbors) are called independent.

The degree of a node is the number of edges incident to the node (the number of neighbors of the node). If each node is adjacent to every other node, the graph is complete. If the degrees of all of the nodes are equal, the graph is regular.

A subgraph is another graph whose vertex set $V^{'}$ is a subset of the vertex set V of the original

graph. The subgraph has an edge set $E^{'}$ which is a subset of the edge set E of the original graph and the ends of each edge in $E^{'}$ are in the vertex set $G^{'}$. An induced subgraph is a subgraph in which, for any pair of nodes i and j in the set V$^{'}$, the edge $\{i, j\}$ is in the set E$^{'}$ only if it is in the set E.

A path is a sequence of nodes such that two successive nodes in the sequence are adjacent. The first node in the sequence is denoted the start node, while the last node in the sequence is denoted the end node. A cycle is a path in a graph in which the start node and the end node are the same. If there is path between any pair of nodes, the graph is connected. A tree graph, or acyclic graph, is a graph that doesn't contain any clcles. A spanning tree is a subgraph such that $V^{'} = V$ and the subgraph is a tree graph.

An edge cut set (or cut set) is a set of edges whose removal results in a graph that is not connected. A vertex cut set is a set of nodes removal results in a graph that is not connected. A minimum cut set is the least number of edges whose removal results in a graph that is not connected.

Here are some aggregate measures and relations for vertices and edges. The number of nodes and edges in a graph are denoted by $|V|$ and $|E|$, respectively. The degree of node v is denoted by $d(v)$. The minimum degree, the maximum degree, and the average degree are

$$\delta(G) = \min_{v \in V} d(v)$$

$$\Delta(G) = \max_{v \in V} d(v)$$

$$d(G) = \frac{1}{|V|} \sum_{i=1}^{|V|} d(i)$$

respectively. These measures can be ordered in the following manner: $\delta(G) \leq d(G) \leq \Delta(G)$. Another expression for the average degree is $d(G) = \frac{2|E|}{|V|}$. The degree distribution $f_G(i)$ gives the fraction of the nodes in G that have degree i for $i = 0, 1, \cdots, |V| - 1$. A graph cannot be

constructed by only knowing the degree distribution. However, degree distributions are important because real systems have some special degree distributions that might tell us something about network dynamics.

Here are some aggregate measures for paths and cycles. The distance between two nodes is the least number of edges in a path between the two nodes. The eccentricity of a node i is the greatest distance between i and any other node. The minimum distance among all cycles is called the girth. The maximum distance among all cycles is called the circumference. The minimum eccentricity of any node is the radius. The maximum eccentricity of any node is the diameter.

2.3.2 Algebraic Graph Theory

All of the entities above help us study the structure of a network. However, they don't tell us anything about network dynamics. Algebraic graph theory relates network structure to network dynamics. Algebraic graph theory is a field that associates matrices with graphs and characterizes properties of these matrices (most notably eigenvalues). The eigenvalues of these matrices give us insight into important properties of the network, such as stability and settling rate.

Many real world systems have dynamics that can be modeled with either a combinatorial Laplacian matrix or an adjacency matrix. For instance, the combinatorial Laplacian can be used to model the dynamics of a resistor capacitor circuit. The combinatorial Laplacian is an n x n matrix defined as follows:

$$L_{ij} = \begin{cases} \sum_{j \neq i} k_{ij}, & \text{i = j} \\ -k_{ij}, & \text{i} \neq \text{j, i and j adjacent} \\ 0, & \text{Otherwise} \end{cases},$$

where k_{ij} is the weight of edge $\{i, j\}$. The adjacency matrix can be used to model dynamics in molecular networks, such as protein-protein interactions. The adjacency matrix is defined as

follows:

$$A_{ij} = \begin{cases} k_{ij}, & \{i,j\} \in \mathrm{E(G)} \\ 0, & \text{Otherwise} \end{cases}$$

In the case where the combinatorial Laplacian and adjacency matrices are unweighted, i.e. $k_{ij} = 1, \forall\ i, j$, the matrices can be related to each other. Let D be the diagonal matrix with the degree of each node on the diagonal, that is

$$D = \begin{pmatrix} d(1) & & & 0 \\ & d(2) & & \\ & & \ddots & \\ 0 & & & d(n) \end{pmatrix}.$$

The combinatorial Laplacian can be found from L = D - A.

Now that the graph structure has been related to matrices, we need a tool for analyzing these matrices. The Courant Fisher Theorem is a common method used to find or bound eigenvalues of these matrices. The Courant Fisher Theorem is a way to find eigenvalues of a symmetric matrix through an optimization. Since the eigenvalues of a symmetric matrix are real and nonnegative, they can be ordered as $0 \leq \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{n-1}$.

The smallest eigenvalue of a symmetric matrix A can be found by the Courant Fisher Theorem, which is

$$\lambda_0 = \min_{||x||=1} x^T A x,$$

where $\mathbf{x}$ is an n component column vector. The vector $\mathbf{x}$ that minimizes the expression is the eigenvector associated with the eigenvalue. Conversely, we can find the largest eigenvalue λ_{n-1} with the following optimization:

$$\lambda_{n-1} = \max_{||x||=1} x^T A x.$$

In the case that the matrix A is a combinatorial Laplacian, the expression for λ_0 is equivalent to

$$\lambda_0 = \min_{||x||=1} \sum_{\{i,j\} \in E(G)} (x_i - x_j)^2.$$

From this, it is easy to see that if $x_i = x_j, \forall\ \{i,j\} \in \mathrm{E(G)}$, then $\lambda_0 = 0$ and the associated eigenvector is a vector of all ones, which is denoted by $\vec{\mathbf{1}}$. If the graph is connected, all of the rest of the eigenvalues of A are strictly greater than λ_0. However, if the graph is not connected, the eigenvalue at zero is repeated. The number of disconnected subgraphs is the number of times the zero eigenvalue is repeated.

The rest of the eigenvalues of A can be found in successive order. Let v_0 be the eigenvector corresponding to eigenvalue λ_0. Then, λ_1 can be found from

$$\lambda_1 = \min_{||x||=1, x \perp v_0} x^T A x.$$

In order to find λ_i, all of the previous eigenvalues and eigenvectors need to be found. Let W be the space spanned by all of the previous eigenvectors, i.e. $W = [v_0, v_1, \cdots, v_{i-1}]$. We can find λ_i from

$$\lambda_i = \min_{||x||=1, x \perp W} x^T A x.$$

In a similar manner, we could start at λ_{n-1} and work in decreasing order.

While using the Courant Fisher Theorem to find all eigenvalues and eigenvectors of a symmetric matrix can be a daunting task, finding just a couple of eigenvalues and eigenvectors is straightforward. Often times we are very concerned with λ_1, which Fiedler termed the algebraic connectivity of the graph. This eigenvalue dominates the network dynamics, and thus has been the topic of a large amount of literature ([1,8,9,14]). Since λ_0 and its associated eigenvector are already known, we can find λ_1 from

$$\lambda_1 = \min_{||x||=1, x \perp \vec{\mathbf{1}}} x^T A x.$$

Later, it will be our goal to design graphs, specifically tree graphs, with large algebraic connectivity.

Since we are only concerned with tree graphs, we will only present those results. For a tree graph, $\lambda_1 \leq 1$, and has equality if and only if the graph is a star graph (see Chapter 4). Also, for a tree with at least six nodes that is not a star graph, $\lambda_1 < 0.49$ (see [1]). Later, we will use these results to design algorithms for finding tree graphs that yield good settling rates.

Chapter 3

Synchronization and Mode Locking for Rotational Systems

We take the following approach to studying the problem of synchronizing rotating systems. First, Section 3.1 develops the single integrator model that we use, and poses the syncrhonization problem. In Section 3.2, we explain an algorithm for finding mode lock equilibrium points for a network, and also give necessary and sufficient conditions for the existence of mode lock.

3.1 Model And Problem Formulation

We first formulate our model for communicating agents with cyclic states and observations by describing the internal dynamics of each agent and the decentralized observation topology of the network. We then describe the agreement task, and hence mention several control paradigms for achieving agreement. We focus especially on static linear control in this paper, and so we elaborate on the closed loop dynamics in this case.

3.1.1 Communicating Agent Network Model

We study a network of n **agents**, where each agent i has a scalar state x_i, $x_i \in \Re$. We define the **network state** as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

For our purposes, this state represents a cyclic quantity, i.e. one that is indistinguishable modulo 2π. Since the states are indistinguishable modulo 2π, we find it useful to define **angles** $z_i = ((x_i + \pi)\ mod\ 2\pi) - \pi$. Notice that each of the angles z_i are mappings of the states x_i to the unit circle, i.e. to the interval $[-\pi, \pi]$. For convenience, we define the **network angle** as

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}.$$

We model each agent as having single integrator internal dynamics, that is

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

where u_i is the input of agent i. This agent model is appropriate both for hardware systems, such as processor clocks [11,20], and in network algorithms for agreement (see [16,18,21–24]). The input is generated by the feedback controller or algorithm, as explained below.

We pursue agreement in a decentralized setting, one where each agent only has partial information of the network state. Specifically, for applications with cyclic states, it is sensible that

the information available to an agent consists of angle differences between itself and its neighbors. Formally, let us define the *relative angle* between two angles a and b as

$$d(a,b) = ((\pi + a - b) \ mod \ 2\pi) - \pi. \tag{3.2}$$

A graphical illustration of $d(a,b)$ is shown below in Figure 3.1. We define the set containing the

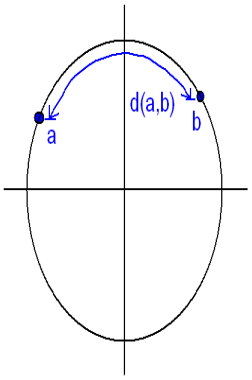

Fig. 3.1: Illustration of d(a,b).

neighboring agents of an agent i as $\mathcal{N}(i)$. We assume that each agent i makes observations

$$y_{ij} = d(z_i, z_j), \forall \ j \in \mathcal{N}(i). \tag{3.3}$$

That is, each agent measures the relative angle to each of its neighbors. The observation topology defines a network of systems with single integrator dynamics, which we refer to as a single integrator network (SIN).

3.1.2 The Synchronization Problem

Our goal is to find a control law (algorithm) such that each agent's angle asymptotically approaches a common value. In the most general sense, we want to design a controller for each agent i, i.e. a rule

$$u_i = f_i(y_{ij}(\tau), t), j \in \mathcal{N}(i), 0 \leq \tau \leq t, \tag{3.4}$$

that generates the input u_i from the past and current observations made by agent i to achieve this goal of synchronization.

First, let us more formally define the notion of synchronization. We say an agent i is *stationary* if

$$f_i(y_{ij}(\tau), t) = 0 \ \forall \ t, j \in \mathcal{N}(i), 0 \leq \tau \leq t, \tag{3.5}$$

i.e. the angle z_i is no longer evolving with time. The SIN is stationary if all of the agents are stationary, or equivalently the SIN is at an equilibrium point. The SIN is *synchronized* if it is stationary and each angle z_i is the same. When the SIN is synchronized, we say it is in a *synchronized state*.

Our goal is to find a control law such that the SIN asymptotically approaches a synchronized state from any initial state. Formally, we say *global asymptotic synchronization* has been achieved if, for any intitial state $x(0)$, the network angle $z(t)$ asymptotically approaches a synchronized state.*

Global asymptotic synchronization requires a synchronized state to be the only stationary configuration. If the SIN is stationary, and at least one angle z_i is different from the other angles,

*The definition of global asymptotic synchronization is concerned with attractivity of the equilibrium; in fact, our controllers achieve the stronger notion of asymptotic stability, i.e. attractivity together with stability in the sense of Lyapunov. We have not been explicit about this in our formulation to avoid a lot of ugly notation.

the SIN is said to be in *mode lock*. We call these equilibrium points *mode-lock equilibrium points*. If mode-lock equilibrium points exist, our goal of global asymptotic synchronization cannot be achieved since the SIN's state will remain at the mode-lock equilibrium point if it starts there. Further, just as the network angle asymptotically approaches a synchronized state, the network angle can also asymptotically approach a mode-lock equilibrium point for a certain set of initial conditions.

Mode lock is a critical issue in the SIN. Because y_{ij} is a nonlinear function of x, even a simple LTI control law f can result in mode-lock equilibrium points, as illustrated below in Example 1.

Example 1

Consider using the simple and widely used averaging control law. This control law aims to drive the angle of each agent i to the average of its neighbors' angles by actuating the agent with an arithmetic average of its relative angle observations. When each agent has the same angle, the input to each agent is zero and the network remains stationary, so the SIN has a synchronized state. Unfortunately, the SIN may in general also have mode-lock equilibrium points.

For instance, consider a network of four agents where $\mathcal{N}(1) = \{2, 4\},\ \mathcal{N}(2) = \{1, 3\},\ \mathcal{N}(3) = \{2, 4\},\ \mathcal{N}(4) = \{1, 3\}$. It is easily verified that if $z_1 = 0,\ z_2 = \pi/2,\ z_3 = -\pi,\ z_4 = -\pi/2$, each $u_i = 0$, thus the SIN is stationary. Since the SIN is stationary and at least one angle z_i is different from the other angles, the SIN is in mode lock so global asymptotic synchronization cannot be achieved. □

Since an averaging scheme cannot always achieve global asymptotic synchronization, other control methods are needed. One method, proposed in [20], makes all of the mode-lock equilibrium points unstable and hence aims to allow synchronization from all but a measure-zero set of points. This is accomplished by using a piece-wise linear feedback law. This law is similar to the averaging

scheme in Example 1. However, if there is a measurement y_{ij} such that $|y_{ij}| > \pi/2$, the sign of the measurement is flipped before it is averaged with the measurements from other neighbors. Also, a weighted-average control is permitted. This makes the mode lock equilibrium points unstable while maintaining the stability of the equilibria corresponding to the synchronized state.

While this method of using a piece-wise linear feedback law often works beautifully, it does have some drawbacks. The SIN is still susceptible to mode lock (albeit unstable mode lock), despite the use of the piece-wise linear feedback law. Also, the nonlinearity of the control law can create some interesting and potentially undesirable behavior. Not only does the piece-wise linear control law make mode-lock equilibrium points unstable, but it also creates new mode-lock equilibrium points and destroys existing mode-lock equilibrium points, as illustrated below in Example 2. Also, there is as yet no justification that asymptotic synchronization results from other (non-mode-lock) initial conditions: chaotic or periodic behavior could result, for instance. Finally, destabilization of mode lock can only be guaranteed for certain special observation topologies.

Example 2

Consider a network of six agents with the following observation topology: $\mathcal{N}(1) = \{2, 4\}$, $\mathcal{N}(2) = \{1, 3, 5\}$, $\mathcal{N}(3) = \{2, 6\}$, $\mathcal{N}(4) = \{1, 5\}$, $\mathcal{N}(5) = \{2, 4, 6\}$, $\mathcal{N}(6) = \{3, 5\}$. If the averaging control law described in Example 1 is used, the network has a mode-lock equilibrium point at

$$z^T = \begin{pmatrix} 0 & -\frac{14\pi}{15} & -\frac{2\pi}{3} & \frac{14\pi}{15} & -\frac{2\pi}{15} & -\frac{2\pi}{5} \end{pmatrix}.$$

When the piece-wise linear control law is used instead, we find that the SIN has a new mode-lock equilibrium point at

$$z^T = \begin{pmatrix} 0 & -\frac{8\pi}{9} & \frac{2\pi}{3} & \frac{8\pi}{9} & -\frac{2\pi}{9} & \frac{2\pi}{9} \end{pmatrix},$$

and further that the equilibrium point

$$z^T = \begin{pmatrix} 0 & -\frac{14\pi}{15} & -\frac{2\pi}{3} & \frac{14\pi}{15} & -\frac{2\pi}{15} & -\frac{2\pi}{5} \end{pmatrix}$$

has disappeared. Thus, the piece-wise linear control law has destroyed a mode-lock equilibrium point and created a new mode-lock equilibrium point.

The existence of mode-lock equilibrium points is problematic. While these equilibrium points are in fact unstable, the possibility of mode lock still exists. If the initial network angle starts at an unstable mode-lock equilibrium point, it will remain there. Also, if the network starts "close" to one of these unstable mode-lock equilibrium points, it will remain there for a period of time before it starts to converge to the synchronized state (if indeed convergence can even be proven), see Figure 3.2.

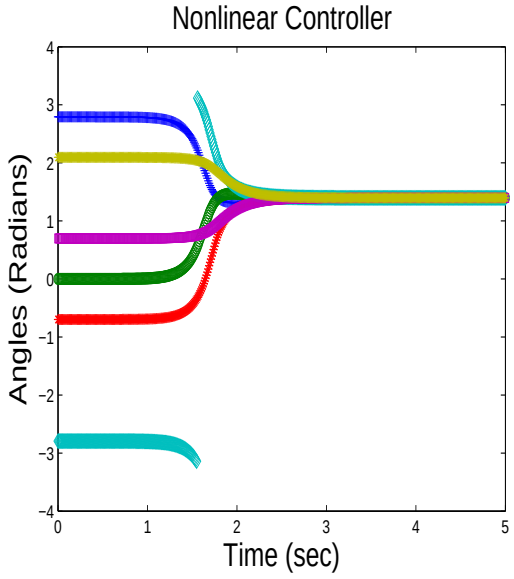

Fig. 3.2: The network remaining close to an unstable mode-lock equilibrium point before it starts to converge to the synchronized state.

Another flaw of the piece-wise linear control law is the existence of a stable manifold around

a mode-lock equilibrium point. This results from the initial network angle lying in some lower dimensional space. If the initial network angle lies on this manifold, the network angle approaches an unstable mode-lock equilibrium point, as illustrated below in Example 3.

Example 3

Consider a network of three agents with the following observation topology: $\mathcal{N}(1) = \{2, 3\}$, $\mathcal{N}(2) = \{1, 3\}$, $\mathcal{N}(3) = \{1, 2\}$. For the initial network angle

$$z^T = \begin{pmatrix} \frac{\pi}{4} & -\frac{11\pi}{16} & -\frac{13\pi}{16} \end{pmatrix},$$

a plot of the angles evolving in time is shown below in Figure 3.3. The initial condition lies on

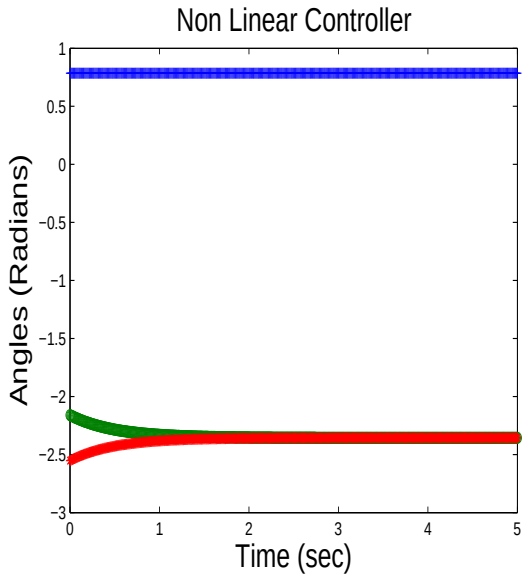

Fig. 3.3: Initial condition lies on stable manifold around mode-lock equilibrium point and network angle approaches unstable mode-lock equilibrium point.

the manifold, and the network angle asymptotically approaches an unstable mode lock equilibrium point. □

3.1.3 Closed Loop Dynamics

In general, each agent in the SIN uses a controller that maps past and present observations y_{ij} to the input u_i. Here, our focus is on static linear control, so we write the closed-loop dynamics for this case. The static linear control law for agent i is the following:

$$u_i = -\sum_{j \in \mathcal{N}(i)} k_{ij} y_{ij}, \tag{3.6}$$

where k_{ij} is a gain that is to be designed. Now, substituing Equation 3.6 into Equation 3.1, we get

$$\dot{x}_i = -\sum_{j \in \mathcal{N}(i)} k_{ij} y_{ij}. \tag{3.7}$$

To write the closed loop dynamics in terms of state variables, we substitute Equation 3.3 into Equation 3.7 and then substitute Equation 3.2 into that result to find

$$\dot{x}_i = -\sum_{j \in \mathcal{N}(i)} k_{ij}[((\pi + z_i - z_j)\ mod\ 2\pi) - \pi], \tag{3.8}$$

which is a nonlinear system dynamic. To characterize the nonlinear system, the modulo operator needs to be investigated. It is easy to verify that the following equalities hold:

$$((\pi + z_i - z_j)\ mod\ 2\pi) - \pi = \begin{cases} z_i - z_j + 2\pi, & z_i - z_j < -\pi \\ z_i - z_j, & -\pi \leq z_i - z_j \leq \pi \\ z_i - z_j - 2\pi, & z_i - z_j > \pi \end{cases} \tag{3.9}$$

Thus, we can model the system as a switched linear system (with state-dependent switching). We observe that the instantaneous input is linear in the angle difference between two agents i and j. Only the additive constant changes with the angle difference in Equation 3.9, between zero and $\pm\ 2\pi$.

We find it convenient to rewrite the model's closed loop dynamics in a vector form. To do so, we

define a graph matrix G_i for each agent in the network. The graph matrix G_i is an m_i x n matrix[†], where m_i is the number of observations of agent i, i.e. the number of agents in the set $\mathcal{N}(i)$. In each row of G_i, the entry in column i is 1, a different entry $j \in \mathcal{N}(i)$ is -1, and the remaining entries are 0. We also define a gain vector K_i for each agent i of the network. The gain vector K_i is an m_i-component row vector that contains the gains k_{ij}, $j \in \mathcal{N}(i)$ (with the order chosen to match the row-order in G_i). We define the **full graph matrix** as

$$G = \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_n \end{pmatrix},$$

and the **full gain matrix** as

$$K = \begin{pmatrix} K_1 & & \mathbf{0} \\ & K_2 & \\ & & \ddots \\ \mathbf{0} & & K_n \end{pmatrix}.$$

Later, it will be our goal to design the full gain matrix K such that the network graph matrix KG is desirable.

The dynamics of the closed loop system can be represented in matrix form as

$$\dot{x} = -KGx + C(z), \tag{3.10}$$

where $C(z)$ is a piecewise constant that is a function of z and hence of x, and where we refer to KG as the network graph matrix. We use the notation $C(z)$ since each component of C is defined

[†]Without loss of generality, the graph matrix can represent a weighted graph. We appropriately scale the gain matrix to accomodate for weighted edges in the graph.

by Equation 3.9 and so it is natural to phrase the closed loop dynamics in terms of z. We will be concerned with steady state solutions and dynamics of Equation 3.10. Specifically, we will seek to design K so that the closed-loop system achieves global asymptotic synchronization, and further so that the approach to the synchronized state is fast.

The matrix KG is very specially structured. In general, the matrix has row sums of zero. In the special case that k_{ij} is chosen to equal k_{ji}, the symmetric matrix KG is the Laplacian matrix. Most of the results that we present assume this condition on K. The properties of KG (in the general and symmetric cases) simplify eigenanalysis and hence transient analysis.

Network Graph We define a network graph for the SIN as follows. We associate a node in the graph with each agent in the network. There is an edge from agent i to agent j if row i and column j, $i \neq j$, of KG is nonzero, i.e. $j \in \mathcal{N}(i)$. This edge is a weighted edge, with the weight equal to k_{ij}. In our case, if there is an edge from node i to node j there is also an edge from node$j$ to node i; however, in the general, the gains k_{ij} and k_{ji} are not equal, so it natural to view the network graph as a weighted, directed graph. In the special case that $k_{ij} = k_{ji}$, the network graph is a weighted, undirected graph. The network graph matrix KG is a matrix representation of the network graph.

3.2 Mode Locking With A Linear Control Law

This section contains an algorithm for finding mode-lock equilibrium points when the static linear controller is used. Also, we show that the network angle always converges to an equilibrium point, and there is no periodic or chaotic behavior. Last, we give a theorem stating a relationship between network structure and the existence of mode-lock equilibrium points.

3.2.1 Finding Mode Lock Equilibrium Points

Mode lock equilibrium points are steady state solutions of the n simultaneous differential equations described by Equation 3.8. To find the mode-lock equilibrium points, each $\dot{x_i}$ is set equal to zero. The i^{th} equation in steady is

$$0 = -\sum_{j \in \mathcal{N}(i)} k_{ij}[((\pi + z_i - z_j)\ mod\ 2\pi) - \pi] \tag{3.11}$$

A system of linear equations would result in a unique solution z (to within an additive constant). However, these equations are piece-wise linear, so different regimes may yield different solutions. Also, if z is a solution, then $\bar{z} = z + \delta$, where $\delta \in [-\pi, \pi)$, is also a solution since the relative angles between each of the agents remain the same. Thus, there are an infinite number of solutions, but due to the rotational nature of the system, we are not interested in the solutions that differ only by the addition of a constant. For consistency, we constrain the first components of all our solutions z to be zero, in which case each linear peice in Equation 3.11 has at most one solution. Let us now quantify these possibilities. From Equation 3.9 we see that each $((\pi + z_i - z_j)\ mod\ 2\pi) - \pi$ can result in one of three expresions. The system of equations described by Equation 3.11 thus has 3^e (e is the number of edges in the graph) domains with different linear expressions for the right side. To ensure that all the solutions are found, Equation 3.11 needs to be solved for each of these 3^e possibilities. In the case that $k_{ij} = k_{ji}$, there are $3^{1/2e}$ possibilities.

After a possible solution z of the system of equations described by Equation 3.11 is found, it needs to be verified that z is actually a solution. This can be done by simply calculating $z_i - z_j\ \forall\ i, j \in \mathcal{N}(i)$. Recall that the expressions described by Equation 3.11 only hold for certain domains. One simply has to check whether the solution falls in the proper domain.

The above procedure is illustrated in Example 4.

Example 4

Consider a network of four agents with the network graph matrix

$$KG = \begin{pmatrix} 11 & -2 & -4 & -5 \\ -2 & 9 & 0 & -7 \\ -4 & 0 & 10 & -6 \\ -5 & -7 & -6 & 18 \end{pmatrix}.$$

In the interest of space, we consider only one of the domains over which the right side of Equation 3.11 is linear. The chosen domain is $-\pi \leq z_1 - z_2 \leq \pi$, $-\pi \leq z_1 - z_3 \leq \pi$, $-\pi \leq z_1 - z_4 \leq \pi$, $-\pi \leq z_2 - z_4 \leq \pi$, and $z_3 - z_4 > \pi$. For this domain, the system of equations decribed by Equation 3.11 is

$$\begin{aligned} 0 &= -2(z_1 - z_2) - 4(z_1 - z_3) - 5(z_1 - z_4) \\ 0 &= -2(z_2 - z_1) - 7(z_2 - z_4) \\ 0 &= -4(z_3 - z_1) - 6(z_3 - z_4 - 2\pi) \\ 0 &= -5(z_4 - z_1) - 7(z_4 - z_2) - 6(z_4 - z_3 + 2\pi) \end{aligned} .$$

Solving the system of equations results in a solution $z^T = \begin{pmatrix} 0 & -1.3096 & 2.7596 & -1.6838 \end{pmatrix}$.

To verify that this is a solution, the differences $z_i - z_j$ were calculated and are

$$\begin{aligned} z_1 - z_2 &= 1.3096 & z_2 - z_1 &= -1.3096 \\ z_1 - z_3 &= -2.7596 & z_3 - z_1 &= 2.7596 \\ z_1 - z_4 &= 1.6838 & z_4 - z_1 &= -1.6838 \\ z_2 - z_4 &= .3742 & z_4 - z_2 &= -.3742 \\ z_3 - z_4 &= 4.4434 & z_4 - z_3 &= -4.4434 \end{aligned}$$

Thus, we see that z is in the proper domain and so is a solution.

Solving the system of equations for all possible $3^{e/2}$ combinations resulted in only one other distinct mode lock equilibrium point, which is $z^T = \begin{pmatrix} 0 & -2.7284 & -2.1048 & 2.7752 \end{pmatrix}$. $\square$

3.2.2 Conditions For Mode Lock and Global Asymptotic Synchronization

Here, we will prove that, in the case of static linear control, and from any initial network angle, the network angle z(t) asymptotically approaches an equilibrium point. Also, we will a state condition (with proof) for the existence of mode lock equilibrium points.

First, let us define some of the notation we will use below. Let D be the diagonal matrix with each of the gains k_{ij} on the diagonal, with the order matching the row order of the full graph matrix G. Note that D is an e x e symmetric matrix with all non-negative entries. We let q be the e x 1 vector containing the additive constants determined by the equalities in Equation 3.9. As with D, the entries of q are ordered to match the row order of G.

Lemma 1 *For any general linear static control law such that each gain $k_{ij} \geq 0$ and $k_{ij} = k_{ji}\ \forall\ i, j \in \mathcal{N}(i)$, and for any network graph, the network angle asymptotically approaches an equilibrium point.*

Proof: First, we write the closed loop dynamics as

$$\dot{x} = -\frac{1}{2}(G^T D(Gx + q)). \tag{3.12}$$

First, we will show that this is equivalent to Equation 3.10. Note that in general $G^T D \neq K$, but $G^T DG = 2KG$. This is due to the special structure of G, along with the fact that $k_{ij} = k_{ji},\ \forall\ i, j \in \mathcal{N}(i)$. To show that $G^T DG = 2KG$, we first start by defining G in a more precise manner. If the

edges of the graph are numbered $1, 2, \ldots, e$, then G is as follows:

$$\begin{aligned} G_{ji} &= 1, && \text{node } i \text{ is the tail and some node } m \text{ is the head of edge } j \\ G_{ji} &= -1, && \text{some node } m \text{ is the tail and node } i \text{ is the head of edge } j \\ G_{ji} &= 0, && \text{Otherwise} \end{aligned} \quad \cdot$$

Now, we look at the product DG. This product is:

$$\begin{aligned} (DG)_{ji} &= k_{im}, && \text{node } i \text{ is the tail and node } m \text{ is the head of edge } j \\ (DG)_{ji} &= -k_{mi}, && \text{node } m \text{ is the tail and node } i \text{ is the head of edge } j \\ (DG)_{ji} &= 0, && \text{Otherwise} \end{aligned} \quad \cdot$$

The matrix G^T follows from G, and is:

$$\begin{aligned} (G^T)_{sj} &= 1, && \text{node } s \text{ is the tail and some node } m \text{ is the head of edge } j \\ (G^T)_{sj} &= -1, && \text{some node } m \text{ is the tail and node } s \text{ is the head of edge } j \\ (G^T)_{sj} &= 0, && \text{Otherwise} \end{aligned} \quad \cdot$$

Now, multiplying G^T by DG yields a square n x n matrix, defined as follows:

$$(G^T DG)_{si} = \sum_{j=1}^{e} (G^T)_{sj} (DG)_{ji}.$$

Recall $(G^T)_{sj} = 1$ for each edge that has node s as the tail and some node m as the head, and $(G^T)_{sj} = -1$ for each edge that has node s as the head and some node m as the tail. Also, $(DG)_{ij} = k_{im}$ for each edge that has node i as the tail and node m as the head, and $(DG)_{ij} = -k_{mi}$ for each edge that has node i as the head and node m as the tail. For diagonal entries, (s = i), the summation over all of the edges result in four cases that are nonzero:

Node s is the tail of edge j, which in this case $(G^T)_{sj} = 1$

1) node i is the tail of edge j, which in this case $(DG)_{ji} = k_{im}$

2) node i is the head of edge j, which in this case $(DG)_{ji} = -k_{mi}$

Node s is the head of edge j, which in this case $(G^T)_{sj} = -1$

3) node i is the tail of edge j, which in this case $(DG)_{ji} = k_{im}$

4) node i is the head of edge j, which in this case $(DG)_{ji} = -k_{mi}$.

First, note that case 2 and case 3 are not possible, since the same node cannot be the head and tail of an edge. Also, $(G^T)_{sj} = \pm 1$, and $(DG)_{ji} = \pm k_{im}$ for each node m that is a neighbor of node i (or node s). Thus the summation over all edges is equivalent to a summation over neighboring nodes. From above, case 1 results in a summation of the form $\sum_{m \in N(i)}(1)(k_{im})$ and case 4 results in a summation of the form $\sum_{m \in \mathcal{N}(i)}(-1)(-k_{mi})$. Therefore, the diagonal entries are of the form $\sum_{m \in \mathcal{N}(s)} k_{sm} - \sum_{m \in \mathcal{N}(s)} -k_{ms}$. Since $k_{ms} = k_{sm}$, this is $2\sum_{m \in \mathcal{N}(s)} k_{sm}$.

For the off-diagonal entries ($i \neq s$) of $G^T DG_{si}$, the summation over all of the edges result in the same four cases as above. However, for these entries, case 1 and case 4 are not possible since an edge must have a tail and a head. If node s is the tail of edge j, $(G^T)_{sj} = 1$, and node i must be the head of edge j (if node i is a neighbor of node s), so $(DG)_{ji} = -k_{si}$. Similarly, if node s is the head of edge j, $(G^T)_{sj} = -1$, and node i must be the tail of edge j (if node i is a neighbor of node s), so $(DG)_{ji} = k_{is}$. Thus, each off-diagonal entry of $(G^T DG)$ is of the form $(1)(-k_{si}) + (-1)(k_{is})$, or $-2k_{si}$. Notice that if node i and node s are not neighbors, $G^T DG_{si} = 0$. Since $k_{ij} = k_{ji}$, the matrix KG is a Laplacian matrix. Therefore, $1/2(G^T DG) = KG$ (see Chapter 2 for a description of the Laplacian).

Next, we need to equate $G^T Dq$ and $C(z)$. In the equalities in Equation 3.9, we let c_{si} be the additive constant. Note that $c_{si} = -c_{is}$. Thus, from Equation 3.8, each row of C(z) is of the form $\sum_{i \in \mathcal{N}(s)} k_{si} c_{si}$. The vector q is a vector containing all of these c_{si} and is ordered in the following

way:

$$q_j = c_{si}, \quad \text{node } s \text{ is the tail and node } i \text{ is the head of edge } j. \quad \cdot$$

The matrix $G^T D$ follows from DG, and is:

$$\begin{aligned}
(G^T D)_{rj} &= k_{rm}, && \text{node } r \text{ is the tail and node } m \text{ is the head of edge } j \\
(G^T D)_{rj} &= -k_{mr}, && \text{node } m \text{ is the tail and node } r \text{ is the head of edge } j \quad \cdot \\
(G^T D)_{rj} &= 0, && \text{Otherwise}
\end{aligned}$$

The product $G^T Dq$ is:

$$(G^T Dq)_r = \sum_{j=1}^{e} (G^T D)_{rj} q_j.$$

Note that $G^T D$ is only nonzero for edges that are incident to node r. Thus, the summation over all edges is equivalent to a summation over neighboring nodes. Here, the summation over the neighboring nodes results in only two cases that are nonzero:

1) node r is the tail of and node m is the head of edge j, which in this case $(G^T D)_{rj} = k_{rm}$, and $q_j = c_{rm}$

2) node r is the head and node m is the tail of edge j, which in this case $(G^T D)_{rj} = -k_{mr}$, and $q_j = c_{mr}$

Thus, each row of the product is of the form $\sum_{m \in \mathcal{N}(r)} (k_{rm})(c_{rm}) + \sum_{m \in \mathcal{N}(r)} (-k_{mr})(c_{mr})$. Since $k_{mr} = k_{rm}$, and $c_{mr} = -c_{rm}$, this is $2 \sum_{m \in \mathcal{N}(r)} k_{rm} c_{rm}$. Hence, $1/2(G^T Dq) = C(z)$.

Now that we are sure that the model of the dynamics in Equation 3.12 is equivalent to Equation 3.10, we use a Lyapunov arguement to show convergence of the network angle to an equilibrium point. We consider the quadratic Lyapunov function

$$V(x) = (x^T G^T + q^T) D (Gx + q). \tag{3.13}$$

The Lyapunov function is a weighted sum of the squares of $d(x_i, x_j)$, so is positive semi-definite. The derivative of the Lyapunov function is

$$\dot{V} = \dot{x}^T G^T D(Gx + q) + (x^T G^T + q^T) DG\dot{x}. \tag{3.14}$$

Since D is symmetric, Equation 3.14 is equivalent to

$$\dot{V} = 2(x^T G^T + q^T) DG\dot{x}. \tag{3.15}$$

Now, substitue Equation 3.12 into Equation 3.15 to get

$$\dot{V} = 2(x^T G^T + q^T) DG(-\frac{1}{2}(G^T D(Gx + q))). \tag{3.16}$$

After some algebra, we find that

$$\dot{V} = -(Gx + q)^T (G^T D)^T (G^T D)(Gx + q), \tag{3.17}$$

or equivalently

$$\dot{V} = -4||\dot{x}||^2. \tag{3.18}$$

Therefore, $\dot{V}$ is negative semi-definite, and if $\dot{V} = 0$, this implies that $\dot{x} = 0$. ■

Here we note that the arguement does not take into account the switching of q. This is due to the fact that x is continuous, and there are only jumps in $\dot{x}$. Additionally, $\dot{V}$ is negative on both sides of switch in q. Also, the switching is not dense, since x has to go 2π before switching is invoked. Thus, the arguement is valid.

Also, we note that if the network angle converges to a sycnrhonized state, $V(x)$ approaches zero, and $V(0) = 0$. If the network angle does not converge, and instead approaches a mode-lock equilibrium point, $V(x)$ approaches some γ, $\gamma > 0$. In both cases, when an equilibrium point is reached, $\dot{V} = 0$. The above Lemma shows that periodic or chaotic behavior is not possilbe. This guarantees asymptotic convergence of the network angle to an equilibrium point.

The above Lemma shows that periodic or chaotic behavior is not possilbe. This guarantees asymptotic convergence of the network angle to an equilibrium point. The Lyapunov function is illustrated below in Example 5 and Example 6.

Example 5

Consider a network of four agents with the network graph matrix

$$KG = \begin{pmatrix} 11 & -2 & -4 & -5 \\ -2 & 9 & 0 & -7 \\ -4 & 0 & 10 & -6 \\ -5 & -7 & -6 & 18 \end{pmatrix}.$$

Below, Figure 3.4a shows the angles of each agent evolving in time, starting from the initial network angle

$$z^T = \begin{pmatrix} -1 & 0.9 & -2.2 & 2 \end{pmatrix},$$

while Figure 3.4b shows the Lyapunov function. Figure 3.4c and Figure 3.4d show the results for the initial network angle

$$z^T = \begin{pmatrix} -1.5 & .75 & -2.2 & 2.5 \end{pmatrix}.$$

In Figure 3.4c and Figure 3.4d, the switching in the network dynamics is apparent. Even though the dynamics are switching, the Lyapunov function is still decreasing, and the synchronized state is asymptotically approached. □

Example 6

This example is a network of 20 agents. Figure 3.5a and Figure 3.5c show the network angle evolving in time for two different initial network angles while Figure 3.5b and Figure 3.5d show the Lyapunov function.

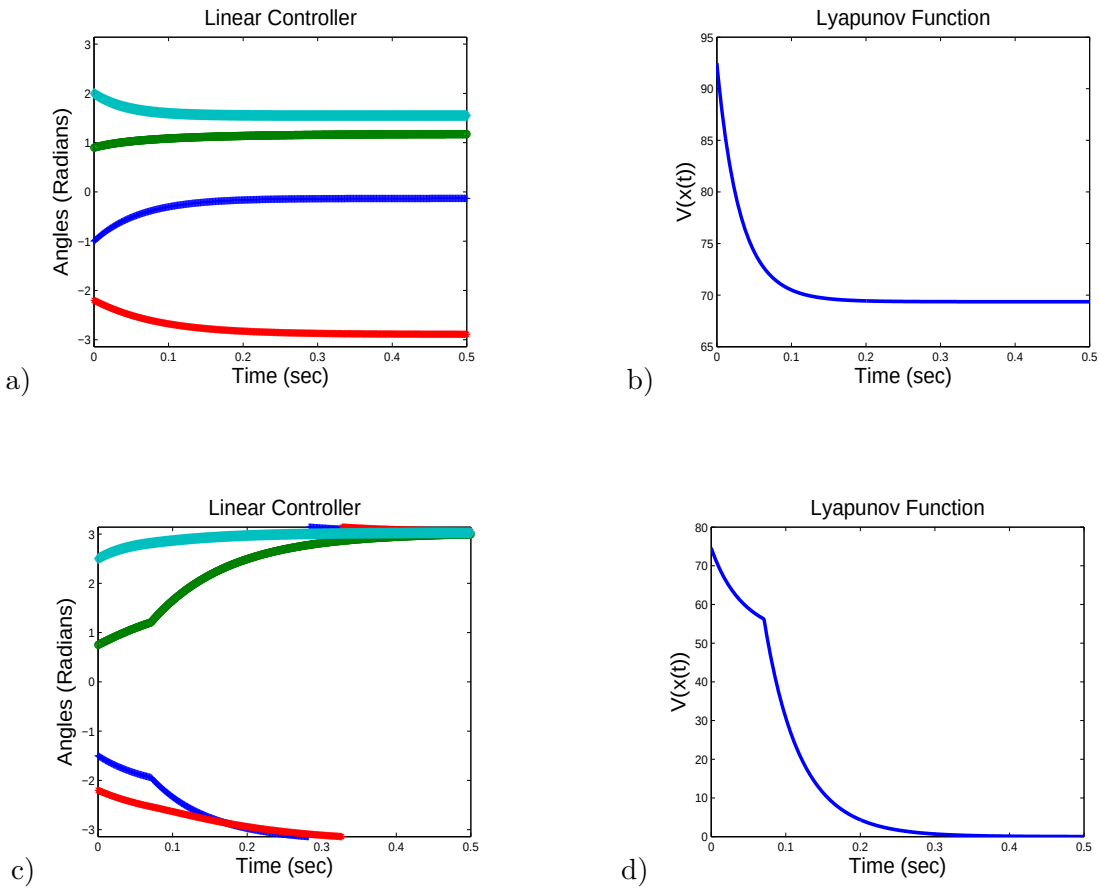

Fig. 3.4: Network angle and Lyapunov function for network of four agents.

Now that we know the network angle approaches an equilibrium point, we need to relate network structure to the existence of mode-lock equilibrium points. This will help us design controllers to ahcieve our goal of global asymptotic synchronization. A theorem relating network structure to the existence of mode-lock equilibrium points is given below in Theorem 1.

Theorem 1 *For any general linear static control law, any network graph that is a tree graph does not have any mode lock equilibrium points. Furthermore, the SIN achieves global asymptotic synchronization in this case.*

Proof: A tree graph has at least two nodes that have degree one. Consider any node in the graph

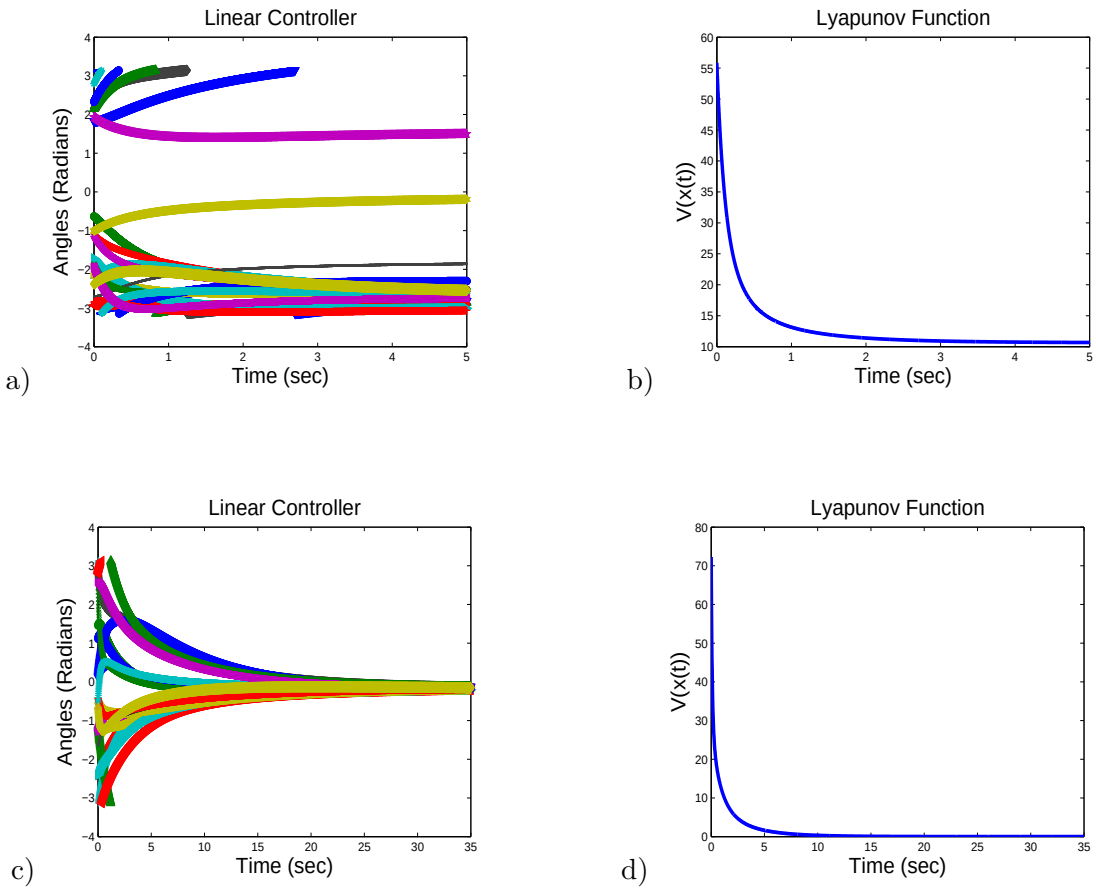

Fig. 3.5: Network angle and Lyapunov function for network of 20 agents.

that has degree one, denote this node by A. Node A is only connected to one other node, say node B. Thus, the dynamics for agent A are

$$\dot{x}_A = k_{AB}(((\pi + z_B - z_A) \ mod \ 2\pi) - \pi).$$

From Equation 3.9, it is easy to verify that in steady state, $z_A = z_B$ (x_A is equal to x_B or is a 2π multiple of x_B), therefore the angle of agent A will be the same as the angle of agent B. Thus, we can remove agent A and the edge between agent A and agent B from the graph. The subgraph that results will in turn have at least two nodes that have degree one. We recursively apply this process to the leaves (nodes that are only connected to one other node) of the graph, finally arriving at a

graph with a single agent. Thus, a tree graph will not mode lock.

Global asymptotic synchronization follows from Lemma 1. ■

Conjecture: Based on numerous simulations, we believe that in the case where KG represents a tree graph, the restrictions on K can be lessened. Specifically, the requirement $k_{ij} = k_{ji}$ can be removed. Also, the condition $k_{ij} \geq 0$ is sufficient, but may not be necessary. This is a powerful result in that any LTI controller, with each $k_{ij} \geq 0$, will achieve the goal of global asymptotic synchronization. In addition, some of the k_{ij} may be negative, but this is not fully understood. An example where $k_{ij} \neq k_{ji}$ is shown below in Example 7.

Example 7

Consider a network of seven agents with the following network graph matrix:

$$KG = \begin{pmatrix} 4 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 13 & -13 & 0 & 0 & 0 & 0 \\ -15 & -8 & -33 & -1 & 0 & -9 & 0 \\ 0 & 0 & -16 & 32 & -16 & 0 & 0 \\ 0 & 0 & 0 & -19 & 19 & 0 & 0 \\ 0 & 0 & -17 & 0 & 0 & 25 & -8 \\ 0 & 0 & 0 & 0 & 0 & -13 & 13 \end{pmatrix}.$$

Note that this represents a tree graph. Figure 3.6 shows the network angle evolving in time, starting from the initial network angle

$$z^T = \begin{pmatrix} -1.6884 & 1.2183 & -0.5981 & -2.7068 & -2.3135 & 1.4729 & -2.8351 \end{pmatrix}.$$

The fact that tree graphs do not mode lock is the basis of our methodology for designing controllers for SINs.

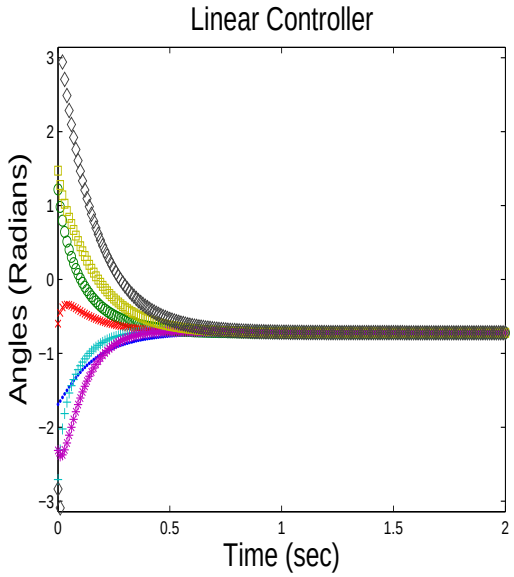

Fig. 3.6: Tree graph with the symmetric condition on K removed still asymptotically approaches a synchronized state.

Chapter 4

Control Design

Now that we know how a SIN's topological structure affects its asymptotics, we can design controllers to achieve our goal of global asymptotic synchronization. Specifically, we pursue designs for which the network graph is a tree, since we know that these designs will not mode lock. Also, we seek to design tree network graphs that have fast settling to the synchronized state. This convergence rate is dependent on the eigenvalues of the network graph matrix. As time evolves, and given that the network state is close to the synchronized state (so that switching is not invoked), each state trajectory $x(t)$ behaves according to $\sum_i c_i e^{-\lambda_i t}$, where each λ_i is an eigenvalue of KG. The rate of convergence is most strongly dependent on the eigenvalue λ_1. More precisely, the eigenvalue λ_0 is identically zero, and its associated dynamics do not effect the rate of convergence, but only the steady state value to which the SIN converges. We are not concerned with this value, only the fact that the SIN does converge. Instead, the algebraic connectivity dominates the settling dynamics of the SIN, so we want to find tree graphs that have the largest algebraic connectivity.

The algorithms presented below are deeply connected to the field of algebraic graph theory (see [5]), as well as the idea of eigenvalue sensitivity. Eigenvalue sensitivity is a measure of how sensitive the eigenvalues of a matrix are to small perturbations of the entries of the matrix. Eigenvalue sensitivity notions are widely used in several application areas, including power networks ([6]),

discrete control systems ([28]), and biological networks ([2]).

Our controller design strategies make use of the ideas of eigenvalue sensitivity and algebraic graph theory. Given any network graph, we wish to change the gains k_{ij} in such a manner that KG represents a tree graph, while keeping λ_1 as large as possible. Again, we require that K is chosen such that KG is symmetric. We have developed three different heuristic algorithms, which are explained below.

4.1 Fixed Observation Topology

Here, we consider a network where the obervations that each agent can make are predefined, and we can only adjust the gains. Therefore, we want to choose the gains in a manner such that the network graph matrix KG represents a spanning tree of the original network graph. Also, we wish to choose a spanning tree that yields a"good" algebraic connectivity, and hence a "good" settling rate. Here,"good" means that the chosen spanning tree may not yield the largest algebraic connectivity, but the rate of convergence is satisfactory. We have developed two different algorithms to find spanning trees, as described below.

4.1.1 Eigenvalue Sensitivity Approach

Given a network graph, we wish to remove edges from the graph (i.e. set controller gains to zero) until we are left with a connected tree graph. In addition, this resulting tree graph needs to have a satisfactory algebraic connectivity. Below, we first review the classical eigenvalue-sensitivity equation, and then describe our eigenvalue-sensitivity algorithm.

Review: consider a matrix A and a distinct eigenvalue λ associated with A. Also, let v be the right eigenvector associated with λ, and let w^T be the left eigenvector associated with λ. We wish to find an expression for how λ changes with the entries of A.

We let the matrix ΔA describe the changes to the entries in the matrix A. For small enough ΔA, $A + \Delta$A has an eigenvalue near λ, say $\lambda + \Delta\lambda$, which is distinct, and a corresponding right eigenvector that only varies slightly from v, say $v + \Delta$v.

In order to find how $\Delta\lambda$ changes with ΔA, we start with the expression $Av = \lambda v$. To find an eigenvalue of $A + \Delta$A, we use $(A + \Delta\text{A})(v + \Delta\text{v}) = (\lambda + \Delta\lambda)(v + \Delta\text{v})$. After expanding the expression, we are left with

$$Av + A\,\Delta\text{v} + \Delta\text{A}\,v + \Delta\text{A}\,\Delta\text{v} = \lambda v + \lambda\,\Delta\text{v} + \Delta\lambda\,v + \Delta\lambda\,\Delta\text{v}\,.$$

Now, we multiply both sides of the expression from the left by w^T and use the fact that $Av = \lambda v$ to get

$$w^T A\,\Delta\text{v} + w^T\,\Delta\text{A}\,v + w^T\,\Delta\text{A}\,\Delta\text{v} = w^T\lambda\,\Delta\text{v} + w^T\,\Delta\lambda\,v + w^T\,\Delta\lambda\,\Delta\text{v}\,.$$

Next, we make use of the expression for finding a left eigenvector, which is $w^T A = w^T\lambda$. Thus, the first terms on each side of the above equality are equivalent so we are left with

$$w^T\,\Delta\text{A}\,v + w^T\,\Delta\text{A}\,\Delta\text{v} = w^T\,\Delta\lambda\,v + w^T\,\Delta\lambda\,\Delta\text{v}\,.$$

Notice that $\Delta\lambda$ is a scalar, so the first term on the right side of the above equality is equivalent to $\Delta\lambda\,w^T v$. We can normalize w^T and v such that $w^T v = 1$. After doing this, we are left with

$$w^T\,\Delta\text{A}\,v + w^T\,\Delta\text{A}\,\Delta\text{v} = \Delta\lambda + w^T\,\Delta\lambda\,\Delta\text{v}\,.$$

If we ignore the higher order terms, we find that $\Delta\lambda = w^T\,\Delta\text{A}\,v$ is a first order approximation.

We are interested in the sensitivity of the algebraic connectivity of KG to the removal of an edge of KG. In the above formula, the matrix A is the network graph matrix KG in our case, and

ΔA represents the removal of an edge of the network graph. If the edge $\{i,\ j\}$ is removed, then $\Delta A_{ii} = \Delta A_{jj} = -k_{ij}$ and $\Delta A_{ij} = \Delta A_{ji} = k_{ij}$. The rest of the entries of ΔA are zero. Since KG is symmetric, the left eigenvector is the transpose of the right eigenvector. Therefore, with a little algebra we find that $\Delta\lambda = k_{ij}(v_i - v_j)^2$, where v_i and v_j are the i^{th} and the j^{th} components of the right eigenvector, respectively.

From the above sensitivity expression, it is clear that removing the edge with the minimum weighted difference in eigenvector components results in a small change in the algebraic connectivity of the graph. This result gave us insight into finding spanning trees. We have developed a heuristic algorithm from this idea, which is listed below.

1) calculate the eigenvector $\mathbf{v}$ associated with the second smallest eigenvalue of KG

2) calculate $k_{ij}(v_i - v_j)\ \forall\ i, j \in \mathcal{N}(i)$

3) remove the edge $\{i,\ j\}$ that has the minimum weighted difference of eigenvector components, i.e set k_{ij} (and k_{ji}) to zero

4) find the new network graph matrix

5) repeat until the resulting network graph is a tree graph

It should be noted that it is possible that the algorithm finds a graph that is not connected. To avoid this problem, we change the algorithm as follows: if edge $\{i,\ j\}$ has the minimum difference $k_{ij}(v_i - v_j)$ but the removal of this edge would result in a graph that is not connected, then the edge with the next smallest difference is removed instead. This is of great importance due to the fact that it is not possible for a network that is not connected to reach a synchronized state. Also, the algorithm is not guaranteed to find the optimal solution, but the results are generally satisfactory.

To illustrate the algorithm, we consider the following example.

Example 8

Consider the network graph matrix

$$KG = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 \\ -1 & -1 & 3 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 2 \end{pmatrix}.$$

The first iteration of the algorithm will be shown to illustrate the algorithm in detail.

The eigenvector $\mathbf{v}$ associated with second smallest eigenvalue λ_1 is

$$\mathbf{v}^T = \begin{pmatrix} -0.1380 & 0.2560 & 0.2560 & -0.8115 & 0.4375 \end{pmatrix}.$$

The edge that has the minimum difference of eigenvector components is the edge $\{2, 3\}$.

After this edge is removed, the resulting graph matrix is

$$KG = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 2 \end{pmatrix}.$$

The steps above are repeated until the resulting graph is a tree graph.

The final tree graph is

$$KG_{tree} = \begin{pmatrix} 3 & -1 & -1 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 2 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix}.$$

The algebraic connectivity of the dense graph is 0.8299 while the algebraic connectivity of the tree graph is 0.5188. □

4.1.2 Bushiness Approach

This algorithm is based on a number of observations regarding algebraic connectivities of trees and other graphs. The articles [1] and [8] give many results on how algebraic connectivity is related to the network topology. Also, many lower bounds on the algebraic connectivity are given in terms of graph properties, such as the number of nodes, the number of edges, the diameter of the graph, and several others. We use these bounds as guidelines for how we want to find spanning trees.

The main observation we made is that graphs with small diameters typically have larger algebraic connectivities. Also, graphs that are more strongly and centrally connected yield a larger algebraic connectivity. While this algorithm is more of an ad hoc type of approach than the eigenvalue-sensitivity approach, in many cases it confirms the results of the eigenvalue-sensitivity approach and in some cases does slightly better. This algorithm begins with an empty graph (graph of nodes with no edges) and add edges (by setting k_{ij} to one) as follows:

1) find an agent that is centrally located and has many neighbors

2) connect this agent to all of its neighbors

3) find the node with the next-highest degree (number of neighbors) that is already connected to some other agent

4) connect this agent to all of it's neighbors that are not already connected

5) repeat until all of the nodes are connected

If possible, branches of the tree that are connected closer to the center of the graph are preferable. Also, it is possible for a node to be connected to more than one other node. These are usually nodes that are the last to be connected. In this case, the change in the algebraic connectivity of the tree graph is so small that the node can be connected to any of the possiblities. If the maximum algebraic connectivity is desired, a trial and error approach is the best way to determine connections for these last nodes. The algorithm is illustrated below in Example 9.

Example 9

Consider the network of 12 agents shown below in Figure 4.1a. The algorithm starts with an empty graph. A centrally connected agent with the largest degree is found, and all of its neighbors are connected, see Figure 4.1b. Of the agents that are now connected, agent 7 has the highest degree so it is connected to all of its neighbors that are not already connected (Figure 4.1c). All that remains to be connected are each of the corner agents. Agent 1 can be connected to either agent 2 or agent 5; angets 5, 16, and 20 also have more than one other node that they can be connected to. A trial and error approach was used to find the best connections for these agents, and the connected tree graph shown in Figure 4.1d. □

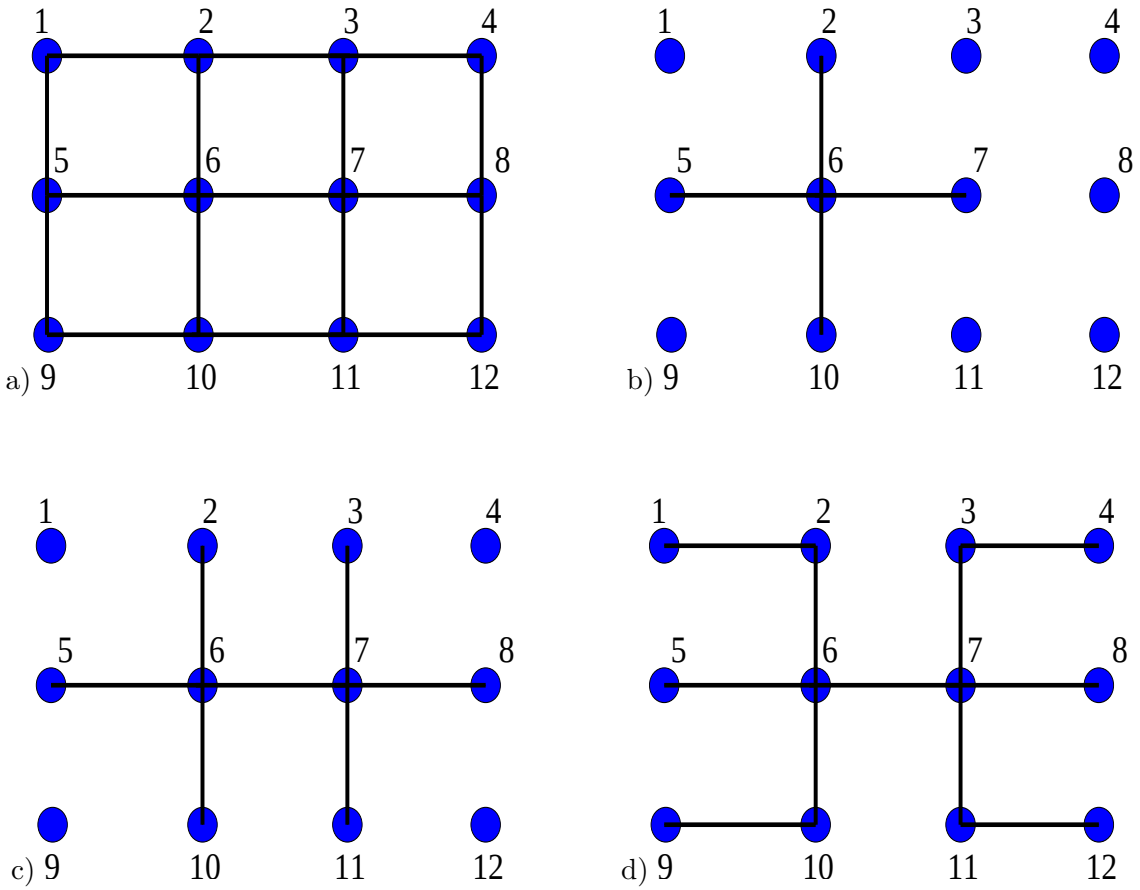

Fig. 4.1: Applying the bushiness algorithm.

4.2 Freedom To Design The Network

In this case, we are given the freedom to design the network, i.e we are able to choose the observations an agent can make, and we can adjust the gains. By setting the gains arbitrarily large, the algebraic connectivity of the network graph can be made arbitrarily large. However, this can lead to a number of problems, most notably actuator saturation. This is a problem that we wish to avoid altogether, so we fix all of the gains at one. Therefore, we want to choose the observations that each agent can make in a way that eliminates the possibility of mode lock and leads to satisfactory settling rates. The following theorem provides insight into how we want choose

the observations for each agent.

Theorem 2 *Among tree graphs with n nodes, the star graph has the largest algebraic connectivity.*

Proof: If there are n nodes in the network, a star graph has the following graph structure:

$$KG = \begin{pmatrix} n-1 & -1 & -1 & \cdots & -1 & -1 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ & \vdots & & \vdots & & \\ -1 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Since the gains are all fixed at one, the resulting network graph matrix is symmetric. First, we will prove that the algebraic connectivity of the star graph is one. Then, we will show that the star graph has the largest algebraic connectivity among tree graphs.

In order to find the second smallest eigenvalue λ_1, consider the associated eigenvector v such that KGv = λ_1v. Let v = $\begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix}$. Therefore, KGv = $\begin{pmatrix} (n-1)v_0 - \sum_{i=1}^{n-1} v_i \\ v_1 - v_0 \\ \vdots \\ v_{n-1} - v_0 \end{pmatrix}$. Let $v_0 = 0$. Then, KGv = $\begin{pmatrix} -(v_1 + v_2 + \ldots + v_{n-1}) \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix}$. Now, set $v_i = \gamma$ and v_j = -γ, where γ is any number, $i \neq j$, $1 \leq i \leq$ n-1, $1 \leq j \leq$ n-1, and $v_k = 0$, $1 \leq k \leq$ n-1, $k \neq i, j$. If v is selected in this manner, $KGv = v$. Therefore, $\lambda_1 = 1$. In addition, there are n-1 entries in v that can take the value of $\pm\gamma$, but only two at a time. Thus, there are n-2 linearly independent vectors v such that $KGv = v$.

Hence, $\lambda_1 = 1$, λ_1 is repeated n-2 times, and λ_1 is simple (has n-2 linearly independent eigenvectors associated with it).

We now know n-1 eigenvalues of the star graph. All that is left to do is to show that the last eigenvalue is greater than one. Assume that the remaining eigenvalue is the largest eigenvalue. By the Courant Fischer Theorem, we have

$$\lambda = max\ \frac{v^T KGv}{v^T v}.$$

Or, equivalently

$$\lambda = max\ \frac{\sum_{\{i,j\}\in E(KG)}(v_i - v_j)^2}{\sum_{i=0}^{n-1} v_i^2}.$$

This becomes

$$\lambda = max\ \frac{\sum_{i=1}^{n-1}(v_0 - v_i)^2}{\sum_{i=0}^{n-1} v_i^2}.$$

Let v be the vector

$$v = \begin{pmatrix} \frac{1}{\sqrt{n}} \\ -\frac{1}{\sqrt{n}} \\ \frac{1}{\sqrt{n}} \\ -\frac{1}{\sqrt{n}} \\ \vdots \end{pmatrix}$$

If v is selected in this way, the numerator in the above expression becomes 2 if n is even, and 2 - 2/n if n is odd. The vector v is normalized, so the denominator is 1. Thus, $\lambda = 2$ if n is even, and $\lambda = 2$ - 2/n if n is odd. Therefore, $\lambda > 1\ \forall\ n > 2$. We know that the vector that maximizes the above expression is the eigenvector. Obviously, the vector v is not an eigenvector, since $KGv \neq \lambda$v. Hence $\lambda > 2 - 2/n > 1$. The algebraic connectivity of a star graph is 1.

Now we want to show that the star graph is the tree graph with the largest algebraic connectivity. In order to do this, we first start with a star graph. Then, we remove one of the nodes that is

connected to the center node, and also the edge that this node is incident to. Append this node to one of the leaves in the graph. Without loss of generality, call the center node node 0, the node that was moved node 1, and the node that the removed node is appended to node 2. Then, the graph has the following structure:

$$KG = \begin{pmatrix} n-2 & 0 & -1 & -1 & -1 & \cdots & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ & \vdots & & & & \vdots & & \\ -1 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Let v be the vector such that $v_i = \gamma$ and v_j = -γ, where γ is any number, $i \neq j$, 3 $\leq i \leq$ n-1, 3 $\leq j \leq$ n-1, and $v_k = 0$, $1 \leq k \leq$ n-1, $k \neq i, j$. We know from above that $\frac{v^T KGv}{v^t v} = 1$. However, the vector v is not an eigenvector in this case. We know that the minimizing vector is an eigenvector, so there is another vector that results in a value of $\lambda_1 < 1$.

Therefore, this graph has an algebraic connectivity less than that of the star graph. ■

If we can design the network observation topology, we want to design a star structure. If the number of agents in the network becomes large, it may not be feasible to implement a star structure. Some applications may have a maximum degree constraint, so an agent has a limited number of observations in can make. In a case like this, a series of star graphs is the best. A series of star graphs is created by starting with a center node and connecting as many leaves to it as desired/possible. Then, as many nodes as desired/possible are connected to each of the leaves, keeping in mind that each of the leaves already have one node connected to them. This process of working outward is then repeated until all of the nodes are included in the graph.

Remark: While the control strategies above achieve our goal of global asymptotic syncrhonization, they are not very robust. This is due to working exclusively with tree graphs. In a tree graph, if a single link fails, the network is no longer connected.

To robustify the design, we propose a switching controller. The switching controller incorportates the above design strategies along with the fact that the SIN can reach a synchronized state, despite the existence of mode-lock equilibria. If the network state is "close" to a synchronized state, it will approach the synchronized state. If the network state is "close" to a synchronized state, the resulting measurements each agent makes are sufficiently small. Thus, if we use a tree topology until all of the measurements of each agent are sufficiently small, we can then switch back to the dense network graph and still not mode lock. Not only will this make the network more robust, it will increase the settling rate.

A major issue with this switching controller is knowing when to switch from the tree graph to the original network graph. For any general linear static control law, once all of the $z_i's$ are contained in a half circle, a syncrhonized state will be reached despite the existence of stable mode-lock equilibrium points. This makes sense since there is no switching in the dynamics once all of the relative angles are small enough. In this domain, the dynamics are simply $\dot{x} = -KGx$, which has no mode-lock equilibrium points.

Chapter 5

Conclusion

In this thesis, we have designed control laws to achieve synchronization for a network of rotating systems. We carefully examined [20], and used that work to motivate our research. Our methods are easy to implement, and work for any general network structure.

Bibliography

[1] N. M. M de Abreu, "Old and new results on algebraic connectivity of graphs", *Linear Algebra and its Applications*, Vol. 423, (2007), pp. 53-73.

[2] D. H. Anderson, "Spectral sensitivity in linear biological models", *Journal of Mathematical Biology*, Vol. 20, No. 2, (1984), pp. 203-221.

[3] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, 1974.

[4] C. C. Chow, "Phase-locking in weakly heterogeneous neuronal networks", *Physica D*, Vol. 118, (1998), pp.343-370.

[5] F. R. K. Chung, *Spectral Graph Theory*, American Mathematical Society: Boston, 1997.

[6] C. L. DeMarco and J. Wassner, "A generalized perturbation approach to coherency", *Proceedings of the IEEE Conference on Control Applications*, September 1995, pp. 611-617.

[7] S. Fairbanks and S. Moore, "Self-timed circuitry for global clocking", Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems, 2005, pp. 86-96.

[8] S. Fallat and S. Kirkland, "Extremizing algebraic connectivity subject to graph theoretic constraints", *The Electronic Journal of Linear Algebra*, Vol. 3, April 1998, pp. 48-74.

[9] M. Fiedler, "Algebraic connectivity of graphs", *Czechoslovak Mathematical Journal*, Vol. 23, (1973), pp. 298-305.

[10] M. Fiedler, "Eigenvectors of acyclic matrices", *Czechoslovak Mathematical Journal*, Vol. 25, (1975), pp. 607-618.

[11] A. Giridhar and P. R. Kumar, "Distributed clock synchronization over wireless networks: algorithms and analysis", *Proceedings of the IEEE Conference on Decision and Control*, 2006.

[12] C. Godsil, G. Royle, *Algebraic Graph Theory*, GTM, Springer, 2001.

[13] G. Goldsztein and H. Strogatz, "Stability of synchronization in networks of digital phase-locked loops", *International Journal of Bifurcation and Chaos*, Vol. 5, No. 4, (1995), pp. 983-990.

[14] R. Grone and R. Merris, "Algebraic connectivity of trees", *Czechoslovak Mathematical Journal*, Vol. 37, (1987), pp. 660-670.

[15] V. Gutnik and A. Chandrakasan, "Active GHz network using distributed PLLs", *IEEE Journal of Solid-State Circuits*, Vol. 35, No. 11, November 2000.

[16] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules", *IEEE Transactions On Automatic Control*, Vol. 48, No. 6, June 2003.

[17] H. K. Khalil, *Nonlinear Systems*, Prentice Hall, 2002.

[18] D. Lucarelli and I-J. Wang, "Decentralized syncrhonization protocols with nearest neighbor communication", *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, 2004, pp. 62-68.

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

[20] Gill A. Pratt and J. Nguyen, "Distributed synchronous clocking", *IEEE Transactions On Parallel And Distributed Systems*, Vol. 6, No. 3, March 1995, pp. 314-328.

[21] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control", *IEEE Control Systems Magazine*, April 2007, pp. 71-82.

[22] S. Roy, A. Saberi, and K. Herlugson,"A control-theoretic perspective on the design of distributed agreement protocols", *International Journal on Robust and Nonlinear Systems*, Special Issue on Communicating-Agent Networks, published online in November 2006.

[23] S. Roy, A. Saberi, and K. Herlugson, "Formation and alignment of distributed sensing agents with double-integrator dynamics and actuator saturation", *IEEE Press monograph Sensor Network Operations*, Jan. 2006

[24] S. Roy, K. Herlugson, and A. Saberi, "A control-theoretic perspective on distributed discrete-valued decision-making", *IEEE Transactions on Mobile Computing*, Vol. 5, No. 8, Aug. 2006, pp. 945-957.

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

[26] S. Strogatz, *Sync: The Emerging Science of Spontaneous Order*, (Hyperion, New York, 2003).

[27] H. Tanaka, M. D-S. Vieira, A. J. Lichtenberg, M. A. Lieberman, and S. Oishi, "Stability of synchronized states in one dimensional networks of second order PLLS", *International Journal of Bifurcation and Chaos*, Vol. 7, No. 3, (1997), pp. 681-690.

[28] J. Wu, S. Chen, J. F. Whidborne, and J. Chu, "Optimal floating-point realizations of finite-precision digital controllers", *Proceedings of the IEEE Conference on Decision and Control*, December 2002, pp. 2570-2575.