Distributed Gradient Systems and Dynamic Coordination

Thesis by

Demetri P. Spanos

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2006
(Defended December 5th, 2005)

I dedicate my thesis to my family:
to my father, for showing me the beauty and power of mathematics;
to my mother, for showing me that life is good, forgiving, and full of everyday happiness;
finally, to my sister, for never being fooled, much less impressed, by my "serious" facade;
I love you all.

- Demetri

Contents

Acknowledgements

Graduate school and the production of a thesis are complicated things; it is a daunting task to compress everyone and everything that has contributed to one's experience into a brief snippet of text.

Let me begin by thanking all the people with whom I have collaborated over the years, especially Morr, Vijay, and Tim. I would particularly like to thank Reza Olfati-Saber, whose course on coordination set the stage for a large portion of my graduate research (and inspired much of the work that constitutes this thesis). It is hard to guess where I would be now had I not chosen, essentially on a whim, to take his course when I first arrived at Caltech. It has truly been a privilege to have studied at this place and among these people.

No written praise seems sufficient to thank my advisor, Professor Richard Murray; he has been everything one could ask from an advisor, and many things I would never have thought to ask for. His interminable supply of energy and curiosity has been a constant source of motivation, and he has frequently (and repeatedly) re-ignited my interest in my own research through his example. I can only hope one day to be for someone what he has been for me.

I would also like to thank all the other members of my thesis committee, Steven Low, Jerry Marsden, and Joel Burdick, not only for their time, effort, and attention, but also for their patience with my "last minute" editing shenanigans; I hope at least not to have bored them with the result.

I wish to thank all of my teachers in life, but in particular wish to single out five professors from whom I learned "the fundamentals": Steve Cox, Bill Symes, Richard Tapia, Danny Sorensen, and Stephen Semmes. No day goes by in which I don't think about or use something these people taught me.

I would like to thank all the friends I have made during my time at Caltech: I consider myself tremendously fortunate to have spent my time with so many talented and interesting people. I would like to particularly thank my friends Max and Spork for allowing me to spend a large part of my "thesis hermitage" in their home; the perpetual contentment exuded by a well-fed cat was the perfect counterbalance to any thesis-related angst.

Finally, I wish to thank my family for their continual love and support, even though I suspect my work must look like gibberish to them; I would not be where I am today if not for them.

Abstract

Many systems comprised of interconnected sub-units exhibit coordinated behaviors; social groups, networked computers, financial markets, and numerous biological systems come to mind. There has been long-standing interest in developing a scientific understanding of coordination, both for explanatory power in the natural and economic sciences, and also for *constructive* power in engineering and applied sciences. This thesis is an abstract study of coordination, focused on developing a systematic "design theory" for producing interconnected systems with *specifiable* coordinated behavior; this is in contrast to the bulk of previous work on this subject, in which any design component has been primarily ad-hoc.

The main theoretical contribution of this work is a geometric formalism in which to cast distributed systems. This has numerous advantages and "naturally" parametrizes a wide class of distributed interaction mechanisms in a uniform way. We make use of this framework to present a model for distributed optimization, and we introduce the *distributed gradient* as a general design tool for synthesizing dynamics for distributed systems. The distributed optimization model is a useful abstraction in its own right and motivates a definition for a distributed extremum. As one might expect, the distributed gradient is zero at a distributed extremum, and the dynamics of a distributed gradient flow must converge to a distributed extremum. This forms the basis for a wide variety of designs, and we are in fact able to recover a widely studied distributed averaging algorithm as a very special case.

We also make use of our geometric model to introduce the notion of *coordination capacity*; intuitively, this is an upper bound on the "complexity" of coordination that is feasible given a particular distributed interaction structure. This gives intuitive results for local, distributed, and global control architectures, and allows formal statements to be made regarding the possibility of "solving" certain optimization problems under a particular distributed interaction model.

Finally, we present a number of applications to illustrate the theoretical approach presented; these range from "standard" distributed systems tasks (leader election and clock synchronization) to more exotic tasks like graph coloring, distributed account balancing, and distributed statistical computations.

Chapter 1

Introduction and Background

Imagine a group of friends, each owing some of the others some amount of money. It is common in these situations for "cyclic" debt relationships to arise, in which A owes B, B owes C, and C owes A some equal amount of money:

$$A \to B \to C \to A$$

As no one enjoys having to pay off debts, it is desirable for the group of friends to find some rearrangement of the debt so that the minimum amount of "paying up" has to be done among the group. Small groups of people solve this problem every day; the solution usually amounts to someone realizing that if they were to coordinate their debts, then everyone would still receive the same net payment, and no one would have to pay anyone anything; this is clearly an improvement over the status quo. A larger group of friends may not have such a clear solution to this problem; in the absence of a central accountant who can observe all the debts, it is possible that the "cycle" of debts may spread over so many people that none of them can individually determine that the total outstanding debt can be reduced, as below:

$$A \to B \to C \to \ldots \to Z \to A$$

While the large social group may not accomplish what a central accountant would, they can perhaps make some local improvements in which "close friends" resolve their debts as well as possible; they can *locally coordinate* their actions in such a way as to improve the *overall welfare* of the group (measured by total outstanding debt), while simultaneously respecting the *global rule* that no one should be required to pay more than they owe or receive less than they are owed.

The "distributed accounting" scenario presented above touches the heart of a fairly complex set of questions, which we will collectively label *distributed coordination.* This work is a study of distributed coordination: how it can be formulated mathematically, when these problems are tractable without global control, and dynamic mechanisms for achieving whatever coordination is possible subject to the constraints imposed by the "locality" of interaction.

1.1 Scope and Overview

This thesis is a study of coordination among multiple, spatially separated, but interconnected agents or systems. In simple terms, we are interested in designing "protocols" for members of some network that allow them to achieve some kind of global coordination, despite the fact that they are constrained by locality of information and the lack of a central planning agent.

The main theme of this work is *systematic design* of distributed coordination mechanisms; we would like to have a general language for specifying coordination tasks, as well as a set of tools for converting that specification into dynamics. We wish to have a sensible model for what it means to be "distributed" that is both natural to intuition and useful from an analysis perspective; there are many different types of distributed systems, and we wish to make as few *a priori* assumptions as possible regarding the way the interconnection network influences the system's dynamics. Given such a modeling framework, we would like to know whether certain coordination tasks are possible whereas others are not for a particular interconnection model.

"Coordination" is a word that has taken on many meanings in the literature; it is frequently used in lieu of, or in concert with, the idea of "cooperation". It is our view that cooperation is a semantic notion, and that it is more appropriate for describing situations in which agents have *a priori* motivations or incentives; our work is strictly aimed at abstract entities without individual goals and so, although one might view the emergence of a useful global behavior as "cooperation", we view it simply as coordinated implementation of a global objective. In this sense, the view we will present is about abstract properties of dynamic coordination as opposed to modeling specific behaviors or interactions; nonetheless, the technical tools we will present will be relevant to "practical" modeling situations, and in particular we will address our motivational "distributed accounting" problem. Thus, although our viewpoint is abstract, the tools developed from this viewpoint will be more broadly applicable than one might expect given their formal origins.

In this work we will ultimately take a somewhat reductionist view of coordination, and approach it as a special variant of optimization. While optimization-based formalisms for coordinated control are abundant, the framework presented herein addresses questions previously unanswered and indeed unformulated in existing work. In the language of this work it is possible to understand, in a geometric setting, fundamental limitations of various network structures in solving various optimization problems. We are also able to give rigorous meaning to the idea of "distributed optimality", which roughly corresponds to a configuration upon which no improvement can be made given the information constraints of some interconnection structure. Skipping ahead to some technical details, a large part of our modeling approach will be the view that distributed interaction over a network imposes certain "tangent space constraints" on the evolution of the system; this abstraction will provide a useful and flexible language for describing our main technical tools.

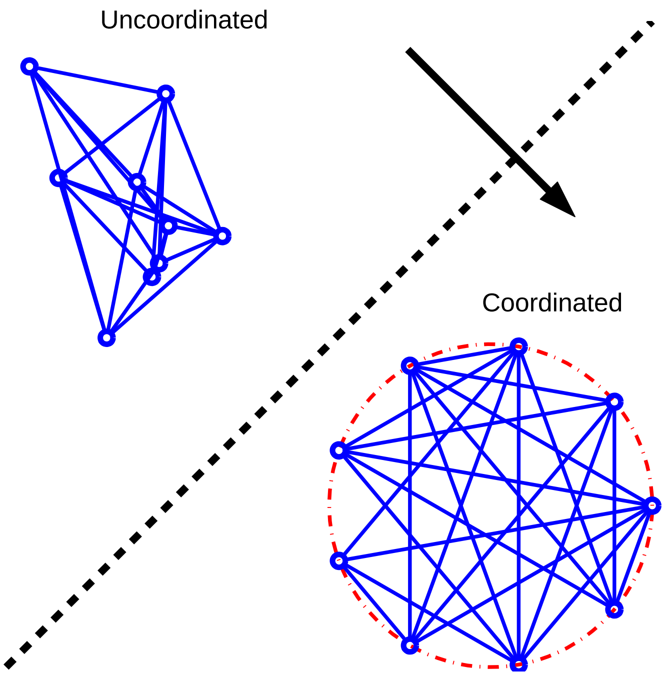


Figure 1.1: Schematic illustration of distributed coordination. Several interconnected units communicate with each other in order to reach some globally organized state.

Chapter 2 collects the basic mathematical requirements for understanding this work and provides a case study that will motivate the approach we take throughout the text. The bulk of the technical contribution of the work is presented in Chapter 3, whereas Chapter 4 presents a number of examples in which the tools from Chapter 3 find useful application. Within Chapter 3, the presentation is organized around four main lines of thought:

- Section 3.1 presents a standard approach to the design of distributed systems: potential functions. Although this is a useful and long-standing tool in the dynamics community, it suffers from some limitations from a design perspective; this will motivate the development of the remaining text.

- Section 3.2 introduces an alternative viewpoint in which we explicitly seek to address some of the limitations of potential-based methods; it introduces all the main abstractions of the work, which allow general and flexible modeling of distributed systems.

- Section 3.3 utilizes the abstract tools developed in Section 3.2 and casts them as components of a basic design framework for distributed systems; it introduces *distributed gradient* dynamics as a general design tool for this purpose.

- Finally, Section 3.4 provides analysis for a useful special case of problems in which we require various extended notions of coordination (tracking, robustness, and reconfiguration).

1.2 Background and Previous Work

Our subject touches the research of many diverse communities, and here we will attempt to provide a short account of previous developments that have informed our work and set the background for the discussion to come.

Perhaps the most relevant community with which to begin our discussion is Computer Science researchers working on parallel and networked computational architectures. Considerable effort was devoted in the 1980s to the development of robust parallel architectures for which systematic proof methodologies could be developed; the need for formal methods (as opposed to ad hoc proof techniques) was motivated by the great complexity arising from multiple asynchronously interacting computational units. Much of this style of work is encapsulated in the book of Chandy and Misra [3], which remains an authoritative reference on the design of parallel systems. This branch of computer science, and this body of research in particular, has had a significant influence on our work in that it has strongly motivated the need for *separation of functional specification from system implementation*. We will revisit this theme repeatedly.

Somewhat later work in the field of parallel and distributed computing addressed problems of coordination under various assumptions regarding reliability of the hardware implementing the parallel algorithms. A large part of this branch of work focused on the need for robustness to network imperfections, and the need to quantify the implications of such imperfections on the performance of a distributed algorithm. Perhaps most importantly, this body of work addressed numerous practical aspects of coordination on a network and established a set of "benchmark" applications (for example, the Byzantine Generals problem) of distributed systems. A representative text providing a thorough treatment of this area is that of Lynch [16]; this has provided motivation for some of our coordination applications in Chapter 4 (clock synchronization and leader election), although it should be noted that our framework is considerably more abstract and so one should not attempt to interpret the connection too literally.

With the rapid spread of the Internet, significant efforts in Computer Science have been devoted not only to understanding parallel algorithms implemented on networks, but the performance of distributed mechanisms on networks themselves; the fundamental objects of study in this field of research have been communication protocols. A representative treatment of this complex branch of computer networking technology can be found in the work of Floyd [7]. Interestingly, much recent effort has examined the role of distributed optimization as a *model* for routing and congestion control protocols on the Internet, as is seen in the works of Kelly, Maulloo, and Tan [13] and of Low and Lapsley [15]. This idea of optimization *as a model* and not merely as a tool has been fundamental in our own work and pervades this entire text.

Similar developments have been seen in the field of computational optimization, which has exhibited considerable interest in parallel and distributed mechanisms in the last twenty years. Again, problems of communication unreliability, delay, and asynchronous operation have been fundamental in this field of research. Much of our view of this area is informed by the classic textbook of Bertsekas and Tsitsiklis [2], which also provides an instance of a "consensus" algorithm we will later denote Φ_L; this branch of research has provided many useful tools for analyzing distributed computational techniques for optimization, many of which have been fundamentally *dynamic* analysis tools, mimicking the structure of Lyapunov theory.

The area of distributed computation for optimization has seen a tremendous resurgence of interest in the last few years, motivated primarily by the increasing availability of networking hardware. Problems that were largely academic when they were first developed are now prime for application with modern communications technology. With this new technological development has come corresponding attention among academic researchers; the number of groups working on this area is too large to account for systematically, but representative studies can be found in the works of Schouwenaars, How, and Ferron [24], Raffard, Tomlin, and Boyd [21], and references therein. Most of these studies have focused on imposing distributed solution methodologies on "classical" optimization; this is a very powerful class of techniques, but we note that it is largely distinct from the view of distributed optimization we will present here, in which we *explicitly model* the underlying distributed architecture and include this within our formalism for optimization.

The technological boom that has rekindled the interest in distributed optimization has also rekindled interest in another "forgotten" field: distributed control systems. This area saw considerable interest in the 1970s, of which the works of Wang and Davison [28] and Corfmat and Morse [4] are representative. Until recently, interest in this field has waned due to the lack of widely available hardware platforms; now it is one of the most active areas of contemporary research in dynamics and control. The "prototypical" problem in this area is that of cooperative formation maintenance; this is a subject of significant research even today, but much progress has been made in developing useful tools for analyzing such systems. A large body of research has stemmed from earlier works such as Swaroop and Hedrick [27], and somewhat more recent interest has developed from works such as Leonard and Fiorelli [14], Jadbabaie, Lin, and Morse [12], and Fax and Murray [6].

In parallel to the developments within computation and communication, and the resulting implications for optimization and control research, similar progress has been made in the development of cheap reliable sensing devices; within the computer science community this has spurred much interest in sensor networks and a slew of associated research topics. Representative works can be found in Estrin, Govindan, Heidemann, and Kumar [5], Akyildiz, Su, Sankarsubramnian, and Cayirci [1], and Heinzelmann, Kulikc, and Balakrishnan [10]. Similar work, although from a systems-oriented perspective, was carried out regarding sensor fusion over networks; a classical reference on this subject which provides a now-standard distributed Kalman filter, is the work of Rao and Durrant-Whyte

[22]. Such applications have motivated our examination data processing in Chapter 4.

The final topic we wish to examine in presenting the background for our work is a flurry of recent interest in a very special system; this has gone by many names and has been studied in many ways, but for most purposes is identical to the aforementioned "agreement" algorithm presented in Bertsekas and Tsitsiklis [2]. Most of the current interest in this subject has been motivated by the works of Fax and Murray [6], and Olfati-Saber and Murray [19]; the former utilized an "agreement-like" algorithm to achieve cooperative control of vehicle formations, whereas the latter studied "consensus" problems on networks. All of these works have made use of an amazing property of the algorithm, that it is extremely robust to changes in the underlying network topology; moreover, recent analytical and experimental work of Mehyar, Spanos, Ponsajapan, Low, and Murray [17], and parallel theoretical developments of Xiao, Boyd, and Lall [30] have shown that this algorithm in fact works very robustly in a realistic distributed asynchronous setting. Much of the motivation for our work lies in a desire to unravel the structure enabling this property, and using that structure to synthesize new systems that execute other useful operations.

There are numerous ongoing developments in the study of these "consensus" algorithms, both in terms of theoretical analysis and design of applications. From the theoretical side, there has been interest in its behavior on "directed" networks, which are networks on which communication links do not provide bidirectional data exchange. This subject was examined in Olfati-Saber and Murray [19], and also in related versions in the works of Moreau [18], and Hatano and Mesbahi [8]. All of these have examined various convergence properties under various assumptions regarding the underlying communications model. Similarly, there has been interesting recent work by Xiao and Boyd [29] on optimal design of certain parameters of this algorithm in order to ensure maximally fast convergence properties. The speed of convergence was also of primary concern in the recent work of Olfati-Saber [23], which examined the behavior of this algorithm on "small-world" networks; a very interesting transition was shown in worst-case convergence rate as a function of connection density and "re-wiring probability" (a measure of how well-mixed the network connections are).

In parallel, there has been rapid development of *applications* using these techniques as an algorithmic base: Xiao, Boyd, and Lall [30] and Spanos and Murray [25] have presented independent but very similar mechanisms for sensor fusion using this scheme. The former is differentiated by a novel analysis of asynchronous behavior, whereas the latter is differentiated by the presence of a *dynamic* mechanism to provide an approximate Kalman Filter (as opposed to a static sensor fusion algorithm). This dynamic mechanism was introduced in Spanos, Olfati-Saber, and Murray [26] and provided a novel treatment of time-varying coordination, as well as a mechanism for robust design and for adaptation to splitting and merging of networks. Though simple, this analysis provided the first steps toward the work that constitutes this thesis; in particular, it was the first (indirect) presentation of what we will later describe as the "Stability—Invariance—Equilibrium" architecture.

1.3 Summary of Contributions

Before proceeding with our exposition, we summarize here the specific technical contributions provided in each of the main theoretical sections of the thesis:

Section 3.1

This section examines potential-based design methods, and provides a characterization of the coordination powers of pairwise quadratic potentials; we show that all such systems accomplish "reflection alignment".

Section 3.2

This section introduces the dual ideas of *interactions* and *coordinations.* These are tangent bundle objects that model the ability of local interactions to influence global state as well as global requirements for coordinated action. We characterize interaction and coordination in terms of dimensionality properties, and introduce the notion of coordination capacity. We also introduce the *distribution operator*, which is an algebraic representation of a distributed interconnection structure. We show that we naturally recover the Laplacian as the distribution operator for a pairwise coordinated interaction. Interactions also provide a natural mechanism for examining distributed optimization, and coordination capacity allows us to characterize whether certain optimizations can be implemented according to a given distributed interaction. We introduce the concepts of distributed extrema and distributed gradients, and show that they are naturally related in the same way that the classical gradient is related to extrema of a function.

Section 3.3

This section presents our overall design procedure, in which we *specify* coordination tasks as optimization problems and synthesize *implementations* based on distributed gradient dynamics. We characterize the convergence of distributed gradient dynamics as well as its equilibrium structure; this naturally provides a complete implementation of an architecture we will call "Stability—Invariance—Equilibrium". Finally, we briefly examine the notion of "distributed efficiency", which in the special case of the Laplacian distribution operator corresponds directly with existing work on spectral properties of graphs.

Section 3.4

This section introduces some extensions to the basic coordination framework in a special framework, which we call "linear coordination". We are able to treat dynamic tracking of coordinated states, robustness of the coordination dynamics, and adaptability to dynamic reconfiguration of a network.

Chapter 2

Mathematical Preliminaries

This chapter covers a variety of mathematical tools that will be used throughout the text, and also sets the notation for the upcoming discussion. The treatment is intended to be minimal, and focused on the concepts necessary to an understanding of the definitions and tools provided in Chapter 3.

Section 2.1 introduces basic ideas from Graph Theory, which will provide a unified abstraction for objects interconnected in some network-like structure. Such structures will form the heart of our discussion throughout the text, and we will be particularly interested in the role played by this interconnection structure in the dynamic behavior of the network.

Section 2.2 covers relevant aspects of Differential Geometry, which provides an analytical framework for representing states and dynamics of distributed systems. We will only make use of very simple concepts that should be accessible to anyone with an understanding of linear algebra and multivariable calculus. The tools of Differential Geometry will provide concise and elegant abstractions for presenting the main results of Chapter 3.

Section 2.3 ties the previous two sections together in order to define two organizational abstractions we will use repeatedly: distributed manifolds and distributed dynamical systems. These objects allow us to formally define the idea of dynamics induced by an interconnection pattern; in subsequent chapters we will provide analysis tools for examining the behaviors of such systems, and various applications where such systems exhibit useful behaviors.

Finally, Section 2.4 presents a case-study of a well known distributed-averaging system. This serves two purposes: first, it presents an archetypical instance of a distributed dynamical system with coordinated behavior; most of the ideas in the remainder of the work amount to generalizing various properties exhibited by this system. Second, we will later revisit this example and cast it in the novel formalism of Chapter 3, and show how its behavior can be cast in a language that permits natural generalization to a wide variety of other coordination tasks.

2.1 Basic Graph Theory

Graphs will form our primary abstraction for systems with interconnection structure. We will only require a small portion of graph theory, and readers with a passing familiarity can easily skip this section; specifically, we will only deal with finite graphs and will only require the standard notions of connectedness.

Definition 1 (Graph). *Let V be a finite set of N elements, labeled by $i \in \{1, 2, \ldots, N\}$. A* graph *is a pair $G = (V, E)$, where E is a set of elements of the form (i, j), with $i, j \in V$. If $(i, j) \in E \Leftrightarrow (j, i) \in E$ we will call such a graph* undirected*; otherwise, we will refer to it as a* directed *graph.*

Thus, a graph is some finite set of objects along with a list of connections between the elements. An undirected graph has the property that the connection relationship is symmetric; if i is connected to j, then j is connected to i.

We will refer to elements of V as *nodes*, and elements of E as *edges* or *links*. When there is no danger of confusion, we will make use of notations such as $i \in G$ or $(i, j) \in G$ (as opposed to $i \in V$ and $(i, j) \in E$).

For the moment, we do not attach any *a priori* significance to the nodes or edges of a graph; we will eventually discuss a objects that will associate state spaces and dynamics to the nodes of a graph, and we will relate properties of the dynamics to the edge structure.

One particular property of the edge structure that will appear repeatedly in our discussion is the *neighbhorhood* of a node: the set of all other nodes to which it is connected by edges.

Definition 2 (Neighborhood). *Let $G = (V, E)$ be a graph. For any node $i \in V$, we define the set $N_i = \{j \mid (i, j) \in E\}$, the* neighbhorhood *of i.*

We will need two more elementary concepts from graph theory, *paths* and *connectedness.*

Definition 3 (Path). *Let $G = (V, E)$ be graph. A* path *in G is a finite sequence $\{v_k\}_{k=0}^{m}$ of nodes in V, with the property that for all $k < m$, $(v_k, v_{k+1}) \in E$. We will say that this is a path from v_0 to v_m, and that it has* length *m.*

Interpreting the graph as a discrete surface, the above definition of a path provides a discrete notion of a continuous path on the surface. This now allows us to define the notion of *connectedness* for a graph:

Definition 4 (Connectedness). *Let $G = (V, E)$ be a graph. We will say that the graph is* connected *if, for each i and j in V, there exists a path from i to j.*

One can associate various matrices to graphs; the study of these matrices and associated linear algebraic properties is known as *algebraic graph theory.* For our purposes, we will only require a few concepts from algebraic graph theory, in order to define the *Laplacian* matrix of a graph.

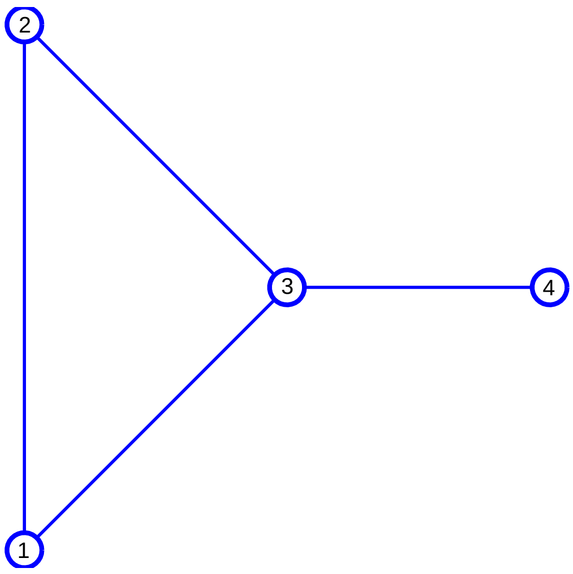


Figure 2.1: A connected undirected graph with four nodes.

Definition 5 (Adjacency Matrix). *Let $G = (V, E)$ be a graph, with $|V| = n$. We define the* adjacency matrix $A_{ij} \in \mathbf{R}^{n \times n}$ *as follows:*

$$A_{ij} \quad = \quad \begin{cases} 1 \Leftrightarrow (i,j) \in E \\ \quad 0 \quad else \end{cases}$$

Clearly, A_{ij} is symmetric if and only if G is an undirected graph.

Definition 6 (Degree Matrix). *Let $G = (V, E)$ be a graph, with $|V| = n$. We define the* degree matrix D_{ij}*:*

$$D_{ij} \quad = \quad \begin{cases} \sum_j A_{ij} \Leftrightarrow i = j \\ \quad 0 \quad else \end{cases}$$

We will occasionally need to explicitly discuss the dependence of A and D on the underlying graph, and in such cases will write $A(G)$ and $D(G)$ respectively.

Definition 7 (Laplacian Matrix). *Let $G = (V, E)$ be a graph, with $|V| = n$. We define the* Laplacian Matrix *of the graph, $L \doteq D - A$.*

As an example, consider the graph of Figure 2.1. The edges of this graph are $(1, 2), (2, 1), (2, 3)$,

$(3,2),(3,1),(1,3),(3,4),(4,3)$. This yields the following adjacency, degree, and Laplacian matrices:

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

The Laplacian matrix is a graph-theoretic version of the classical Laplacian or diffusion operator ∇^2; we will return to this parallel later. For now, we collect a few results that will be of use throughout our discussion, but before doing so we introduce a few linear algebraic notations that will enable us to make our presentation compact.

Definition 8 (Some Notation).

1. *We define $\mathbf{e}_i \in \mathbf{R}^n$ to be the vector with a 1 in the i-th component, and 0 elsewhere.*

2. *For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbf{R}^n$, we define $\mathbf{x} \otimes \mathbf{y}$ to be the matrix defined by $(\mathbf{x} \otimes \mathbf{y})\,\mathbf{v} = \langle \mathbf{y}, \mathbf{v} \rangle \mathbf{x}$, where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product. If all elements of $\mathbf{R}^n$ are taken to be column vectors, then $(\mathbf{x} \otimes \mathbf{y}) = \mathbf{x}\mathbf{y}^T$.*

3. *We define $\mathbf{1}_n$ and $\mathbf{0}_n$ in $\mathbf{R}^n$ to be the vectors of all ones and all zeros, respectively.*

Proposition 1 (Fundamental Properties of the Laplacian Matrix). *Let $G = (V, E)$ be a connected graph with $|V| = n$, and let L be the associated Laplacian matrix; then, we have:*

1. *L is positive semidefinite.*

2. *L has a one-dimensional nullspace, along $\mathbf{1}_n$*

3. *There exists a one-dimensional subspace of vectors along $\mathbf{q} \in \mathbf{R}^n$ such that $\mathbf{q}^T L = \mathbf{0}_n$; if G is undirected, $\mathbf{q}$ can be taken to be $\mathbf{1}_n$.*

Proof. To see the first point, observe that L can be written as a sum of rank one matrices, each corresponding to an edge in G:

$$L \;=\; \sum_{(i,j)\in E} \mathbf{e}_i \otimes \mathbf{e}_i - \mathbf{e}_i \otimes \mathbf{e}_j.$$

The vectors $\mathbf{e}_i + \mathbf{e}_j$ and $\mathbf{e}_i$ are eigenvectors of the matrix associated with the edge (i,j), with eigenvalues 0 and 1 respectively. Thus each such matrix is positive semidefinite, and the sum of positive semidefinite matrices is also positive semidefinite.

The annihilation of $\mathbf{1}_n$ follows directly from the definition of the Laplacian matrix as $D - A$; simply note that $L\mathbf{1}_n = D\mathbf{1}_n - A\mathbf{1}_n$ and the i-th component of this vector is $D_{ii} - \sum_j A_{ij} = 0$.

To show that this is the only such subspace, recall the decomposition above of L into a sum of matrices associated with edges, and consider the quadratic form $\mathbf{v}^T L\mathbf{v}$. Note that for any nullspace vector, this quadratic form must evaluate to 0. However, since each term in the sum is positive semidefinite, any nullspace vector must be annihilated by each term; this implies that for each edge $(i, j) \in E$, the i and j components of any nullspace vector must be equal. Now, recall that G is a connected graph, and that there exists a path between any two nodes; this implies that in fact every pair of components i and j in any nullspace vector must be equal (thus, it must be parallel to $\mathbf{1}_n$).

The final point is a standard result from linear algebra; since L has a one-dimensional nullspace, so does L^T. When G is undirected, L is symmetric, and so the left nullspace is the same as the right nullspace. □

These properties will be utilized in Section 2.4 in discussing the behavior of a prototypical distributed coordination mechanism. We will also revisit the Laplacian in Chapter 3, and show the above properties as natural consequences of the Laplacian being a *distributed gradient.*

2.2 Elements of Differential Geometry

In this section we will introduce basic concepts from differential geometry, but a few comments are in order before proceeding. General differential geometry is quite dense, both conceptually and in its notation; despite its power and elegance, the aesthetic benefit of the abstract differential geometric formalism seems, in our case, vastly outweighed by the amount of machinery necessary to cast our work in that language. As a consequence, we will present a highly simplified version that should be accessible with standard multivariable calculus and linear algebra in $\mathbf{R}^n$. In particular:

1. We will not be using any index conventions, either pertaining to summation or to subscripts and superscripts.

2. We will only work with subsets of $\mathbf{R}^n$, and so will never need to introduce charts; we will also exploit the Euclidean inner-product associated with $\mathbf{R}^n$, but we will not make any reference to general Riemmanian manifolds.

3. We will not deal with exterior calculus, tensors, or with cotangent-space objects, even though some of the concepts herein are more compactly represented in that language. In particular, we will frequently see things like ∇f instead of $\mathbf{d}f$.

With these caveats in mind, we can now proceed to define the concepts relevant to our development; we begin with the notion of a smooth manifold. In order to make this definition, we recall a bit of notation: for a mapping $\mathbf{f}$ from $\mathbf{R}^n$ to $\mathbf{R}^m$, we define $\mathbf{Df}(\mathbf{x})$, to be the matrix of partial derivatives of $\mathbf{f}$ evaluated at $\mathbf{x}$, $\mathbf{Df}(\mathbf{x})_{ij} = \frac{\partial \mathbf{f}_i}{\partial \mathbf{x}_i}(\mathbf{x})$.

Definition 9 (Smooth Manifold). *A* smooth manifold*, M, is a subset of $\mathbf{R}^n$ defined by an equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}_m$, where $\mathbf{f}$ is a smooth mapping from $\mathbf{R}^n$ to $\mathbf{R}^m$ for some m, with the property that* $\text{rank}(\mathbf{Df}) = m \ \forall \ \mathbf{x} \ \in M$. *We will call say that M has* dimension $n - m$, *and* codimension m.

Thus a manifold is a surface in $\mathbf{R}^n$ with the property that at each point $\mathbf{x}$, it is approximated by a $n - m$ dimensional vector space (the space defined by the linear equations $\mathbf{Df}(\mathbf{x})\mathbf{v} = \mathbf{0}_m$). This naturally motivates the definition of the tangent space of M at $\mathbf{x}$, denoted $T_{\mathbf{x}}M$.

Definition 10 (Tangent Space at a Point). *Let M be a smooth manifold defined by an equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}_m$. We define the* Tangent Space *at $\mathbf{x}$, $T_{\mathbf{x}}M$ to be the set of elements $(\mathbf{x}, \mathbf{v})$, with $\mathbf{v} \in \mathbf{R}^n$, such that:* $\mathbf{Df}(\mathbf{x})\mathbf{v} = \mathbf{0}_m$.

This has a natural vector space structure defined by $(\mathbf{x}, \mathbf{v}) + (\mathbf{x}, \mathbf{w}) = (\mathbf{x}, \mathbf{v} + \mathbf{w})$, and from our above discussion, it is clear that the dimension of this space is $n - m$; we have thus formalized our concept of a manifold being a surface which is locally approximated by a $n - m$ dimensional vector space. We can now piece together all such local approximations to obtain a fundamental concept of differential geometry, the *Tangent Bundle*:

Definition 11 (Tangent Bundle of a Manifold). *Let M be a smooth manifold. We define the* Tangent Bundle *of M, TM to be the set of all elements $(\mathbf{x}, \mathbf{v})$ with $\mathbf{x} \in M$ and $(\mathbf{x}, \mathbf{v}) \in T_{\mathbf{x}}M$.*

With the notions of Tangent Spaces and the Tangent Bundle in hand, we can define *vector fields* and *tangent subbundles.*

Definition 12 (Vector Field). *Let M be a smooth manifold. By a* vector field *on M, we mean a mapping $X : \mathbf{x} \in M \to (\mathbf{x}, \mathbf{v}(\mathbf{x})) \in T_{\mathbf{x}}M$.*

A vector field simply attaches a tangent vector to every point on the manifold M; in the following section we will view this as imposition of dynamics on M. In addition to vector fields, which assign individual tangent vectors at each point of the manifold, we will also need to discuss *tangent subbundles* which assign a subspace of tangent vectors to each point of the manifold.

Definition 13 (Tangent Subbundle). *Let M be a manifold of dimension n, and let TM be its tangent bundle. A* Tangent Subbundle $B_{\mathbf{x}}$ *of dimension m assigns to each point in the manifold a set of elements of the form $(\mathbf{x}, \mathbf{v})$ such that, for each $\mathbf{x}$, the set $B_{\mathbf{x}}$ is a subspace of $T_{\mathbf{x}}M$ with dimension m. We will say that $B_{\mathbf{x}}$ has* dimension m *and* codimension $n - m$.

Tangent Subbundles will provide a unified language in which to describe the two main abstractions of Chapter 3, *interactions* and *coordinations.*

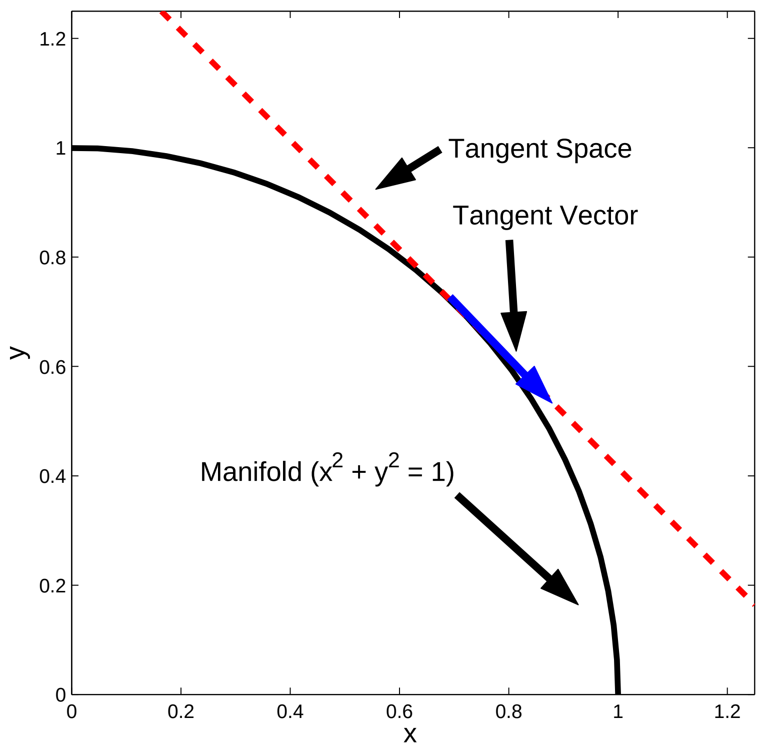


Figure 2.2: A Manifold, a Tangent Vector, and a Tangent Space

2.3 Distributed Manifolds and Dynamical Systems

In this section we present some unifying notation and terminology, in order to be able to discuss distributed dynamics in a systematic way. The underlying motivation lies in the necessity to talk about distributed dynamics for a family of graphs simultaneously, and while it is possible (and in fact, frequently done in the literature) to specify such systems piecemeal, it becomes very cumbersome when one must assign differing state-spaces and dynamics to each node in the graph. The terminology presented below is an attempt to provide a general framework in which to discuss all such systems.

We begin, then, with the notion of a *distributed manifold*, which one can think of as a mapping from all possible interconnection structures to a set of states associated with each node.

Definition 14. *A* distributed manifold *is a mapping from a set of graphs* Γ *to a set of manifolds of the form*

$$M(G) = \prod_{i \in G} M_i$$

where each M_i *is a manifold, and* $\prod$ *denotes the Cartesian product.*

Thus, a distributed manifold assigns, for each a graph, a manifold to each node in the graph; we will frequently refer to this manifold as the *local state space* of the node, and we will denote it as above by M_i or $M_i(G)$ when we wish to make the graph dependence explicit. We will denote the local state by $\mathbf{x}(i) \in M_i(G)$; note the use of the parenthetical notation (as opposed to a subscripted notation), which is somewhat unorthodox in the distributed systems literature. When referring to the distributed manifold itself, rather than the image of a specific graph $M(G)$, we will use the notation $M(\cdot)$. Clearly, if $M(\cdot)$ is a distributed manifold, $M(G)$ is a manifold for each G.

As a canonical example of a distributed manifold, consider the assignment $M_i(G) = \mathbf{R}$, which assigns a real-valued state to each node in the graph; for a given $G = (V, E)$, $M(G)$ is then the space of all real-valued functions over V. Similarly, one can consider a distributed manifold of the form $M_i(G) = \mathbf{S}_1$, which assigns an element of the unit circle to each node of the graph; such systems arise, for example, in the study of distributed oscillator synchronization such as in the Kuramoto equation.

The examples presented above are straightforward, and in some sense do not justify the introduction of additional terminology and notation; however, we will have occasion to discuss systems in which the state spaces at each node can be quite different (and in fact, possibly varying in time), so it is necessary to have an abstraction sufficiently general to be able to deal with all such systems in a uniform fashion.

We now introduce the notion of a *distributed dynamical system*, which associates dynamics to a distributed manifold. In order to obtain some perspective, one can imagine a distributed dynamical system as some kind of "high level" abstraction of temporal behavior that is spread over some kind of spatial structure; continuous analogues would include diffusion, wave propagation, fluid flow, and a variety of other continuum theories which associate spatiotemporal dynamics to surfaces. A distributed dynamical system, then, can be considered a discrete analogue of these continuum theories.

Definition 15. *A* distributed dynamical system Φ *is a mapping from a set of graphs* Γ *to a set of pairs* $(M(G), X(G))$*, where* $M(\cdot)$ *is a distributed manifold, and* $X(G)$ *is a vector-field over* $M(G)$ *for each* $G \in \Gamma$*.*

Again, we will use the notation $\Phi(\cdot)$ when referring to the distributed dynamical system, as opposed to the image of a particular graph $\Phi(G)$. For a particular choice of G, $\Phi(G)$ is a pairing of a manifold and a vector field on that manifold, which implies an associated dynamical system, or flow. When we wish to refer to the time evolution of some particular local state, we will use the notation $x(i, t)$.

Thus, a distributed dynamical system is simply an organizational mechanism for describing states and dynamics associated with a variety of graph structures. In Chapter 4, we will describe several applications which will involve the interplay of a variety of distributed dynamical systems, and so it will be useful to be able to refer to them compactly, while simultaneously parametrizing our results over a family of interconnection patterns.

As a trivial example of a distributed dynamical system, one can consider

$$\begin{aligned} M_i(G) &= \mathbf{R} \\ \dot{x}(i) &= f(x(i)) \end{aligned}$$

for some fixed function f; this is a distributed dynamical system which associates to each node a real-valued state-space and dynamics completely decoupled from that of every other node. The remainder of our work will focus on distributed dynamical systems exploiting coupling between nodes to effect some kind of spatiotemporal coordination.

2.4 Case Study: A Distributed Averaging Protocol

In this section we examine a commonly studied distributed dynamical system (see Olfati-Saber and Murray [19] for an extensive treatment) with the interesting property that it asymptotically carries out an averaging operation. This system, and numerous variants, have motivated a renewed interest in the theory of distributed systems among the dynamics community, both because of its simplicity and elegance, but also because it seems to offer the promise of a robust distributed computational tool which can naturally be treated as part of a larger, spatially distributed dynamical process.

Consider then, the following distributed dynamical system: for a graph $G = (V, E)$, we assign a real valued local state space $M_i = \mathbf{R}$ to each node, and impose the following dynamics for each node:

$$\dot{x}(i) = \sum_{j \in N_i} \left(x(j) - x(i)\right).$$

We denote this system $\Phi_L(\cdot)$, for reasons that will soon become apparent.

Let us consider, for the moment, the particular dynamical system $\Phi_L(G)$ for some fixed, connected, and undirected G with n nodes. Let us denote the concatenated state vector by $\mathbf{x} \in \mathbf{R}^n$ (i.e. $\mathbf{x}_i(t) = x(i, t)$). From the above specification for the dynamics of $x(i)$, we obtain the following dynamics for $\mathbf{x}$:

$$\dot{\mathbf{x}} = -L(G)\mathbf{x}.$$

Now, we recall from Section 2.1 that L is a positive semidefinite matrix (and so $-L$ has non-positive real eigenvalues); this implies that for arbitrary initial conditions $\mathbf{x}(0)$, $\mathbf{x}$ converges to some limiting value $\mathbf{x}(\infty)$. Now, recalling that $\mathbf{1}_n$ is a basis for the nullspace of L, we know that $\mathbf{x}(\infty) = \lambda \mathbf{1}_n$ for some $\lambda \in \mathbf{R}$. Finally, we note that

$$\begin{aligned} \mathbf{1}_n^T \dot{\mathbf{x}} &= -\mathbf{1}_n^T L(G)\mathbf{x} \\ &= \mathbf{0}_n^T \mathbf{x} \\ &= 0. \end{aligned}$$

This in turn implies:

$$\begin{aligned} \mathbf{1}_n^T \mathbf{x}(0) &= \mathbf{1}_n^T \mathbf{x}(\infty) \\ &= \lambda \mathbf{1}_n^T \mathbf{1}_n \\ &= \lambda n \end{aligned}$$

From the above equation, we see that for each i,

$$x(i,t) \to \frac{1}{n} \sum_{i \in G} x(i,0),$$

i.e. the average of the initial values. This system has, in essence, performed a distributed computation, with inputs specified by the initial conditions, and an identical output that is asymptotically reached by every node. Further, this system has the interesting property that it is robust to dynamic switches in the topology. To see this, let us suppose that the graph $G(t) = (V, E(t))$ has a time-varying edge structure, but that it is always connected; then we have

$$\begin{aligned} \frac{d}{dt}\frac{1}{2}\left\| \mathbf{x} - \frac{\mathbf{1}_n \mathbf{1}_n^T}{\mathbf{1}_n^T \mathbf{1}_n}\mathbf{x} \right\|^2 &= -\mathbf{x}^T L(G(t))\mathbf{x} \\ &\leq -\lambda_2(t) \left\| \mathbf{x} - \frac{\mathbf{1}_n \mathbf{1}_n^T}{\mathbf{1}_n^T \mathbf{1}_n}\mathbf{x} \right\|, \end{aligned}$$

where $\lambda_2(t)$ is the smallest non-zero eigenvalue of $L(t)$. While different edge structures will imply different values for λ_2, for a finite number of nodes there are only finitely many possibilities and so there is some minimum value. Since this minimum value is positive, we see that the function above is in fact a Lyapunov-like function for this system; wherever it is nonzero, it decreases. We thus see that the system robustly executes the averaging operation for an arbitrary time-varying edge structure, provided that the network remains connected.

While the above properties are interesting in themselves, the more important point is that these are not properties of the specific differential equations associated with a particular G, but rather properties of the distributed dynamical system $\Phi_L(\cdot)$ on undirected graphs. It is, in fact, very like

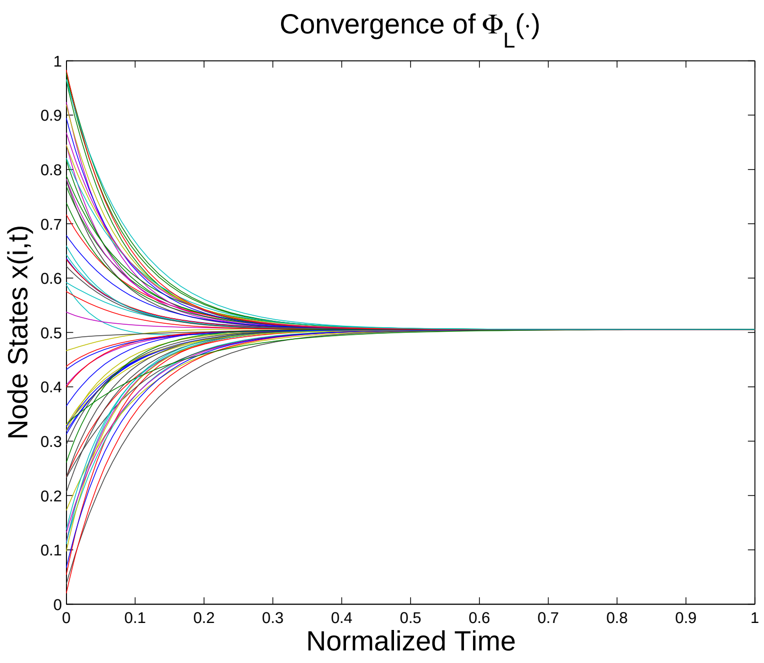


Figure 2.3: Convergence behavior of the distributed averaging system $\Phi_L(\cdot)$

.

the classical diffusion equation:

$$\frac{\partial f(x,t)}{\partial t} = -\nabla^2 f(x,t).$$

There is more to this analogy than mere formal similarity, or the fact that the Laplacian matrix L is frequently used as a discrete approximation to the diffusion operator. The real power of classical diffusion theory (and that which we seem to have recovered here) is that its qualitative dynamics does not depend on the surface over which the diffusion process occurs; the asymptotic flattening of gradients is a robust feature that is independent of the underlying surface. Indeed, this kind of structure is prevalent in physical theories: the minimization of certain functions (potentials), and the conservation of certain quantities (energy, heat, mass, momentum).

The fundamental question, then, is whether we can build a “coordination design theory” that provides the robust qualitative dynamics observed for $\Phi_L(\cdot)$, while giving us the power to specify other kinds of coordination criteria. In order to get a better idea for how we might conceive of a theory that generalizes the behavior of $\Phi_L(\cdot)$, we observe three fundamental facts that drive its coordinating behavior:

Stability

The system provably converges to some limiting value, because some bounded function of the states decreases monotonically.

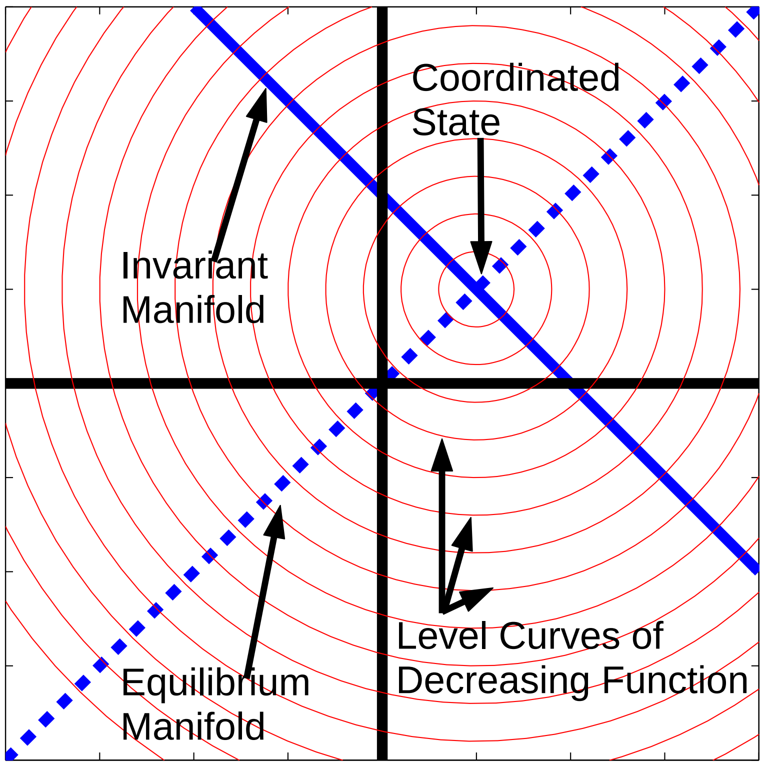


Figure 2.4: Illustration of the interplay between stability, invariance, and equilibrium structure.

A Structured Equilibrium Set

The limiting value is constrained to be in some submanifold of the overall state space.

An Invariant Manifold

The states evolve in some smaller-dimensional submanifold of the overall state space (corresponding to a dynamically conserved quantity).

These properties suffice to prove that the system must converge to a point in the intersection of the equilibrium manifold and the invariant manifold. This analysis is deceiving in its simplicity; note that we have only made use of very abstract properties of the system, and have not at any time made use of details such as the linearity of the dynamics, or the structure of the graph.

The "Stability—Invariance—Equilibrium" architecture suggests a very powerful and general formalism for designing mechanisms for distributed coordination: if one can control the equilibrium and invariant manifolds of some distributed dynamical system, while ensuring that the system is stable (in the sense described above), one effectively has a language for specifying coordination dynamics. The remainder of the thesis will show how one can in fact accomplish these goals for a wide variety of systems; the resulting formalism provides techniques for synthesizing distributed coordination, as well as a systematic language for proving what kinds of coordination are possible under different models for distributed interaction.

Chapter 3

Dynamic Coordination: Principles

This chapter examines in detail the underpinnings of distributed dynamic coordination, and introduces the main theoretical contributions of the thesis.

Section 3.1 presents what one might call the "classical" approach to designing the dynamics of distributed systems. This approach associates an interaction potential between adjacent members of the network, and the resulting dynamics is simply the gradient flow of the sum of the potentials (or the Hamiltonian flow derived from the generalized forces implied by the potential). This is a well-studied approach, and has a long history in the physical sciences; however, it suffers from some rather serious drawbacks from a design perspective, mostly stemming from the fact that it imposes an *a priori* coupling between the coordination to be achieved and the interaction structure on which it is implemented.

Sections 3.2 and 3.3 present a new viewpoint based on a differential geometric formalism, and introduce the dual notions of *interaction* and *coordination.* This abstraction will allow us to rigorously examine the idea of distributed optimality, and also to introduce a concept that will be fundamental tool for design, the *distributed gradient.* We will revisit the distributed averaging system $\Phi_L(\cdot)$ introduced in Chapter 2 and show that it is in fact a distributed gradient flow for a natural optimization problem that is *independent* of the underlying network structure; this allows us to achieve the desired goal of separating overall coordination functionality from the details of the implementation on the network. Further, this mechanism will allow us to parametrize coordinated behavior across an arbitrary distributed interaction pattern, as opposed to being confined to pair-wise interactions.

Finally, Section 3.4 shows how to robustly achieve "dynamic" and "reconfigurable" coordination" for linear systems, in which a quadratic global objective is optimized subject to linear constraints. The "dynamic" component allows tracking of time-varying coordinated states, and reconfigurability allows the nodes to adapt to dynamic network structure.

3.1 The “Classical” Viewpoint: Potential Functions

In this section we briefly examine the use of potential functions as a mechanism for designing distributed coordination mechanisms. We begin by discussing the motivation for potential-based designs, and by casting the distributed averaging system $\Phi_L(\cdot)$ as a potential-based system stemming from a one-variable quadratic potential. We then examine the class of all multi-variable quadratic potentials, and characterize their coordinating abilities; unfortunately, these systems only achieve a relatively small class of coordinated behaviors. Finally, we discuss some limitations of the potential based approach; the remainder of the chapter proposes an alternative viewpoint which addresses these weaknesses, and provides a fairly general framework for designing distributed systems through a combination of distributed optimization and differential geometry.

3.1.1 Potentials as a Source of Distributed Systems

Potential-based interactions present an attractive formalism for distributed systems, in part because of their basis in the physical sciences, but also because they give a systematic way to “distribute” dynamics over a large number of interacting components. Although the idea underlying potential-based systems is common currency in many scientific fields, we will explore it here in a context that will motivate our upcoming work.

At the heart of potential-based designs is the idea of a gradient flow, dynamics which follows the gradient of some function defined over all the possible configurations of some set of variables. The power of this formalism is in its “additivity”; one can simply add up potential functions and obtain new gradient flows. This enables one to specify the dynamics of some large interconnected system by, for example, associating a potential function to each edge and then adding together all such local potentials. One then obtains a global potential function that couples the states of all the members of the network, and induces global dynamics for these states. If the local potentials are chosen so as to depend only on local variables, this naturally provides a mechanism for distributed dynamics, in which each node in the network updates its state according to information that is locally observable. This situation is illustrated in Figure 3.1.

Local potential functions thus provide a flexible framework in which to define distributed dynamical systems; one can certainly imagine a wide variety of candidates for local potential functions, and each one induces an associated distributed dynamical system of the form:

$$\dot{\mathbf{x}}(i) = -\sum_{j \in N_i} \frac{\partial}{\partial \mathbf{x}(i)} f(\mathbf{x}(i), \mathbf{x}(j)),$$

where f is the local potential function, and the notation $\frac{\partial}{\partial \mathbf{x}(i)}$ means the vector of partial derivatives with respect to the components of $\mathbf{x}(i)$. Potential systems also have the advantage of providing

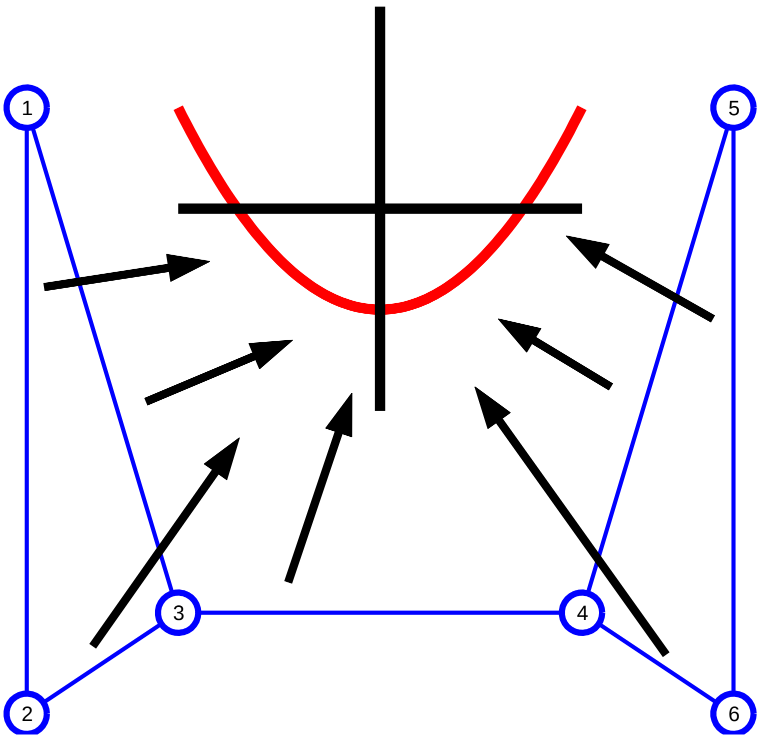


Figure 3.1: Illustration of a distributed system with a local potential function on each link; the aggregate potential is the sum of all the local potentials.

straightforward convergence analysis; since the global potential function decreases monotonically, then an *a priori* lower bound on the function suffices to prove that the system will converge to an equilibrium (or a family of equilibria).

One can interpret the distributed averaging system $\Phi_L(\cdot)$ as arising from a local potential function:

$$f(x,y) = \frac{1}{2}(x-y)^2.$$

Associating such a function to each edge in some graph G, and computing the associated partial derivatives, we obtain

$$\begin{aligned}\dot{x}(i) &= -\sum_{j\in N_i} \frac{\partial}{\partial x(i)} \frac{(x(i)-x(j))^2}{2} \\ &= \sum_{j\in N_i} \left(x(j)-x(i)\right),\end{aligned}$$

which is precisely the dynamics we introduced for $\Phi_L(G)$ in Chapter 2. With this view in mind, the previous comments regarding the use of potentials as a framework for describing distributed systems naturally motivates two fundamental questions about this design paradigm:

- The system $\Phi_L(\cdot)$ provides one kind of distributed coordination, and is induced by an appropriate choice of a local potential; what other kinds of coordination capabilities can one achieve by choosing different potential functions?

- The coordinating and computational powers of $\Phi_L(\cdot)$ are not particularly transparent when viewed from the "local potential" viewpoint; how can one provide a design formalism that allows more explicit specification of the desired coordination, and which systematically synthesizes an appropriate distributed dynamical system?

We will partially address the first question in the following section; in the remainder of the chapter, we will return to the "Stability—Invariance—Equilbrium" viewpoint presented in Chapter 2, and combine it with ideas from differential geometry and distributed optimization to obtain a basic theory addressing the second point.

3.1.2 Characterization of Pairwise Quadratic Potentials

In this section we present a complete characterization of the coordinating power of pairwise quadratic potentials. The main result we will show is that such systems all accomplish a type of coordination we call "reflection alignment": the states converge to a subspace in which every pair of neighbors is related by a reflection, according to

$$\begin{aligned} \mathbf{x}(i) &= R\mathbf{x}(j) \\ R^2 &= I_{n\times n} \\ \mathbf{x}(i) &\in \mathbf{R}^n. \end{aligned}$$

Let us begin, then, by formally stating our assumptions:

Dynamics Structure

We will be considering a family of distributed dynamical systems parametrized by a symmetric matrix $P \in R^{2n\times 2n}$ for some positive integer n; we will denote these systems by $\Phi_P(\cdot)$. For a graph $G = (V, E)$, the local state space for each $i \in V$ will be $\mathbf{R}^n$, and we will denote each local state by $\mathbf{x}(i) \in \mathbf{R}^n$.

Symmetric Interaction Potential

We will assume that the matrix P defines a positive-semidefinite quadratic form on $\mathbf{R}^{2n}$; partitioning P into $n \times n$ blocks as

$$P = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

the requirement that this define a quadratic form implies that A and D must be symmetric, and $B = C^T$. The positive-semidefiniteness requirement implies that A and D are positive-semidefinite.

We will interpret this matrix as defining an interaction potential between every pair of neighbors i and j, according to:

$$\begin{pmatrix} \mathbf{x}(i) \\ \mathbf{x}(j) \end{pmatrix}^T \begin{pmatrix} A & B \\ B^T & D \end{pmatrix} \begin{pmatrix} \mathbf{x}(i) \\ \mathbf{x}(j) \end{pmatrix}.$$

Identity Independence

We assume that the above quadratic form is invariant under relabeling of i, and j, i.e.:

$$\begin{pmatrix} A & B \\ B^T & D \end{pmatrix} = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \begin{pmatrix} A & B \\ B^T & D \end{pmatrix} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} = \begin{pmatrix} D & B^T \\ B & A \end{pmatrix},$$

where we have exploited the previous requirement that $C = B^T$. This implies that B must be a symmetric matrix, and also that $A = D$.

Unambiguous Coordination

We will assume that for any specific choice of $\mathbf{x}(i)$, there exists a unique choice of $\mathbf{x}(j)$ that achieves the global minimum of the quadratic form. Using some standard linear algebraic arguments, this can be shown to imply that A and B must be full-rank matrices.

Gradient Dynamics

For a given graph $G = (V, E)$ the dynamical system system $\Phi_P(G)$ assigns a state $\mathbf{x}(i) \in \mathbf{R}^n$ to each $i \in V$, and the resulting dynamics is the gradient flow for the following potential function:

$$\sum_{(i,j)\in E} \begin{pmatrix} \mathbf{x}(i) \\ \mathbf{x}(j) \end{pmatrix}^T \begin{pmatrix} A & B \\ B & A \end{pmatrix} \begin{pmatrix} \mathbf{x}(i) \\ \mathbf{x}(j) \end{pmatrix}.$$

Specifically, we have:

$$\dot{\mathbf{x}}(i) \quad = \quad -\sum_{j\in N_i} \left(A\mathbf{x}(i) + B\mathbf{x}(j)\right).$$

As can be seen from the final equation above, these assumptions in fact highly constrain the types of dynamics that can arise, and so some commentary is in order to justify these choices. We have associated a real multivariable state to each member of the network, and defined an interaction potential between every pair of neighbors, which induces an associated gradient flow; this portion of the assumptions is standard. We have further assumed that the interaction potential is agnostic of the specific identity of the nodes, which requires that the potential structure depend only on the interconnection pattern. The only non-standard assumption is the requirement that minimizing the potential while holding one neighbor's value fixed yield a one-to-one mapping between neighbor states, which is essentially a well-posedness condition; for a network involving only two nodes, we require that specification of one's state uniquely specify the *coordinated* state of the pair.

We have already seen one system of the form above, the distributed averaging system $\Phi_L(\cdot)$; one can recover $\Phi_L(\cdot)$ with the choice $A = 1$ and $B = -1$. We now proceed to analyze the possible coordinated states imposed by this entire family of distributed dynamical systems; we begin with a technical lemma.

Lemma 1 (Idempotence of $A^{-1}B$). *Consider an instance of $\Phi_P(\cdot)$ for some choice of P, and let A and B be the matrices specifying P as above; then, we have*

$$\left(A^{-1}B\right)^2 = I.$$

Proof. From the "unambiguous coordination" assumption, we have that for every $\mathbf{x}$ there is a unique $\mathbf{y}$ such the expression

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}^T \begin{pmatrix} A & B \\ B & A \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

is globally minimized; computing the gradient of the above expression, we have the following set of linear equations:

$$\begin{pmatrix} A\mathbf{x} \\ B\mathbf{x} \end{pmatrix} = \begin{pmatrix} -B\mathbf{y} \\ -A\mathbf{y} \end{pmatrix}.$$

Exploiting the invertibility of A and B, we have:

$$\begin{aligned} \mathbf{x} &= -A^{-1}B\mathbf{y} \\ &= \left(-A^{-1}B\right)\left(-A^{-1}B\mathbf{x}\right) \\ &= \left(A^{-1}B\right)^2 \mathbf{x} \end{aligned}$$

for all $\mathbf{x}$; this implies that $\left(A^{-1}B\right)^2 = I$ as desired. $\square$

With this result in hand, we can prove the main result of this section.

Proposition 2 (Reflection Alignment). *Consider an instance of $\Phi_P(\cdot)$ as specified above. Then, there exists a fixed reflection matrix $R(P) \in \mathbf{R}^{n\times n}$ such that, for every connected graph $G = (V, E)$, $\Phi_P(G)$ has the following property:*

$$\lim_{t\to\infty} \mathbf{x}(i,t) = \lim_{t\to\infty} R\mathbf{x}(j,t) \quad \textit{for all } j \in N_i$$

for all $i \in G$. If P is expressed as matrices A and B as above, then $R = -A^{-1}B$.

Proof. First recall that for each G, $\Phi_P(G)$ is a gradient flow for a positive-semidefinite potential function, and so the $\mathbf{x}(i)$ vectors must converge to a limiting value. Let us consider any node i^* whose limiting value $\lim_{t\to\infty} \mathbf{x}(i^*, t)$ has maximal Euclidean norm among all the nodes in the network. Then, in the limit we have:

$$|N_{i^*}|\mathbf{x}(i^*) = \sum_{j\in N_{i^*}} -A^{-1}B\mathbf{x}(j)$$

where we have dropped the explicit t dependence, and $|N_{i^*}|$ denotes the number of elements of N_{i^*}. Now, since $A^{-1}B$ is idempotent, it is unitary, and so the norm of each term in the sum on the right-hand size is bounded above by the norm of $\mathbf{x}(i^*)$. Since the norm of the right-hand side must equal the norm of the left-hand side, and there are exactly $|N_{i^*}|$ terms in the sum, we must have

$$\mathbf{x}(i^*) = -A^{-1}B\mathbf{x}(j) \text{ for all } j \in N_{i^*}.$$

Now, each member of N_{i^*} must have a state vector $\mathbf{x}(j)$ with norm equal to that of $\mathbf{x}(i^*)$ (i.e. they are also maximal vectors in the Euclidean norm). This allows us to repeat the preceding argument for all members of N_{i^*}, extending the above relation to all nodes within two hops of i^*; applying this recursively, and exploiting the connectedness of G, we now have that

$$\mathbf{x}(i) = -A^{-1}B\mathbf{x}(j) \text{ for all } (i,j) \in E.$$

This situation is depicted in Figure 3.2. Finally, the fact that $-A^{-1}B$ is a reflection follows from the fact that it is its own inverse; this completes the proof. □

This proposition achieves the goal of characterizing the coordinating power of an arbitrary distributed dynamical system arising from pairwise quadratic interaction potential; however, this is only part of the picture. We know that $\Phi_L(\cdot)$ is one such system, and we also know that this system not only drives each member of the network to the same state, but it also carries out an averaging computation. While we have characterized the coordinated states as a particular subspace, we have not yet discussed which point in this space is selected; unfortunately, except in the case where $-A^{-1}B = I$, the answer depends critically on the underlying graph structure. We will return to answer this question (and exploit this sensitivity) in Section 4.3, where we discuss applications of distributed dynamical systems to graph coloring.

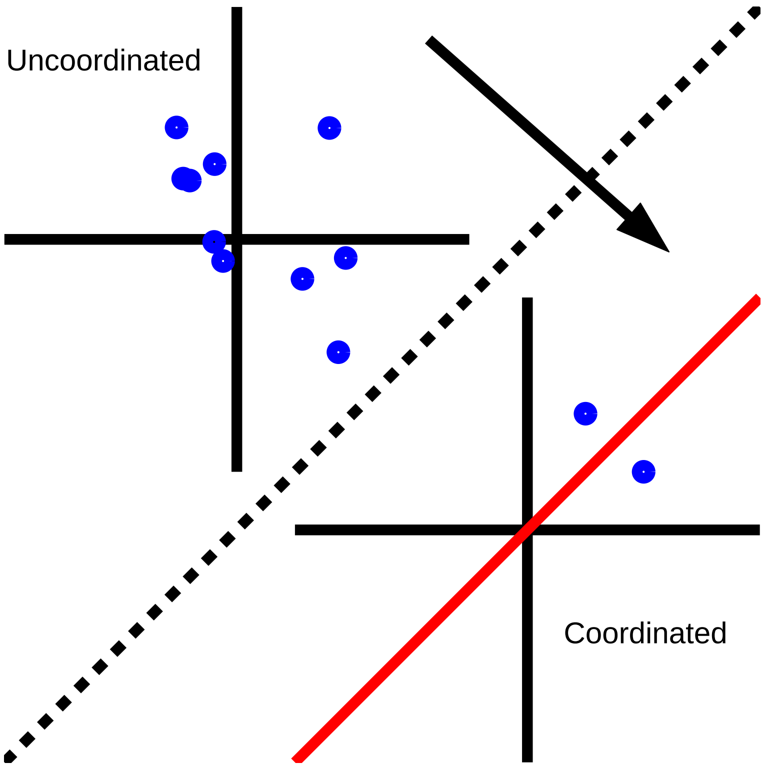


Figure 3.2: All systems defined by pairwise quadratic interaction potentials achieve a coordinated state called "reflection alignment"; every pair of neighbors is related by a reflection across some fixed subspace of their common local state space.

3.1.3 Limitations and Lessons of Potential-based Systems

Despite their long history and popularity, potential-based methods suffer from a number of drawbacks from a design perspective.

We have shown in the previous section that regardless of the dimension of the local state, pairwise quadratic potentials can only accomplish a very special kind of coordination, in which every pair of neighbors is asymptotically related by a reflection. While these systems will find applications in Chapter 4, it is hardly the entire scope of what one might hope from a design theory for distributed coordination. Furthermore, one can certainly imagine distributed interaction mechanisms that are not confined to pairwise interactions; however, these are hardly the only problems present in potential-based interpretations.

Perhaps the most severe problem in potential-based design is that it fundamentally conflates the coordination task to be performed with the specific dynamics that will implement that coordination. The coordination task is encoded in the global sum of the potentials, whereas the individual node dynamics is encoded in the potential functions themselves. Let us examine this once again in the context of $\Phi_L(\cdot)$.

The distributed averaging behavior of $\Phi_L(\cdot)$ is independent of the underlying graph on which it is carried out; indeed, as pointed out before, the averaging behavior is fundamentally a property of $\Phi_L(\cdot)$ and not of any particular $\Phi_L(G)$. However, when one examines this system from a potential-based viewpoint, one is confronted with the rather displeasing fact that the global potential to be minimized does not share this invariance; indeed, one has a different global potential function for each G. Nonetheless, all these potentials achieve the same asymptotic coordinating behavior, so the global potential function is clearly encoding much more than simply the coordination task. From a design perspective, one would like a representational formalism that allows one to formally separate the coordination task from the dynamics which will implement that task.

A secondary drawback of potential-based interpretations is that it is not immediately clear, given a set of potential functions, what coordinating task they will execute (if any); one must carry out an analysis of the dynamics induced by each global potential function. This task is further complicated by the possibility of graph dependence, as we discussed in the previous section.

One should also consider the "converse" of the above statement regarding non-transparency: just as it is not immediately clear what coordinating task a local potential function will accomplish, there is also no clear way to systematically obtain local potential functions that implement a given coordination task. As a consequence, there is a significant *ad hoc* component to any potential-based design.

In the upcoming sections, we will present a novel formalism that will address all of these concerns; however, a bit of motivation based on our understanding of potentials is in order.

As stated at the beginning of this section, the primary power of potentials from a design viewpoint is that they provide a naturally distributed mechanism for synthesizing dynamics (given a choice of a potential function). One simply adds up potentials and obtains a composite gradient flow; let us pause to consider what is actually happening from a geometric perspective. The global potential function is simply the sum of all the local potential functions, and the gradient of this global potential is simply the sum of all the local gradients. Geometrically, each pair of interacting nodes contributes a tangent vector to the overall sum; this situation is depicted in Figure 3.3.

One can also see another significant limitation of potential methods, which only becomes clear in a geometric interpretation: the tangent vectors introduced by each local potential function are each "exact differentials", in the sense that they are each the gradient of some function defined on the global state manifold; there is no particular reason to impose this constraint.

It is also worth noting that the tangent vectors produced by each pair of neighbors are not necessarily descent directions for the global potential; only their sum is guaranteed to be a descent direction. Thus, under this interpretation, the individual nodes do not have any "distributed proof" that the vectors they contribute "make progress" towards the global goal; we point this out to highlight the contrast with the view we will present in the following sections.

In summary, in order to address the drawbacks we have discussed in our treatment of potential

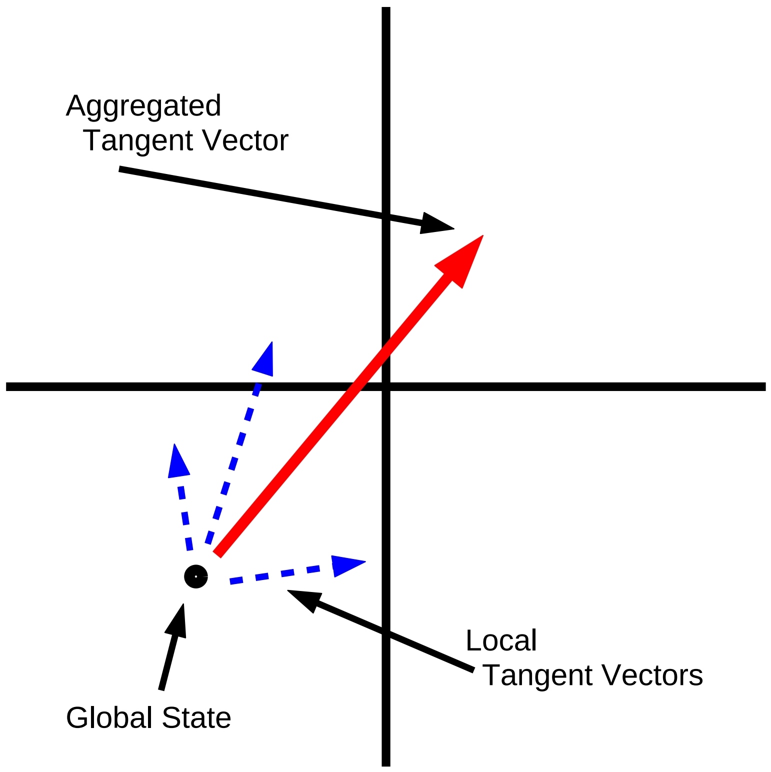


Figure 3.3: Tangent vectors are generated by each local potential, and the overall tangent vector describing the dynamics of the global state is the sum of all these tangent vectors.

systems, our goals for the upcoming discussion are as follows:

- The formalism presented must clearly separate the coordination task to be accomplished from the details of the dynamics that will implement the coordination; insofar as is possible, the specification of coordination tasks should not intrinsically hinge on the underlying network structure.

- The formalism should systematically convert the specification for the coordination task into an appropriate distributed dynamical system; this "compilation" of the dynamics should not depend on particular properties of the network on which it is implemented, and should also allow the freedom to parametrize over different kinds of distributed interaction (i.e. other than purely pairwise computation of tangent vectors).

- The distributed dynamical system should be such that each node produces a *local guarantee* that its actions directly contribute to the progress of the global coordination task, and that its actions respect any global state constraints; in the language of tangent vectors, there must be *distributed proof* that each tangent vector is both a descent direction, and also in some set of allowable directions.

3.2 A Novel Viewpoint: Geometry, Dynamics, and Optimization

In this section we introduce some new abstractions that will be of use in designing dynamics for distributed coordination, but first we will briefly give some perspective on how we will approach the problem.

It is a somewhat standard viewpoint to interpret stable dynamical systems as mappings from initial values to limiting values, and hence to view them as "function evaluators"; these systems take inputs as initial values and provide outputs in the final values, executing the computation through the evolution of the dynamical variables (See Figure 3.4). Indeed, in Chapter 2, our "Stability - Invariance - Equilibrium" architecture provides a systematic way to understand this computational mechanism for a certain class of systems, and suggests the possibility of building a design formalism around these three properties.

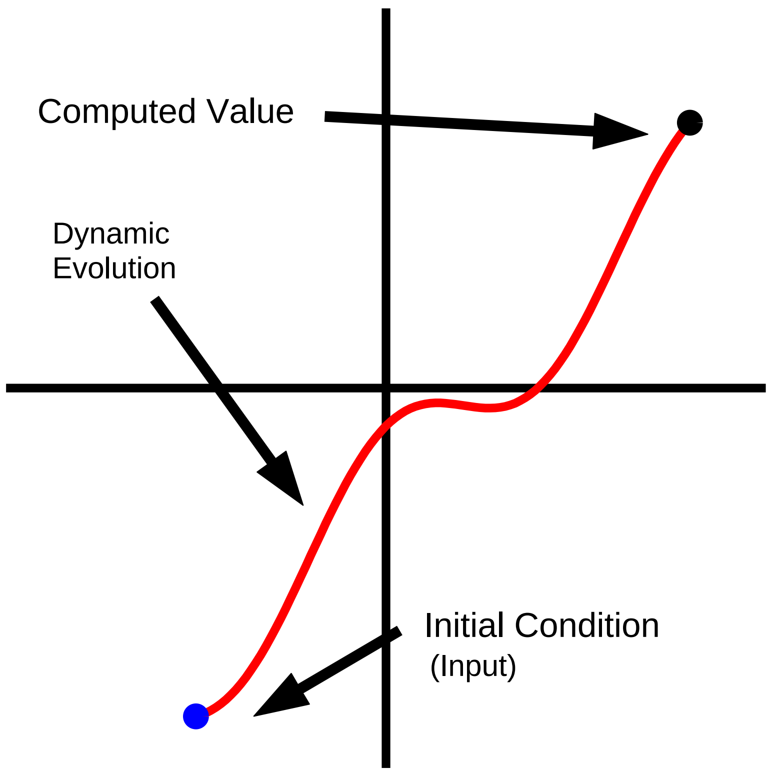


Figure 3.4: A stable dynamical system can execute a computation with inputs specified in the initial values of its states, and outputs defined by the limiting values of the states.

A somewhat less standard, but conceptually similar viewpoint applies to optimization problems.

Consider the following optimization:

$$\begin{aligned} \min_{x,y} \quad & ax^2 + by^2 \\ \text{s.t.} \quad & cx + dy = e. \end{aligned}$$

The optimal solution to this problem (assuming $a, b > 0$ for well-posedness) is :

$$\begin{aligned} x^* &= \frac{ec}{a\left(\frac{c^2}{a} + \frac{d^2}{b}\right)} \\ y^* &= \frac{ed}{b\left(\frac{c^2}{a} + \frac{d^2}{b}\right)}. \end{aligned}$$

Here we plainly see that the optimal solution is parametrized by the values a and b entering the objective function, and the values c, d, and e entering the constraint equation. We thus see that one can also view optimization problems as inducing some function evaluation, mapping parameters from the objective and constraints to the optimal solution; this situation is illustrated in Figure 3.5.

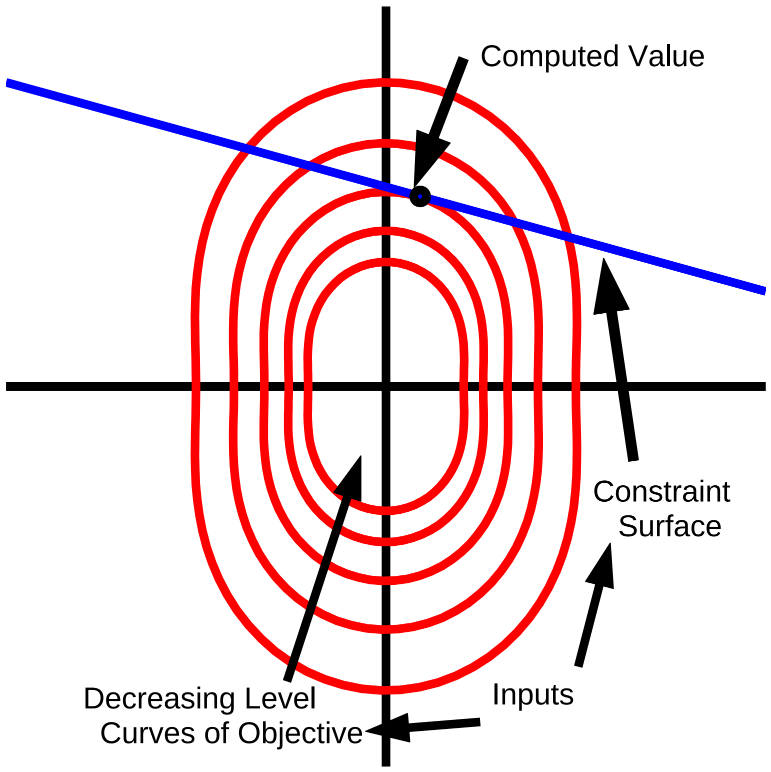


Figure 3.5: An optimization problem with a unique minimum can encode a computation; inputs are specified as parameters of the objective function and the constraint equation, and outputs are specified by the optimal solution.

So, we have presented an interpretation in which both dynamical systems and optimization problems can induce function evaluation; if we can encode our notion of coordination as a function mapping local inputs to a global coordinated state, then these tools provide a natural language for coordination. The view we wish to advance, then, is that of using dynamical systems to *implement* coordination as *specified* by optimization problems; this approach will allow us to achieve the separation of implementation from functionality as desired. The question, of course, is how to combine these disparate mechanisms; an answer is naturally available in our "Stability - Invariance - Equilibrium" architecture, which we will now reiterate in light of the current discussion:

Stability

One can use a dynamical system under whose flow the value of some global objective function decreases monotonically; this allows us to ensure convergence.

Invariance

An invariant quantity in the dynamics allows one to enforce a global constraint on the state variables; this allows us to encode inputs from the constraints of an optimization problem in the initial values and the invariant manifold.

Equilibrium

One can design a dynamical system such that an equilibrium can only occur when the (projected) gradient is zero; this allows one to encode information from the objective function of an optimization problem into the asymptotic behavior of the dynamical system.

To notationally represent the idea of a dynamical system implementing a computation specified by an optimization, we will use the following pattern:

$$\begin{array}{rl} \min & \lim_{t\to\infty} f(\mathbf{x}(t)) \\ \text{s.t.} & g(\mathbf{x}(t)) = g(\mathbf{x}(0)) \end{array}$$

The statements made above merely relate dynamical systems to optimization problems as a mechanism for evaluating a specified function, or achieving a certain coordinated state; there is no connection yet to design of distributed systems. This is evident in our notation; while we have notated the fact that a function is to be optimized, and that the constraint is to be respected as a dynamically invariant manifold, we have not yet added any representation of the fact that this must respect some distributed interconnection structure. The remainder of this section will present some geometric tools that will allow us to represent this structure, and to systematically implement the design architecture proposed above. We will first introduce the notions of *interaction* and *coordination*, which are geometric abstractions for local computational powers and local enforcement of global constraints. We will then discuss how interactions naturally induce a mechanism for distributing "global goals" (represented by a global tangent vector) over local interactions while preserving global constraints. Finally, we will use these ideas to present a simple theory of distributed optimization.

3.2.1 Interactions and Coordinations

In this section we introduce two fundamental geometric notions that will be central to our work; *interactions* and *coordinations*. We will begin with some motivational discussion for our modeling formalism.

In thinking about the dynamics of distributed systems, we need some basic language in which to describe the fact that the dynamic evolution of the global state is induced by the superposition of numerous "local behaviors". Let us consider, for example, the canonical distributed manifold $M(\cdot)$ which assigns to each member of a network a real-valued state $x(i) \in \mathbf{R}$. Now, let us consider a few possibilities for "decision making" in this network:

- An obvious possibility is one in which each node makes decisions entirely independently, and produces a tangent vector modifying only its own state.
- Another extreme scenario is fully centralized control, in which a single decision-making entity selects a tangent vector that updates every node's state.
- Yet another situation is one in which tangent vectors are produced pairwise: each pair of neighbors "confers" and produces a tangent vector that modifies *both* of their states.

Clearly, there are many variations on this type of modeling, and we will shortly present a unified abstraction for representing these differing scenarios.

Before presenting our formal definition for distributed interactions, it is important to note a a subtle but fundamental deviation in the above reasoning from typical models for distributed systems: we have explicitly separated *how choices are made* from *what choices are made*. Indeed, we can capture the essence of a distributed system in the following observations:

- Multiple decision-making units make choices that affect the dynamic evolution of the system; each unit selects a tangent vector, and the overall dynamics is induced by the sum of these tangent vectors.
- Decision-making authority is localized, and no single unit can impose its choices on the entire network; thus, no unit can select vectors from the entire tangent space.

This motivates our basic philosophical viewpoint, from which the remainder of our mathematical abstractions will stem:

A distributed interaction structure is essentially a tangent space constraint.

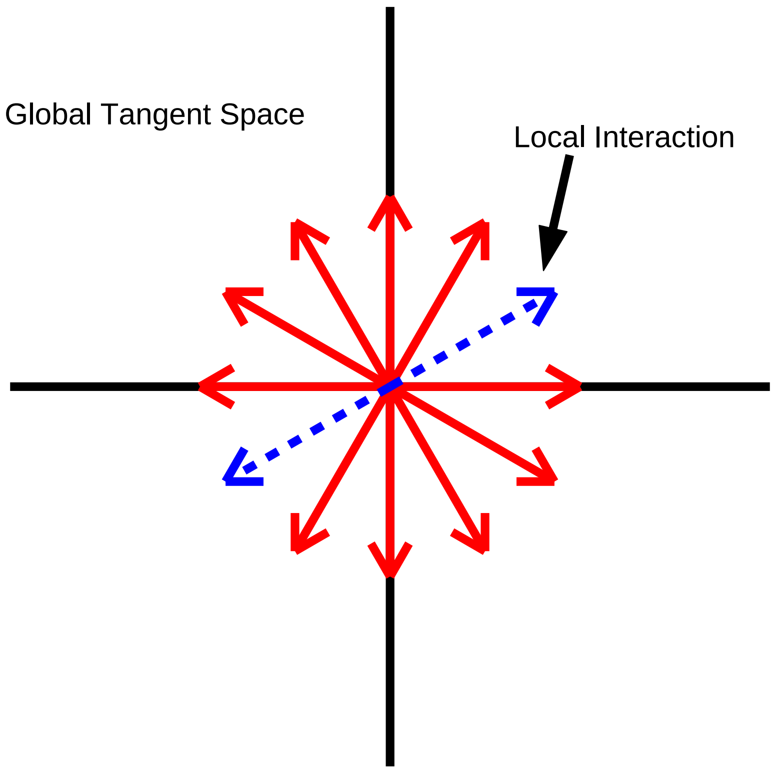


Figure 3.6: Schematic representation of a local interaction; a centralized planner would have access to all the tangent vectors in the tangent space, but a local interaction can only utilize some subspace of tangent vectors.

We will now proceed with our formal development of these ideas.

Definition 16 (Interaction). *Let M be a manifold. By an* interaction *P on M, we mean a mapping associating to each vector field X another vector field $P(X)$, with the property that at each $\mathbf{x}$ in M, P defines an orthogonal projection on the tangent space $T_{\mathbf{x}}M$.*

This definition states that P, at each point $\mathbf{x}$ in M, projects vectors in $T_{\mathbf{x}}M$ orthogonally onto some subspace of $T_{\mathbf{x}}M$; as a consequence, we will identify P with an associated tangent subbundle $P_{\mathbf{x}}$, and will use the notation for the linear operator and the tangent subbundle interchangeably. We will systematically abuse notation and write $P_{\mathbf{x}}X$ for the vector field defined by the action of P on a vector field X.

The identification of an interaction with a tangent subbundle conforms with our previous claim a local interaction should be somehow interpreted as a tangent space constraint; however, we have not yet motivated the technical requirement that led to this identification. To do so, let us imagine some vector field X defined on a manifold M, and let us consider an interaction $P_{\mathbf{x}}$; let us suppose for the moment that X represents some kind of global “command” or “instruction” for the distributed

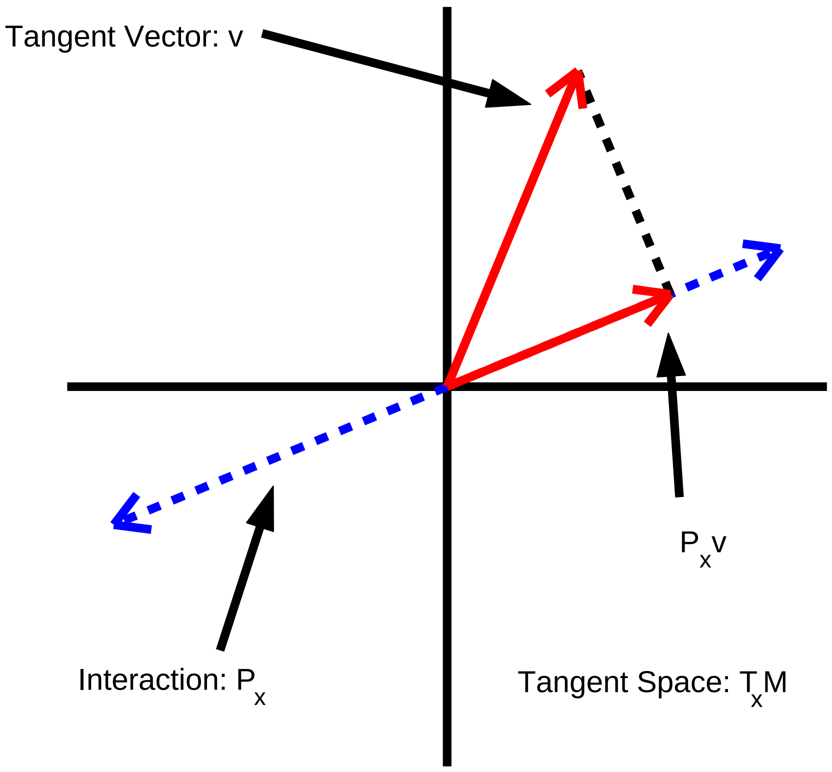


Figure 3.7: An Interaction acting on a vector field; at each point on a manifold, an interaction projects a tangent vector onto some subspace of tangent vectors. The vector field obtained by projecting onto an interaction thus makes a non-negative inner product with the input vector field.

system. Now, consider the following quantity:

$$\langle X(\mathbf{x}), P_{\mathbf{x}} X(\mathbf{x}) \rangle$$

where the notation $X(\mathbf{x})$ indicates the tangent vector defined by the vector field X at the point $\mathbf{x}$. Since $P_{\mathbf{x}} X$ is an orthogonal projection on the tangent space $T_x M$, we then have

$$\langle X(\mathbf{x}), P_{\mathbf{x}} X(\mathbf{x}) \rangle \geq 0$$

for all $\mathbf{x} \in M$. The implication of this inequality is that the new vector field produced by the interaction makes *non-negative progress* toward the global goal encoded in the input vector field. Thus, an interaction is simply a specification for converting a global objective into a locally implementable tangent vector which makes progress towards the global goal.

With our definition of interactions in place, we can now discuss *coordination* of interactions. Intuitively, we would like a coordination to be an object that produces "coordinated interactions"; we would also like this to have a natural distributed structure, in the sense that it respect the local constraints of any given interaction. This motivates the following abstract definition.

Definition 17 (Coordination). *Let M be a manifold. By a* coordination *C on M we mean a mapping associating to each interaction $P_{\mathbf{x}}$ on M another interaction $C(P_{\mathbf{x}})$, with the following properties:*

$$\begin{array}{rcl} C(P_{\mathbf{x}}) & \subset & P_{\mathbf{x}}, \\ P_{\mathbf{x}} \cap C(Q_{\mathbf{x}}) & \subset & C(P_{\mathbf{x}}), \\ P_{\mathbf{x}} \subset Q_{\mathbf{x}} & \Rightarrow & C(P_{\mathbf{x}}) \subset C(Q_{\mathbf{x}}), \end{array}$$

where the $\subset$ notation is to be understood pointwise at each $\mathbf{x}$ in M.

One should think of the action of a coordination C on an interaction $P_{\mathbf{x}}$ as "organizing" the interaction so as to respect some constraint. The first requirement above states that coordinating an interaction does not produce any vectors that were not available to the uncoordinated interaction. The second requirement states that for any two interactions $P_{\mathbf{x}}$ and $Q_{\mathbf{x}}$, any direction in $P_{\mathbf{x}}$ that is a coordinated direction for $Q_{\mathbf{x}}$ is also a coordinated direction for $P_{\mathbf{x}}$; intuitively, this means that the property of being "coordinated" is a global one, and not specific to each interaction. Finally, we require that coordinating two interactions maintain their relative standing in the containment relationship; if $P_{\mathbf{x}}$ is "weaker than" $Q_{\mathbf{x}}$, then $C(P_{\mathbf{x}})$ should be "weaker than" $C(Q_{\mathbf{x}})$.

As with interactions, we will identify coordinations with a tangent subbundle; however, the connection is not as immediately clear as in the previous case. The following lemma will illustrate this relationship.

Lemma 2 (Coordination as Subbundle). *Let M be a manifold, and let C be a coordination on M. Then, C is uniquely specified by its action on the tangent bundle, $C(TM)$.*

Proof. It is clear that given a coordination C, the tangent subbundle $C(TM)$ is uniquely specified. To show the converse, let C be a coordination, and let $P_{\mathbf{x}}$ be an interaction on M. Since $P_{\mathbf{x}} \subset TM$, we have that $C(P_{\mathbf{x}}) \subset C(TM)$. We also have $C(P_{\mathbf{x}}) \subset P_{\mathbf{x}}$, and so combining these we obtain:

$$C(P_{\mathbf{x}}) \subset P_{\mathbf{x}} \cap C(TM).$$

Now, applying the third requirement for a coordination to the above, we find

$$C(P_{\mathbf{x}}) \subset P_{\mathbf{x}} \cap C(TM) \subset C(P_{\mathbf{x}}),$$

which implies

$$C(P_{\mathbf{x}}) = P_{\mathbf{x}} \cap C(TM).$$

$\square$

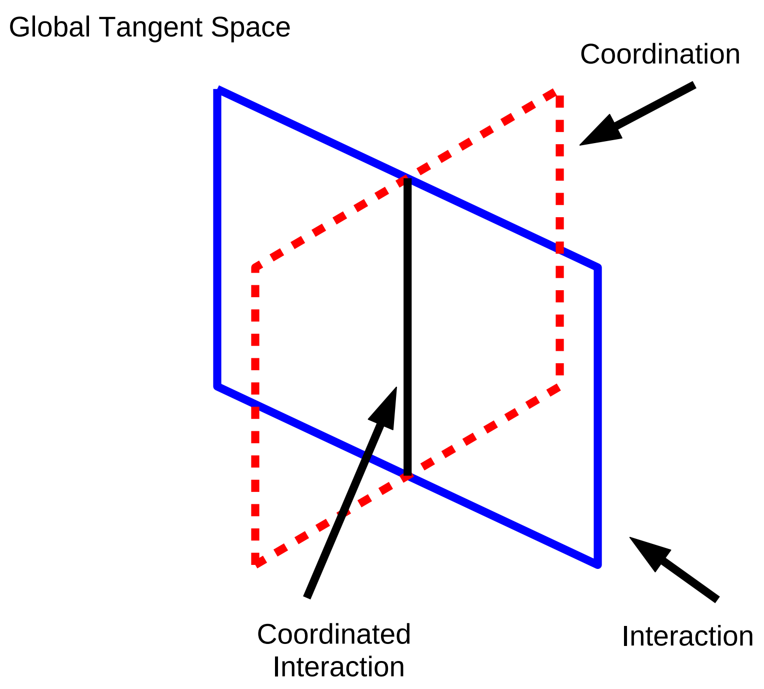


Figure 3.8: A Coordination acting on an Interaction; a Coordination defines a global constraint on allowable tangent directions, and it maps input interactions to output interactions that respect the global constraint.

Corollary 1. *Any coordination C is idempotent, i.e. $C(C(P_{\mathbf{x}})) = C(P_{\mathbf{x}})$.*

The preceding corollary shows another intuitive property of coordination: once an interaction $P_{\mathbf{x}}$ has been coordinated by C, any further application of C has no additional effect.

So, we see that a coordination can be identified with a particular interaction $C(TM)$, which is itself a tangent subbundle. We will thus treat coordinations as subbundles, and will hereafter employ the notation $C_{\mathbf{x}}$ to maintain consistency with our previous notation.

Below, we collect some notational conventions we will employ in the rest of our discussion.

- Both interactions and coordinations have been identified with tangent subbundles; both will be notated with the $P_{\mathbf{x}}$ (or $C_{\mathbf{x}}$) notation for conceptual consistency, and to distinguish them from general linear operators.
- When applying an interaction to a vector field X, we will always use the notation $P_{\mathbf{x}}X$; when applying a coordination to an interaction, we will always use the notation $C_{\mathbf{x}}(P_{\mathbf{x}})$.
- We will employ the notions of *dimension* and *codimension* for interactions and coordinations, by which we mean the dimension (codimension) of the associated tangent subbundles; we will write these as $\dim P_{\mathbf{x}}$ or $\operatorname{codim} C_{\mathbf{x}}$.

So, we now have a formal definition for coordination; we associate to every interaction (set of available directions) some other interaction which is "coordinated". This operation was shown above to be equivalent to intersecting the directions available in the interaction with some global set of coordinated directions, which again conforms with our original philosophical claim that the essence of a distributed system is in tangent space constraints.

One central question we can now begin to address is that of feasibility. Broadly, there is a general question to be answered pertaining to what kinds of coordinated behaviors can be achieved for a particular distributed system; this is beyond our scope for the moment, but we will return to it in the following section. Presently, we would like to know whether a particular interaction can be coordinated in a particular way; this motivates the following definition:

Definition 18 (Feasible Coordination). *Suppose M is a manifold, $P_{\mathbf{x}}$ is an interaction on M, and $C_{\mathbf{x}}$ is a coordination on M; we will say that $C_{\mathbf{x}}$ is* feasible *for $P_{\mathbf{x}}$ if*

$$\dim C_{\mathbf{x}}(P_{\mathbf{x}}) > 0,$$

otherwise, we call $C_{\mathbf{x}}$ infeasible *for $P_{\mathbf{x}}$.*

Thus, we call a coordination feasible for a given interaction if, when applied to the interaction, it yields a non-zero set of available directions. Below, we give a dimensional characterization of feasibility, but first we need a definition to specify that an interaction is not already "partially coordinated". This is merely a technical requirement, and is not an issue "in practice"; all the interactions and coordinations we will discuss in the remainder of the work will satisfy this condition.

Definition 19 (Independence). *We will say that an interaction $P_{\mathbf{x}}$ and a coordination $C_{\mathbf{x}}$ are* independent *if*

$$\dim C_{\mathbf{x}}(P_{\mathbf{x}}) = \dim P_{\mathbf{x}} - \operatorname{codim} C_{\mathbf{x}}$$

As an example where independence breaks down, consider $C_{\mathbf{x}}(P_{\mathbf{x}})$. Clearly, the above equation does not hold, as $C_{\mathbf{x}}(C_{\mathbf{x}}(P_{\mathbf{x}})) = C_{\mathbf{x}}(P_{\mathbf{x}})$. This captures the essence of the requirement; intuitively, if a coordination $C_{\mathbf{x}}$ is viewed as a collection of global constraints, then it is independent of an interaction $P_{\mathbf{x}}$ if and only if the interaction contains vectors which violate each of the constraints.

Proposition 3 (Interaction/Coordination Inequality). *Suppose M is a manifold, $P_{\mathbf{x}}$ is an interaction on M, $C_{\mathbf{x}}$ is a coordination on M that is feasible for and independent of $P_{\mathbf{x}}$. Then,*

$$\dim P_{\mathbf{x}} > \operatorname{codim} C_{\mathbf{x}}.$$

Proof. From the hypotheses, we have

$$\dim C_{\mathbf{x}}(P_{\mathbf{x}}) = \dim P_{\mathbf{x}} - \operatorname{codim} C_{\mathbf{x}} > 0,$$

which immediately proves the desired result. $\square$

Corollary 2. *If $C_{\mathbf{x}}$ and $P_{\mathbf{x}}$ are independent, and* $\operatorname{codim} C_{\mathbf{x}} \geq \dim P_{\mathbf{x}}$ *then $C_{\mathbf{x}}$ is not feasible for $P_{\mathbf{x}}$.*

Corollary 3. *For arbitrary $C_{\mathbf{x}}$ and $P_{\mathbf{x}}$, if* $\dim P_{\mathbf{x}} > \operatorname{codim} C_{\mathbf{x}}$*, then $C_{\mathbf{x}}$ is feasible for $P_{\mathbf{x}}$.*

This proposition, though simple, gives a fundamental relationship between the "power" of an interaction (measured by its dimension) and the "complexity" of some desired coordination (measured by its codimension). The linear algebraic intuition which applies here is the following:

> Two subspaces of dimensions k and m in a vector space of dimension n generically have a zero-dimensional intersection if $k + m \leq n$.

This statement only applies to "generic" intersections, and so in the proposition above we had to require independence to make a categorical statement.

Having introduced our formal abstractions for interactions and coordinations, let us give a few concrete examples to make these ideas clear. In particular, we wish to reinforce the idea that the "power" of an interaction is related to its dimension, and that the "complexity" of a coordination is related to its codimension; let us begin by discussing interactions.

We now revisit the three examples for "network decision making" we presented at the beginning of this section, and cast them as interactions. For the sake of concreteness, let us suppose again that we are dealing with a network in which each member has a real state $x(i) \in \mathbf{R}$, and so the global state vector $\mathbf{x} \in \mathbf{R}^n$ is the concatenated vector, $\mathbf{x}_i = x(i)$.

- Purely local control at a node i can be expressed by the interaction $\mathbf{e}_i \otimes \mathbf{e}_i$, which is just the projection onto the i^{th} component.
- Global control can be cast as an interaction, which is simply the entire tangent bundle TM; every tangent vector in every tangent space is available to a global controller.
- Pairwise control can be expressed by the interaction $\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j$ for some pair of neighbors i and j. This interaction includes any vector which modifies only the states $x(i)$ or $x(j)$.

Now, let us consider a prototypical coordination, one which preserves the sum of the components of the global state vector. Denoting this coordination $S_{\mathbf{x}}$, we see that the associated tangent subbundle (in this case just a subspace of $\mathbf{R}^n$) is the collection of all tangent vectors whose components sum to zero. We will consider the effect of this coordination on each of the interactions presented above.

- This coordination is infeasible for the purely local interaction $\mathbf{e}_i \otimes \mathbf{e}_i$, as every tangent vector in its range violates the constraint; thus $S_{\mathbf{x}}(\mathbf{e}_i \otimes \mathbf{e}_i) = \{0\}$.

- Applying $S_{\mathbf{x}}$ to the global controller, TM, we simply obtain the set of tangent vectors whose components sum to zero. We can write the coordinated interaction as:

$$S_{\mathbf{x}}(TM) = I - \frac{\mathbf{1}_n \mathbf{1}_n^T}{\mathbf{1}_n^T \mathbf{1}_n}.$$

- $S_{\mathbf{x}}$ is feasible for the pairwise interaction $\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j$, and applying it yields:

$$S_{\mathbf{x}}(\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j) = \frac{1}{2}\left(\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j - \mathbf{e}_i \otimes \mathbf{e}_j - \mathbf{e}_j \otimes \mathbf{e}_i\right).$$

  To make this more transparent, we will consider this in the basis $\{\mathbf{e}_i, \mathbf{e}_j\}$, where the associated matrix representation is:

$$\frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}.$$

  This matrix should look very familiar given our previous discussion of Laplacian matrices in Chapter 2, and our presentation of the distributed averaging system $\Phi_L(\cdot)$ as a pairwise potential-based system in Section 3.1. We will return to this connection in Section 3.2.3.

Here we begin to see the relationship between interactions and coordinations, as well as hints of our previous assertion that the "power" of an interaction is related to its dimension, whereas the "complexity" of a coordination is related to its codimension. The "weakest" interaction we discussed, purely local control, did not contain any vectors satisfying the constraints of the coordination $S_{\mathbf{x}}$. The coordination was feasible for the pairwise interaction, and resulted in a one-dimensional coordinated interaction. Had we added an additional linear constraint on the states, then the resulting coordination would have had codimension two, and would have been infeasible for the pairwise interaction as well. Applying the coordination to the "global controller" resulted in the largest possible set of allowable directions, the coordination subbundle itself.

Having established properties of individual interactions, we would now like to answer questions about multiple interactions; this will be carried out in the following section.

3.2.2 Distributing Vector Fields and Distributed Interactions

In this section we will consider properties associated with collections of interactions; this will form the center of our model for general distributed systems. We will introduce the notion of *distributing* a vector field, and relate properties of the distributed vector field to properties of sets of interactions; we begin with some definitions.

Definition 20 (Interaction Set). *Let M be a manifold. An* Interaction Set *$\mathcal{I}$ on M is a set of interactions on M.*

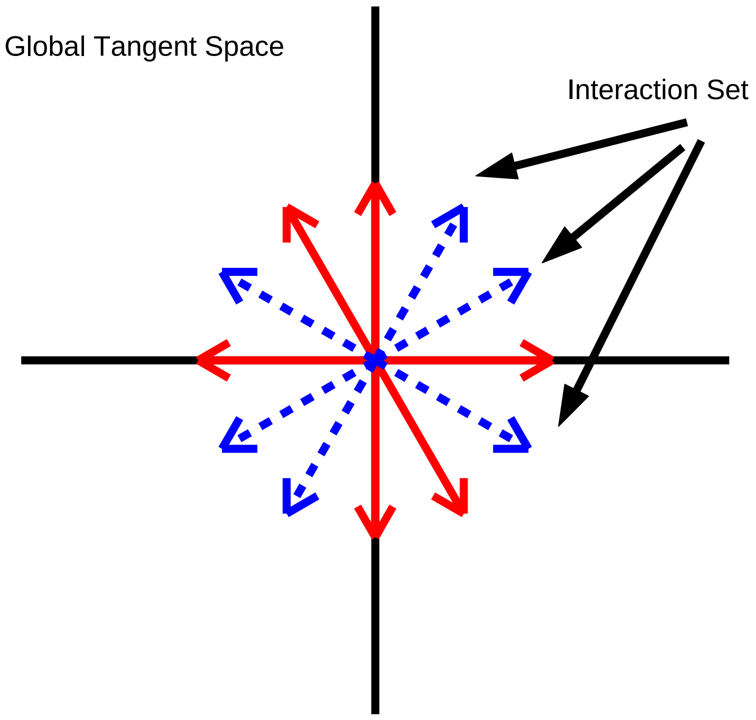


Figure 3.9: An Interaction Set is a collection of interactions; each defines some subspace of tangent vectors associated with a particular local interaction.

Definition 21 (Distributing a Vector Field). *Suppose M is a manifold, $\mathcal{I}$ is an interaction set on M, and V is a vector field on M. We define $V_{\mathcal{I}}$, as follows:*

$$V_{\mathcal{I}} = \sum_{P_{\mathbf{x}} \in \mathcal{I}} P_{\mathbf{x}} V.$$

Lemma 3 (Properties of Distribution). *Suppose M is a manifold and $\mathcal{I}$ is an interaction set on M. Then, we have the following properties:*

1. *At each $\mathbf{x}$ in M, there exists a linear operator on the tangent space $T_{\mathbf{x}}M$, call it $\mathcal{D}_{\mathcal{I}}(\mathbf{x})$, such that for every vector field V on M*

$$V_{\mathcal{I}}(\mathbf{x}) = \mathcal{D}_{\mathcal{I}}(\mathbf{x}) V(\mathbf{x}).$$

2. *At each $\mathbf{x}$, $\mathcal{D}_{\mathcal{I}}(\mathbf{x})$ is self-adjoint and positive-semidefinite.*

3. *If for some $\mathbf{x}$ and some V, $V_{\mathcal{I}}(\mathbf{x}) = 0$, then we have for all $P_{\mathbf{x}}$ in $\mathcal{I}$*

$$P_{\mathbf{x}} V(\mathbf{x}) = 0$$

Proof. From the definition, $V_{\mathcal{I}} = \sum_{P_{\mathbf{x}} \in \mathcal{I}} P_{\mathbf{x}} V$. At each point $\mathbf{x}$, each interaction $P_{\mathbf{x}}$ defines an orthogonal projection onto some subspace of $T_{\mathbf{x}}M$. Each such term is clearly a positive semidefinite and self-adjoint linear operator, and thus the sum is also positive-semidefinite and self-adjoint. To prove the final claim, consider the following quadratic form:

$$\langle V(\mathbf{x}), V_{\mathcal{I}}(\mathbf{x})\rangle = \sum_{P_{\mathbf{x}} \in \mathcal{I}} \langle V(\mathbf{x}), P_{\mathbf{x}} V(\mathbf{x})\rangle = 0.$$

Each term in the sum must be non-negative because each $P_{\mathbf{x}}$ defines an orthogonal projection onto some subspace; thus, each term must be zero. $\square$

Note that although $V_{\mathcal{I}}(\mathbf{x})$ must make a non-negative inner product with $V(\mathbf{x})$ at each $\mathbf{x}$, $\mathcal{D}_{\mathcal{I}}$ does *not* define an interaction, because it is not (in general) idempotent.

Having obtained an algebraic representation of an interaction set, it is natural to examine algebraic properties of these objects, and to see how these properties relate to our interpretation of interaction sets as models of distributed systems. This motivates the following definition:

Definition 22 (Rank of an Interaction Set). *Let M be a manifold, and let $\mathcal{I}$ be an interaction set on that manifold. We will say that the* rank *of $\mathcal{I}$ at a point $\mathbf{x}$ in M is*

$$\operatorname{rank}(\mathcal{I})_{\mathbf{x}} = \operatorname{rank}(\mathcal{D}_{\mathcal{I}}(\mathbf{x})).$$

The rank of an interaction set naturally generalizes the notion of dimension for an interaction. Just as we can coordinate an interaction, we can coordinate an interaction set in the obvious way:

Definition 23 (Coordinating an Interaction Set). *Let M be a manifold, $C_{\mathbf{x}}$ be a coordination on M, and $\mathcal{I}$ be an interaction set on M. We define*

$$C_{\mathbf{x}}(\mathcal{I}) = \{C_{\mathbf{x}}(P_{\mathbf{x}}) \mid P_{\mathbf{x}} \in \mathcal{I}\}.$$

It is natural to ask what level of coordination is feasible for an interaction set; this property will in fact prove to be much more informative than the rank of the interaction set. We thus introduce the notion of capacity:

Definition 24 (Coordination Capacity). *Let M be a manifold, and let $\mathcal{I}$ be an interaction set on M. We define the* coordination capacity *of $\mathcal{I}$ at $\mathbf{x}$, denoted* $\operatorname{cap}(\mathcal{I})_{\mathbf{x}}$ *to be*

$$\begin{array}{rl} \displaystyle\max_{k \in \mathbf{Z}^+} & k \\ s.t. & \operatorname{codim} C_{\mathbf{x}} \leq k \Rightarrow \\ & \qquad \operatorname{rank}\left(C_{\mathbf{x}}(\mathcal{I})\right)_{\mathbf{x}} > 0 \end{array}$$

This definition is quite dense, but provides a very powerful abstraction for analyzing distributed systems; to clarify the situation let us state it informally:

> *We intuitively think that a distributed system with communication capabilities has some power to coordinate its action across the various local interactions that comprise it. Each time a coordination is implemented, global constraints are enforced, and the system may lose some of its power for additional coordination. The coordination capacity is simply the maximum "complexity" of coordination that the system can support and still have some remaining power for additional coordination.*

Clearly, we can obtain some trivial bounds on $\mathrm{cap}(\mathcal{I})_{\mathbf{x}}$:

$$0 \leq \mathrm{cap}(C_{\mathbf{x}}(\mathcal{I}))_{\mathbf{x}} \leq \mathrm{cap}(\mathcal{I})_{\mathbf{x}} \leq \mathrm{rank}(\mathcal{I})_{\mathbf{x}}.$$

We will shortly demonstrate capacities and the distribution operator for a few systems, but before doing so, we will present one final concept: the notion of a distributed interaction.

Definition 25 (Distributed Interaction). *A* distributed interaction *$\mathcal{P}$ is a mapping from a set of graphs Γ to a set of pairs of the form $(M(G), \mathcal{I}(G))$, where $M(\cdot)$ is a distributed manifold, and $\mathcal{I}(G)$ is an interaction set on $M(G)$ for each G. We will say that $\mathcal{P}(\cdot)$ is a distributed interaction on the distributed manifold $M(\cdot)$.*

We will use our standard pattern of $\mathcal{P}(\cdot)$ to refer to the distributed interaction, and $\mathcal{P}(G)$ to refer to the image of a particular graph. When there is no danger of confusion, we will abuse notation and make statements like "$P_{\mathbf{x}} \in \mathcal{P}(G)$", when we really mean that the interaction $P_{\mathbf{x}}$ is in the associated interaction set $\mathcal{I}(G)$ specified by $\mathcal{P}(G)$.

A distributed interaction is a fairly complex object, but it provides a general model for the components of any distributed system. In particular, it specifies for each graph G:

- A local state space for each $i \in G$.
- A collection of available tangent vectors arising from local interactions.

We would like to examine the coordination capacity of various distributed interactions $\mathcal{P}(\cdot)$ on various graphs; in particular, we are interested in establishing capacity properties that are independent of the underlying graph structure. We will now proceed to give some examples of distributed interactions on the canonical distributed manifold $M_{\mathbf{R}}(\cdot)$; in the interest of legibility, we will commit the usual abuse of identifying all the tangent spaces of $\mathbf{R}^n$ with a single copy of $\mathbf{R}^n$, and will thus be able to use linear algebraic notation.

$\mathcal{P}_V(\cdot)$: This is the purely local interaction, defined by the interaction set

$$\mathcal{I}(G) = \{\mathbf{e}_i \otimes \mathbf{e}_i \mid i \in G\}.$$

For each G, this interaction set is a collection of $|V|$ linearly independent orthogonal projections, and so has rank $|V|$. The distribution operator is constant as a function of the global state $\mathbf{x} \in \mathbf{R}^n$, and is given by

$$\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = I_{n \times n}.$$

However, despite the fact that the distribution operator has full rank, this is clearly a very weak interaction. In particular,

$$\text{cap}(\mathcal{I}(G))_{\mathbf{x}} = 0$$

for every G. To see this, consider the coordination $S_{\mathbf{x}}$ defined by the codimension-1 subspace $\mathbf{v}^T \mathbf{1}_n = 0$; none of the interactions in $\mathcal{I}$ contain non-zero vectors in this subspace.

$\mathcal{P}_E(\cdot)$: This is the distributed interaction associated with pairwise interactions across edges in the graph G, and is defined by the interaction set

$$\mathcal{I}(G) = \{\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j \mid (i,j) \in G\}.$$

The associated distribution operator is the degree matrix of the graph,

$$\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = D(G)$$

and this is clearly full rank for any graph in which every node has at least one neighbor. This distributed interaction has non-zero coordination capacity, specifically:

$$\text{cap}(\mathcal{I}(G))_{\mathbf{x}} = 1.$$

Let us again consider the coordination $S_{\mathbf{x}}$ specified by $\mathbf{v}^T \mathbf{1}_n = 0$; the coordinated interaction set is

$$S_{\mathbf{x}}(\mathcal{I}(G)) = \left\{ \frac{1}{2}\left(\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j - \mathbf{e}_i \otimes \mathbf{e}_j - \mathbf{e}_j \otimes \mathbf{e}_i\right) \;\middle|\; (i,j) \in G \right\}.$$

and the associated distribution operator is just a scaled Laplacian matrix

$$\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = \frac{1}{2} L(G).$$

This interaction set has rank $n-1$ whenever G is connected, and coordination capacity

$$\operatorname{cap}(S_{\mathbf{x}}(\mathcal{I}(G)))_{\mathbf{x}} = 0.$$

To see that the coordinated interaction set has zero capacity, one need only consider a codimension-1 coordination of the form $\mathbf{v}^T\mathbf{b} = 0$, where all of the elements of $\mathbf{b}$ are distinct; none of the coordinated interactions contain vectors satisfying this constraint.

$\mathcal{P}_C(\cdot)$: This distributed interaction models the action of a global coordinator, and is defined by the interaction set

$$\mathcal{I}(G) = \{I_{n\times n}\};$$

the associated distribution operator is clearly just the identity operator

$$\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = I_{n\times n}.$$

The coordination capacity of the global controller is

$$\operatorname{cap}(\mathcal{I}(G))_{\mathbf{x}} = n-1,$$

one less than the dimension of the global state space; the global controller has access to all possible tangent vectors and so can implement any constraint that does not eliminate all possible tangent directions.

Applying the same coordination $S_{\mathbf{x}}$ specified above, we obtain:

$$S_{\mathbf{x}}(\mathcal{I}(G)) = \left\{ I_{n\times n} - \frac{\mathbf{1}_n\mathbf{1}_n^T}{\mathbf{1}_n^T\mathbf{1}_n} \right\}$$

and the distribution operator is just the orthogonal projection onto the subspace satisfying the constraint $\mathbf{v}^T\mathbf{1}_n = 0$. The capacity of the coordinated interaction is now $n-2$.

$P_N(\cdot)$: This distributed interaction models situations in which every node coordinates all actions inside its neighborhood; intuitively, each node produces a tangent vector for itself and for its neighbors which respects the global constraint. This is specified by:

$$\mathcal{I}(G) = \left\{ \mathbf{e}_i \otimes \mathbf{e}_i + \sum_{j \in N_i} \mathbf{e}_j \otimes \mathbf{e}_j \;\middle|\; i \in G \right\}.$$

The associated distribution operator is again related to the degree matrix,

$$\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = I_{n \times n} + D(G),$$

which is clearly full rank, irrespective of the connectedness of G.

The coordination capacity of this distributed interaction is bounded below by 1, and increases with increasing connection density; if G is the complete graph, then this in fact reproduces the global controller, and its coordination capacity is $n-1$. The coordinated interaction set for $S_{\mathbf{x}}$ can be calculated to be

$$S_{\mathbf{x}}(\mathcal{I}(G)) = \left\{ \frac{1}{1+d_i} \left[\mathbf{e}_i \otimes \mathbf{e}_i + \sum_{j \in N_i} (\mathbf{e}_j \otimes \mathbf{e}_j - \mathbf{e}_j \otimes \mathbf{e}_i - \mathbf{e}_i \otimes \mathbf{e}_j) \right] \;\middle|\; i \in G \right\},$$

where d_i is the degree of node i.

The expression for the coordinated distribution operator is fairly complex and we will omit it; we will say, however, that it has rank $n-1$ if and only if G is connected, and that the coordination capacity of the coordinated interaction satisfies

$$\operatorname{cap}(S_{\mathbf{x}}(\mathcal{I}(G)))_{\mathbf{x}} \geq \min_i d_i - 1.$$

Again, for a complete G, this distributed interaction reproduces centralized control, and has coordination capacity $n-2$.

We thus see that although many patterns for distributed interaction can have similar distribution operators and rank, they can have vastly differing coordination capacities. These four possibilities are illustrated in Figure 3.10, and the following table summarizes the above results:

Type of Interaction	Symbol	Distribution Operator	Coordination Capacity
Local Control	$\mathcal{P}_V(\cdot)$	$I_{n\times n}$	0
Pairwise Interaction	$\mathcal{P}_E(\cdot)$	$D(G)$	1
Central Control	$\mathcal{P}_C(\cdot)$	$I_{n\times n}$	n
Neighborhood Interaction	$\mathcal{P}_N(\cdot)$	$D(G) + I_{n\times n}$	$\geq \min_{i\in G} d_i$

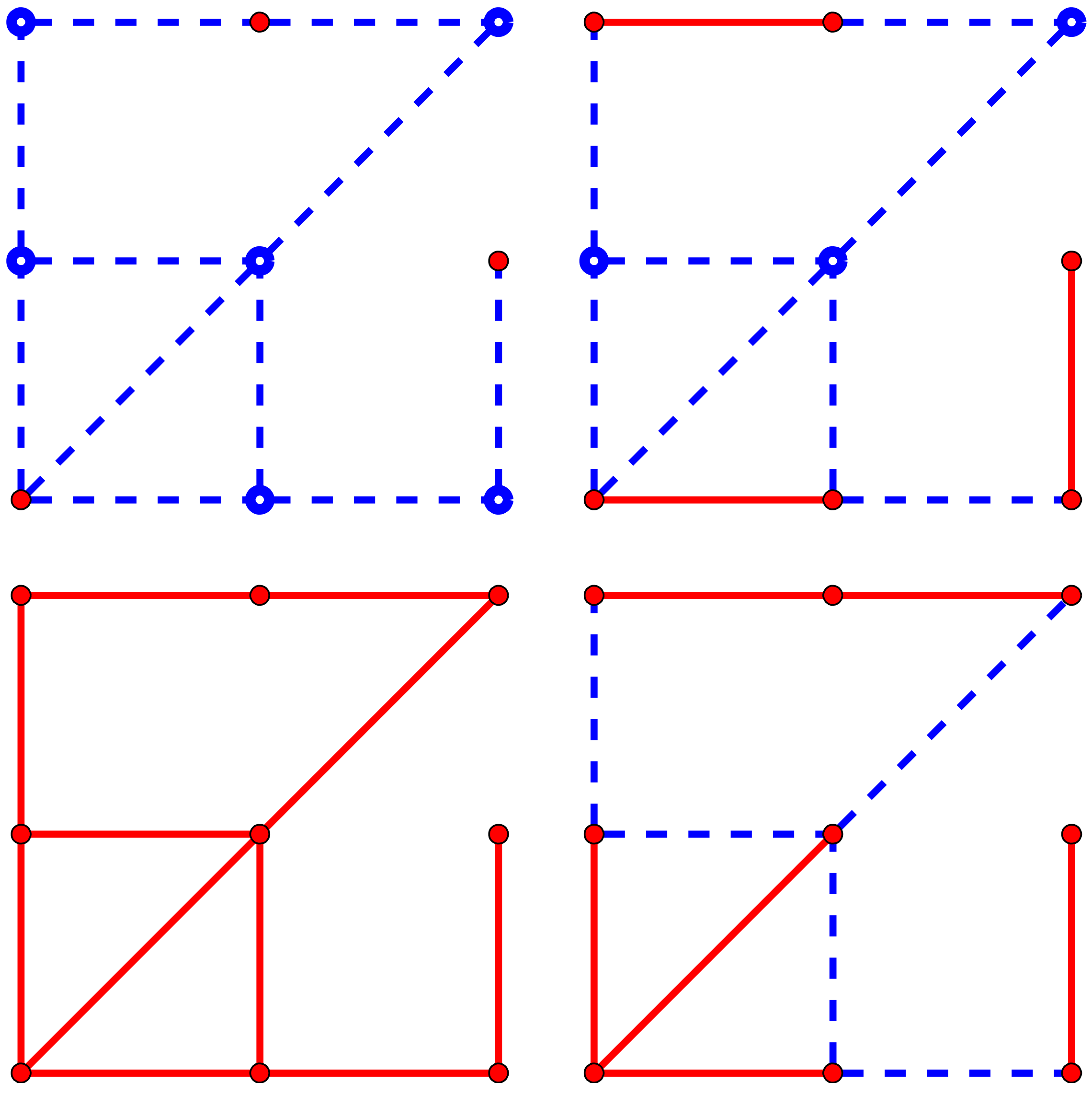


Figure 3.10: Four examples of distributed interactions for a graph: clockwise from the top left, these are the local, pairwise, "neighborhood", and global distributed interactions (not all interactions of each type are shown in the figures); groups of solid-filled nodes linked by solid edges indicate individual interactions in the overall interaction set. Each of these distributed interactions has an associated *distribution operator* $\mathcal{D}_{\mathcal{I}}$ and *coordination capacity* $\text{cap}(\mathcal{I})$; they are organized clockwise in ascending coordination capacity (0, 1, 5, and 8 respectively).

We thus see that a vector can be distributed in a number of ways, for a variety of different interaction structures. Each interaction pattern defines an associated distribution operator, which in turn produces a "distributed version" of an input vector field specifying a global objective. This vector field is highly structured; in particular it is a sum of vector fields, each with the following two very special properties:

- *Distributed Proof of Improvement*

  Each vector field is locally defined by orthogonal projection onto some locally available tangent vectors; thus, it must make a non-negative inner-product with the input vector field. The members participating in the interaction that contributes this tangent vector can collectively prove that this tangent vector makes non-negative progress.

- *Distributed Proof that Constraints are Respected*

  Each vector field *a priori* satisfies any constraints imposed by a coordination, and the members of the local interaction contributing the tangent vector can collectively verify this.

These points are essential to our overall view on distributed coordination. Many general questions (such as whether certain optimization problems can be solved in some distributed fashion) become very clear when one asks whether there can exist distributed proof of the desired property; the additional structure imposed by requiring distributed proof is frequently sufficient to provide meaningful answers. Our notion of coordination capacity will provide a useful tool for answering some such questions.

Having identified the above properties as invariant across distributed interactions, we have also seen two properties that vary significantly for different interaction structures: the rank of the distribution operator, and the manner in which the vector is "skewed" in the distribution process. We will make use of the former property in the following section, where we will use it to characterize *distributed extrema*; we will revisit the latter idea in Section 3.3, when we discuss *efficiency.*

3.2.3 Distributed Optimization

We will now utilize the tools of the preceding section to present a simple model for distributed optimization; once these concepts are in place, we will be able to return to our overall goal of providing a general mechanism for distributed coordination.

We should make it clear from the beginning that we mean something quite special by "distributed optimization"; while the commonplace usage of this phrase tends to center around producing algorithms for solving optimization problems that have some kind of distributed architecture, we mean to propose a *model* for optimization in a distributed setting. Specifically, we want to provide a meaningful characterization of "distributed optimality" that allows us to make statements about the power of locally interacting agents to optimize some global function.

The main idea we wish to convey in this section is the following:

> Smooth equality-constrained optimization characterizes local extrema in terms of tangent-space properties. At a candidate optimum, the gradient of the objective function must be orthogonal to all the allowable tangent directions; if it is not, then there is *local* (in the sense of calculus) proof that the candidate point is not optimal.
>
> Distributed systems naturally have a special class of tangent space constraints encoded by their local interaction structure; it is thus natural to ask when there is *distributed proof* that a candidate configuration of a distributed system is suboptimal.

With this idea in mind, we make the following definition:

Definition 26 (Distributed Extremum). *Let $f(\mathbf{x})$ be a smooth real-valued function on $\mathbf{R}^n$ and let $\mathcal{I}$ be an interaction set on $\mathbf{R}^n$. We will say that a point $\mathbf{x}^*$ is a* distributed extrememum *of f relative to $\mathcal{I}$ if for every interaction $P_\mathbf{x} \in \mathcal{I}$, and every tangent vector $\mathbf{v}$ in $P_\mathbf{x}$ at $\mathbf{x}$,*

$$\left.\frac{d}{dt}\right|_{t=0} f(\mathbf{x}^* + t\mathbf{v}) = 0.$$

Definition 27 (Distributed Gradient). *Let $f(\mathbf{x})$ be a smooth real-valued function on $\mathbf{R}^n$ and let $\mathcal{I}$ be an interaction set on $\mathbf{R}^n$. We define the* distributed gradient *of f relative to $\mathcal{I}$ at x as follows:*

$$\nabla_\mathcal{I} f(\mathbf{x}) = \mathcal{D}_\mathcal{I}(\mathbf{x}) \nabla f(\mathbf{x}).$$

Lemma 4 (Differential Characterization of Distributed Extrema). *Let $f(\mathbf{x})$ be a smooth real-valued function on $\mathbf{R}^n$ and let $\mathcal{I}$ be an interaction set on $\mathbf{R}^n$. Then, a point $\mathbf{x}^*$ is a distributed extremum relative to $\mathcal{I}$ if and only if:*

$$\nabla_\mathcal{I} f(\mathbf{x}) = 0.$$

Proof. $\Rightarrow$ Suppose $\nabla_\mathcal{I} f(\mathbf{x}^*) = 0$. Then, from Lemma 3 on the properties of the distribution operator, we have that $\nabla f(\mathbf{x}^*)$ is annihilated by each $P_\mathbf{x}$ in $\mathcal{I}$ at $\mathbf{x}^*$. Since each $P_\mathbf{x}$ defines an orthogonal projection onto its range, every vector $\mathbf{v}$ in $P_\mathbf{x}$ is orthogonal to $\nabla f(\mathbf{x}^*)$; thus, there is no direction available in any of the interactions that locally changes the value of $f(\mathbf{x})$. $\Leftarrow$ Suppose there is no direction $\mathbf{v}$ in any of the interactions $P_\mathbf{x}$ that makes a non-zero inner-product with $\nabla f(\mathbf{x}^*)$. Since $P_\mathbf{x}$ is an orthogonal projection, its range is orthogonal to its nullspace; thus, $P_\mathbf{x}$ must annihilate every vector that makes a non-zero inner product with $\nabla f(\mathbf{x}^*)$, including $\nabla f(\mathbf{x}^*)$ itself if it is nonzero. Since the distribution operator $\mathcal{D}_\mathcal{I}$ is a sum of these orthogonal projections, it too must annihilate $\nabla f(\mathbf{x}^*)$, and thus we have that $\nabla_\mathcal{I} f(\mathbf{x}^*) = 0$. □

Corollary 4. *Every extremum of f is a distributed extremum of f for arbitrary $\mathcal{I}$.*

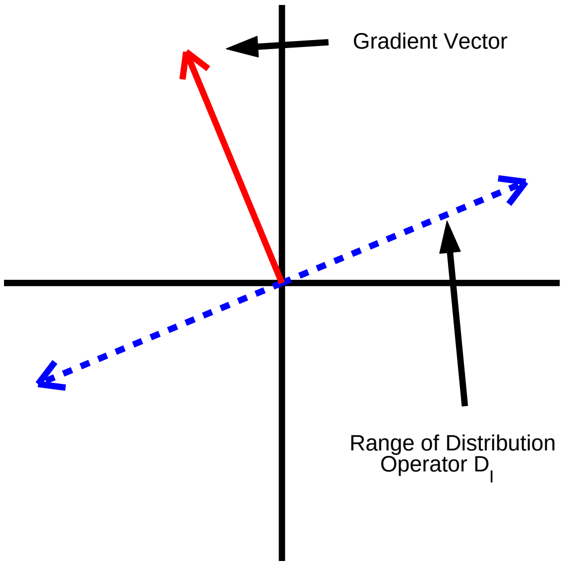


Figure 3.11: Illustration of a distributed extremum. Every tangent vector available to an interaction set (encoded in the range of the associated distribution operator $\mathcal{D}_{\mathcal{I}}$) is orthogonal to the gradient of the function to be optimized.

The preceding corollary is obvious; clearly no distributed solution to an optimization will be better than a centralized solution. Our real goal is to answer the *converse* question we wish to know conditions under which a distributed extremum is also an extremum in the classical sense.

Definition 28 (Distributed Solution of Optimization Problems). *Let f be a smooth real-valued function on $\mathbf{R}^n$, and let $\mathbf{g}(\mathbf{x})$ be a smooth mapping from $\mathbf{R}^n$ to $\mathbf{R}^m$. We will say that an interaction set $\mathcal{I}$ on $\mathbf{R}^n$* weakly solves *the following optimization problem:*

$$\begin{aligned} &\mathcal{O}(f,\mathbf{g}): \\ &\min_{\mathbf{x}\in\mathbf{R}^n} \quad f(\mathbf{x}) \\ &\quad s.t. \quad \mathbf{g}(\mathbf{x}) = 0 \end{aligned}$$

if there exists a coordination $C_{\mathbf{x}}$ with the following properties:

- $\mathbf{g}(\mathbf{x}) = 0 \Rightarrow \mathbf{Dg}(\mathbf{x})\mathcal{D}_{\mathcal{C}_{\mathbf{x}}(\mathcal{I})} = 0.$
- $\mathbf{g}(\mathbf{x}) = 0 \Rightarrow \text{rank}(C_{\mathbf{x}}(I))_{\mathbf{x}} > 0,$

We will say that $\mathcal{I}$ strongly solves *$\mathcal{O}(f,\mathbf{g})$ if there exists a coordination $C_{\mathbf{x}}$ with the above properties, which also satisfies the following:*

- *Every distributed extremum of f relative to $C_{\mathbf{x}}(I)$ satisfying $\mathbf{g}(\mathbf{x}) = 0$ is an extremum of $\mathcal{O}(f,\mathbf{g})$.*

This is another dense definition, so let us attempt to give some intuition. To do so, let us suppose that one has a nominal solution $\mathbf{x}_0$ satisfying the constraints, and would like to improve the value of the objective function if possible under a given interaction structure.

The first part of the definition requires that there exist a coordination which respects the constraint; if one only utilizes tangent vectors available under the coordinated interaction, then this guarantees that a feasible solution remains feasible. The second part of the definition requires that the rank of the coordinated interaction set be positive; this means that there are at least *some* available tangent directions on which to test the improvement of the objective function. Each interaction that is annihilated by the derivative of f is *distributed proof* that the nominal solution is locally optimal relative to some subspace of perturbations. Thus, this means that if $\mathcal{I}$ weakly solves an optimization problem, then every distributed extremum is at least optimal relative to some non-zero subspace of tangent directions. Strong solvability, then, simply means that this distributed proof is sufficient to recover the performance of a global controller.

We thus see that weak solvability guarantees that an optimization problem can be at least "partially" optimized relative to the tangent-space constraints of the interaction set. We would naturally like to know when an optimization is weakly solvable for a class of interactions; the following two lemmas provide such a characterization in terms of coordination capacity.

Lemma 5 (Sufficient Condition for Weak Solvability). *Let M be a manifold, and $\mathcal{I}$ be an interaction set on M with coordination capacity $\operatorname{cap}(\mathcal{I})_{\mathbf{x}} \geq k$ for all $\mathbf{x}$. Consider an optimization problem $\mathcal{O}(f, \mathbf{g})$, where $\operatorname{rank}(\mathbf{Dg}(\mathbf{x})) = m \leq k$ for all $\mathbf{x}$; then $\mathcal{I}$ weakly solves $\mathcal{O}(f, \mathbf{g})$.*

Proof. To prove this, we must exhibit a suitable coordination $C_{\mathbf{x}}$. Consider, at each $\mathbf{x}$ the projection onto the nullspace of $\mathbf{Dg}(\mathbf{x})$; this defines a subspace of tangent vectors at each $\mathbf{x}$, and so an associated tangent subbundle $C_{\mathbf{x}}$. The rank constraint on $\mathbf{Dg}$ implies that $\operatorname{codim} C_{\mathbf{x}} = m \leq \operatorname{cap}(\mathcal{I})_{\mathbf{x}}$; applying the definition of coordination capacity, this implies implies that $\operatorname{rank}(C_{\mathbf{x}}(\mathcal{I}))_{\mathbf{x}} > 0$. This completes the proof. □

This lemma shows that the coordination capacity gives a natural guarantee on a class of optimization problems that can be weakly solved; intuitively, one should interpret this as saying that any optimization problem with fewer constraints than the coordination capacity can be at least partially optimized using the available local interactions. The next lemma shows that the coordination capacity also constrains the class of optimization problems that one can expect to be able to solve "generically".

Lemma 6 (Necessary Condition for Universal Weak Solvability). *Let M be a manifold, and $\mathcal{I}$ be an interaction set on M with coordination capacity $\operatorname{cap}(\mathcal{I})_{\mathbf{x}} < k$ for all $\mathbf{x}$. Then, there exists an optimization problem $\mathcal{O}(f, \mathbf{g})$ with $\operatorname{rank}(\mathbf{Dg}(\mathbf{x})) \leq k$ for all $\mathbf{x}$ that is not weakly solvable by $\mathcal{I}$.*

Proof. Consider the parametrized family of optimization problems $\mathcal{O}(\|\mathbf{x}\|, A)$ specified by:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbf{R}^n} \quad & \|\mathbf{x}\| \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}) = A\mathbf{x} = 0 \\ & A \in \mathbf{R}^{k\times n} \end{aligned}$$

where A is an arbitrary rank k matrix. The constraint function $\mathbf{g}(\mathbf{x})$ defines a codimension-k subbundle of tangent vectors $A_{\mathbf{x}}$. Let us consider a specific point $\mathbf{x}$; by choosing an appropriate A, one can specify an arbitrary codimension-k subspace of $T_{\mathbf{x}}M$. Since $\text{cap}(\mathcal{I})_{\mathbf{x}} < k$, there exists some choice of codimension-k subspace in $T_{\mathbf{x}}M$ such that $\text{rank}(A_{\mathbf{x}}(\mathcal{I}))_{\mathbf{x}} = 0$ (else $\text{cap}(\mathcal{I})_{\mathbf{x}}$ would be greater than or equal to k). Denoting the matrix specifying this subspace A^*, we then have that $\mathcal{O}(f, A^*)$ is not weakly solvable, which proves the desired result. $\square$

Strong solvability, unfortunately, is considerably more difficult to characterize in general; in order to get meaningful results, one must make fairly restrictive assumptions on the allowable objective and constraint functions, and the allowable interaction sets. We will later present a few such special cases: Section 4.3.1 will show an example of an optimization associated with a graph where strong solvability cannot generally be achieved on any non-complete graph, and Section 3.4.1 will show that a very weak restriction on the parameters of the constraint equation implies strong solvability for a special class of problems.

This concludes our treatment of distributed optimization, and indeed, the bulk of the theoretical development of our work; however, at this point the utility of the above tools may not yet be apparent. Let us summarize what we have presented, and how it will play into the remainder of the work:

- We have presented a geometric framework for modeling local interactions, and coordination of those interactions. This abstraction allows us to provide *distributed proof* that vectors contributed by local interactions make positive progress towards a global goal, and also that they respect global constraints.

- We have introduced a notion of capacity for coordination, and shown its connection to solvability of distributed optimization problems under different interaction patterns.

- We have proposed a formal approach for separating functionality from implementation; in the upcoming sections, we will use distributed optimization as a specification of functionality, and we will use distributed gradients as a tool for implementation which provides *distributed proof* that the specified functionality is being carried out.

3.3 A Novel Design Formalism: Specification, Implementation, and Performance

Here we will apply the tools of Section 3.2 to present a general design mechanism for distributed coordination with the following properties:

- *Functionality* will be separated from *implementation.*
- *Functional specifications* will be systematically converted into *distributed dynamics.*
- *Convergence and Performance* of the distributed system will be systematically analyzable in a way that is largely independent of the functional specification, or the network that implements the dynamics.

The goal of this section is primarily to convey a viewpoint, rather than to enumerate a collection of technical results; as a consequence, our development will center around casting the distributed averaging system $\Phi_L(\cdot)$ in our new language.

3.3.1 Specification: Optimization Problems

We introduced the idea of an optimization problem encoding a function in Section 3.2; we will now revisit this in some more detail. Our main goal is to present a simple input-output relationship encoded by an optimization problem; to this end we will make some simplifications for the sake of clarity.

Let us now present an optimization problem which encodes the distributed averaging behavior of $\Phi_L(\cdot)$. Let us suppose we have a vector $\mathbf{u} \in \mathbf{R}^n$, with each component $\mathbf{u}_i$ representing the local input of some node in a network. We would like each member of the network to compute the average of all the local inputs in the network; we will represent their computed values by a global state vector $\mathbf{x} \in \mathbf{R}^n$. Thus, the specification we desire is:

$$\mathbf{x}_i = \frac{1}{n}\sum_j \mathbf{u}_j.$$

Now, consider the following optimization problem:

$$\begin{array}{ll} \min & \|\mathbf{x}\|^2 \\ \text{s.t.} & \mathbf{1}_n^T\mathbf{x} = \mathbf{1}_n^T\mathbf{u} \end{array}$$

This problem has a unique local minimum, and this optimal solution is given by:

$$\mathbf{x}^* = \frac{\mathbf{1}_n\mathbf{1}_n^T}{\mathbf{1}_n^T\mathbf{1}_n}\mathbf{u}.$$

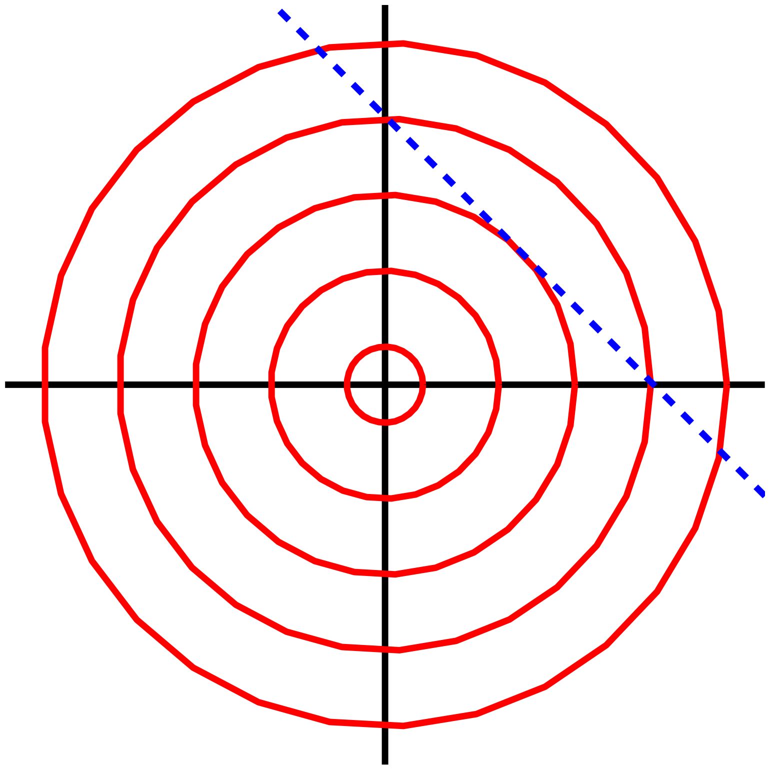


Figure 3.12: An illustration showing the encoding of averaging as an optimization problem. Circles indicate level sets of $\|\mathbf{x}\|^2$, which is to be minimized, and the dashed line indicates the constraint on the sum of the components (required to equal the sum of some local inputs). The optimal solution sets all the components of the vector $\mathbf{x}$ equal to the average of the local inputs.

Evaluating each component of $\mathbf{x}_i$, we see that it is indeed the desired average value; we have thus encoded a function (averaging the components of a vector) in an optimization problem, parametrized by inputs in the constraint equation.

Note that the above optimization problem *specifies* the desired averaging behavior, but makes *no reference* to graphs or dynamics. It is thus truly a representation of functionality that is independent of implementation; we will now present a systematic mechanism for generating implementations, which will naturally recover the Laplacian-based distributed averaging system $\Phi_L(\cdot)$.

3.3.2 Implementation: Distributed Gradient Dynamics

Optimization-based models naturally motivate the examination of gradient dynamics; gradient systems are highly structured, and have a long history of study associated to them. Traditionally, gradient systems have been associated with potential-function models; there are many reasons for this, but a primary reason from a design perspective is that potentials naturally produce distributed dynamics. To see the complication that arises in problems not specified by potentials, consider the gradient of the "averaging" optimization problem:

$$\nabla \|\mathbf{x}\|^2 = 2\mathbf{x}.$$

Note that this vector field does *not* respect the constraint; given a nominal feasible solution, this vector field is not tangent to the constraint surface. Naturally, the quantity one desires here is the *projected gradient*

$$2\left(I_{n\times n} - \frac{\mathbf{1}_n\mathbf{1}_n^T}{\mathbf{1}_n^T\mathbf{1}_n}\right)\mathbf{x}$$

which is simply the gradient projected onto the tangent space of the constraint surface at the point $\mathbf{x}$. Unfortunately, determining this vector requires *globally computed* quantities; indeed, this is the fundamental tension arising in attempting to constrain the dynamics of distributed systems. If one wishes to enforce a *global constraint*, then the standard methods lead to computations which are *intractable* in the sense that they require global control.

Further, let us consider the question of how one might obtain "proof" that the projected gradient algorithm actually implements the specified computation. Note that the projected gradient is defined by a globally computed projection; although it satisfies the constraint, one needs access to *all components* of the distributed gradient vector to have proof that it does so. No node can local information and obtain a guarantee that the dynamics respects the constraint equation. Similarly, the projected gradient provides no distributed guarantee that progress is made; one needs access to the entire vector to verify that its inner product with the gradient is non-negative.

The *distributed gradient*, introduced in Section 3.2.3, naturally solves all of these problems, and motivates the use of dynamics with the following structure:

$$\dot{\mathbf{x}} = -\nabla_{\mathcal{I}} f(\mathbf{x})$$

where $\mathcal{I}$ is some interaction set. Let us review the relevant properties of the distributed gradient that make this dynamics interesting:

- The individual interactions comprising the distributed gradient provide *distributed proof* that the objective function improves, and that the constraint equation is satisfied.
- One need only compute the *gradient* of the objective function, not the *projected gradient*; in many practical situations, this simplification is sufficient to turn an intractable problem into a tractable one.
- The distributed gradient allows us to systematically discuss this type of dynamics for *all* distributed systems simultaneously, as it is parametrized explicitly by $\mathcal{I}$, and the above properties are independent of the specific choice of $\mathcal{I}$.

Before applying the distributed gradient dynamics to our averaging problem, let us prove a small technical lemma that will make analysis of distributed gradient systems very simple.

Lemma 7 (Convergence of Distributed Gradient Flow). *Let $f(\mathbf{x})$ be a smooth real-valued function on $\mathbf{R}^n$ which is bounded below by some constant. Then, for any interaction set $\mathcal{I}$, the dynamics*

$$\dot{\mathbf{x}} = \nabla_{\mathcal{I}} f(\mathbf{x})$$

has the following properties:

1. $\mathbf{x}(\infty) = \lim_{t\to\infty} \mathbf{x}(t)$ *exists for arbitrary initial values* $\mathbf{x}(0)$.
2. $\mathbf{x}(\infty)$ *is a distributed extremum of f relative to $\mathcal{I}$.*

Proof. Recall that the distributed gradient makes a non-negative inner product with the gradient of f; thus,

$$\frac{d}{dt} f(\mathbf{x}(t)) = \langle \nabla f(\mathbf{x}), \nabla_{\mathcal{I}} f(\mathbf{x}) \rangle \leq 0.$$

Since f is assumed bounded below by a constant, we have that $\mathbf{x}(t)$ must converge to a submanifold in which

$$\frac{d}{dt} f(\mathbf{x}(t)) = \langle \nabla f(\mathbf{x}), \nabla_{\mathcal{I}} f(\mathbf{x}) \rangle = 0.$$

Now, at any point $\mathbf{x}$ on this submanifold, the tangent vector $\dot{\mathbf{x}}$ must be orthogonal to the gradient of f at $\mathbf{x}$. However, $\nabla_{\mathcal{I}} f(\mathbf{x})$ is defined by a sum of orthogonal projections of the gradient onto various subspaces; each of these terms must make a non-negative inner product with the gradient vector. Since the overall inner product with the gradient is zero, each of these projection terms must be zero. Thus, $\dot{\mathbf{x}} = 0$ on the submanifold on which f is invariant; applying the smoothness requirement, and the fact that $\mathbf{x}$ converges to this submanifold, this proves the first claim (that the limit exists). Now, from Lemma 4, we know that $\nabla_{\mathcal{I}} f(\mathbf{x}) = 0$ if and only if $\mathbf{x}$ is a distributed extremum of f, which proves the second claim. $\square$

Corollary 5 (Characterization of Equilibrium of Distributed Gradient Dynamics). *For each interaction $P_{\mathbf{x}} \in \mathcal{I}$, we have $P_{\mathbf{x}} \nabla f(\mathbf{x}) = 0$ at equilibrium.*

The preceding statements, though simple, are actually quite powerful. In fact, they precisely embody the "stability - invariance - equilibrium" architecture we have sought throughout this work. Let us examine these three points explicitly.

Stability

The distributed gradient flow provably converges to a limit; this property is independent of the functional specification, or of the interaction pattern that implements the flow.

Invariance

The constraint function is *a priori* invariant if one applies the appropriate coordination to the interaction set; thus, a solution that is initially feasible remains feasible throughout time.

Equilibrium

Every equilibrium is a distributed extremum of f relative to the interaction set $\mathcal{I}$; this gives a collection of constraints for each interaction $P_{\mathbf{x}} \in \mathcal{I}$ that must be satisfied at an equilibrium.

We have thus recovered, from "first principles", the design architecture that we had envisioned at the end of Chapter 2. Further, we have done so in a way that separates the *functionality* of coordination from the *implementation* of coordination, as per our desires in the early sections of Chapter 3. Now, let us conclude our exposition of this design mechanism by deriving $\Phi_L(\cdot)$ with our new tools.

We have already presented an optimization problem encoding an averaging computation. We wish to couple this functional specification with a dynamical system which will implement it; to do so, we will synthesize a dynamical system according to the following specification:

$$\begin{array}{rl} \min & \lim_{t\to\infty} \|\mathbf{x}(t)\|^2 \\ \text{s.t.} & \mathbf{1}_n^T\mathbf{x}(t) = \mathbf{1}_n^T\mathbf{x}(0) \end{array}$$

We will carry out this design on a connected graph G, and we will assume the pairwise interaction set $\mathcal{I} = \mathcal{P}_E(G)$, which we recall is defined by:

$$\mathcal{I} = \{\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j \mid (i,j) \in G\}.$$

Now, we wish to enforce the invariance constraint on this interaction set, so we will apply the coordination $C_{\mathbf{x}}$ specified by $\mathbf{1}_n^T\mathbf{v} = 0$; from our discussion in Section 3.2.2, we know that the distribution operator for $\mathcal{P}_E(G)$ coordinated by $C_{\mathbf{x}}$ is given by:

$$\mathcal{D}_{C_{\mathbf{x}}(\mathcal{I})}(\mathbf{x}) = \frac{1}{2}L(G).$$

The distributed gradient relative to the coordinated interaction set $\mathcal{I}'$ is given by:

$$\begin{array}{rcl} \nabla_{\mathcal{I}'}\|\mathbf{x}\|^2 & = & \mathcal{D}_{\mathcal{I}'}(\mathbf{x})\nabla\|x\|^2 \\ & = & L(G)\mathbf{x}. \end{array}$$

Finally, the distributed gradient dynamics is

$$\dot{\mathbf{x}} = -L(G)\mathbf{x}.$$

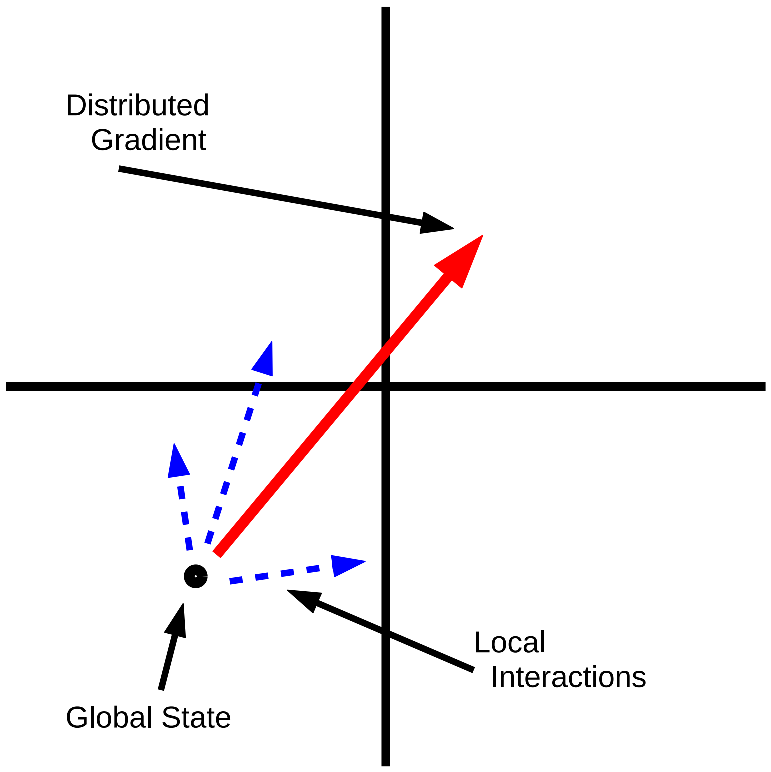


Figure 3.13: Distributed gradient dynamics arising from local interactions; the system continues to evolve until all local interactions produce zero tangent vectors.

So, we have precisely recovered the dynamics specified by $\Phi_L(G)$. With our new tools in hand, we can immediately prove all of its desirable properties.

- From the properties of a distributed gradient flow, we have that $\mathbf{x}$ must go to some equilibrium.
- The invariance of $\mathbf{1}_n^T\mathbf{x}(t)$ is guaranteed by the construction of the distributed gradient.
- From the characterization in Corollary 5, we have that each interaction must annihilate the gradient; thus,

  $$\frac{1}{2}\left(\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j - \mathbf{e}_i \otimes \mathbf{e}_j - \mathbf{e}_j \otimes \mathbf{e}_i\right)\mathbf{x} = 0$$

  for all (i, j) in G, which evaluates to: $x(i) = x(j)$ for every edge (i, j). Since G is connected, this implies that $\mathbf{x}$ is along the subspace spanned by $\mathbf{1}_n$.

This collection of results is quite satisfying, in the sense that we have indeed recovered all of the properties of $\Phi_L(\cdot)$ without using any specific details of its specification; the distributed gradient design formalism provides all the relevant results as desired.

3.3.3 Interlude: Summary of Main Technical Results

We have now introduced most of the technical machinery necessary to understand the bulk of this work. Before continuing with analysis of some special cases and particular properties of the overall mechanism, we collect here a brief summary of what we have accomplished thus far.

Theorem 1 (Characterization of Solvability with Capacity and Distribution). *Let f be a real-valued function defined on some manifold M of dimension n, and consider the optimization problem $\mathcal{O}$:*

$$\min_{\mathbf{x} \in N} \quad f(\mathbf{x}),$$

where N is a submanifold of M and has codimension k. Let $\mathcal{I}$ be an interaction set on M with coordination capacity $\operatorname{cap}(\mathcal{I})_{\mathbf{x}}$. Let $\mathcal{D}_{\mathcal{I}}(\mathbf{x})$ denote the distribution operator associated with the interaction set. Then, the following properties hold:

1. $\operatorname{cap}(\mathcal{I})_{\mathbf{x}} < k \Rightarrow \mathcal{I}$ *does not weakly solve $\mathcal{O}$ for all choices of f and N.*

2. $T_{\mathbf{x}}N \subset \operatorname{range} \mathcal{D}_{\mathcal{I}}(\mathbf{x})$ *for all $\mathbf{x} \in N \Rightarrow \mathcal{I}_{\mathbf{x}}$ strongly solves $\mathcal{O}$.*

Proof. The first property is a restatement of Lemma 6, and the latter follows directly from the fact that every direction in the range of the distribution operator does not annihilate any directions in the tangent bundle of the constraint manifold. □

Theorem 2 (Solution by Distributed Gradient Flow). *Let f be a real-valued function defined on some manifold M of dimension n, and consider the optimization problem $\mathcal{O}$:*

$$\min_{\mathbf{x} \in N} \quad f(\mathbf{x}),$$

where N is a submanifold of M and has codimension k. Let $\mathcal{I}$ be an interaction set on M that weakly solves $\mathcal{O}$. Consider the following dynamics, from an arbitrary initial point in N:

$$\dot{\mathbf{x}} = -\nabla_{\mathcal{I}} f(\mathbf{x}).$$

This dynamics has the following properties:

1. *The flow converges to an equilibrium point.*

2. *The equilibrium is a distributed extremum of f relative to $\mathcal{I}$.*

3. *If the interaction set $\mathcal{I}$ strongly solves the optimization problem $\mathcal{O}$, then any equilibrium point of the dynamics is a locally optimal solution of $\mathcal{O}$.*

Proof. This is a combination of Lemmas 6 and 7, and the definition of a distributed extremum. □

3.3.4 Performance: Efficiency of Distribution

Since we regard each interaction set $\mathcal{I}$ as inducing a distribution operator which converts a *global specification* into a *distributed implementation*, it is natural to ask how well this distributed implementation performs relative to a centralized implementation.

A natural performance metric for distributed gradient systems is the differential reduction in the objective function normalized by the magnitude of the tangent vector; more generally, we would like to know the extent to which the distributed implementation "points in the same direction" as the tangent vector that would have been produced by a centralized implementation. We thus define the following *efficiency function* associating to a vector field V, interaction set $\mathcal{I}$, and point $\mathbf{x}$ in some manifold the following value:

$$\Psi_I(V, \mathbf{x}) = \frac{\langle V(\mathbf{x}), V_{\mathcal{I}}(\mathbf{x})\rangle}{\|V(\mathbf{x})\| \|V_{\mathcal{I}}(\mathbf{x})\|}$$

where $V_{\mathcal{I}}(\mathbf{x}) = \mathcal{D}_{\mathcal{I}}(\mathbf{x})V(\mathbf{x})$ is the distributed version of the vector field V. We will only discuss efficiency for situations where the tangent vector is non-zero and in the range of the distribution operator so the ratio will always be well defined. From the properties of the distribution operator, we have the following *a priori* bounds for the efficiency function:

$$0 < \Psi_I(V, \mathbf{x}) \leq 1.$$

Now, if one considers the "global control" interaction, with the associated distribution operator $\mathcal{D}_{\mathcal{I}}(\mathbf{x}) = I_{T_{\mathbf{x}}M}$, it is clear that a centralized controller achieves unit efficiency (as one would expect). For general interactions, we can obtain a simple bound on Ψ, which intuitively tells us that the efficiency of a distributed interaction is inversely related to how strongly its action "skews" the tangent space. In order to quantify this, we will need the following linear algebraic notion:

Definition 29 (Projected Condition Number). *Let T be a linear operator on $\mathbf{R}^n$, and let $\mathbf{v}$ and $\mathbf{w}$ be arbitrary vectors in* range(T) *with unit norm. We define the* projected condition number *of T to be*

$$\kappa_P(T) = \frac{\max_{\mathbf{v}} \|T\mathbf{v}\|}{\min_{\mathbf{w}} \|T\mathbf{w}\|}.$$

For a self-adjoint T, this is just the ratio of the largest eigenvalue of T (in absolute value) to the smallest *non-zero* eigenvalue; clearly, $\kappa_P(T) \geq 1$. We can now state our bounding lemma.

Lemma 8 (Characterization of Efficiency). *Let M be a manifold, $\mathbf{x}$ be a point in M, $\mathcal{I}$ be an interaction set on M, and V be a vector field such that $V(\mathbf{x})$ is in* range$(\mathcal{D}_{\mathcal{I}}(\mathbf{x}))$*. Then, we have the following bound:*

$$\Psi_{\mathcal{I}}(V, \mathbf{x}) \geq \frac{1}{\kappa_P(\mathcal{D}_{\mathcal{I}}(\mathbf{x}))}.$$

Proof. The efficiency is invariant under scaling of the vector $V(\mathbf{x})$; we can thus assume that this

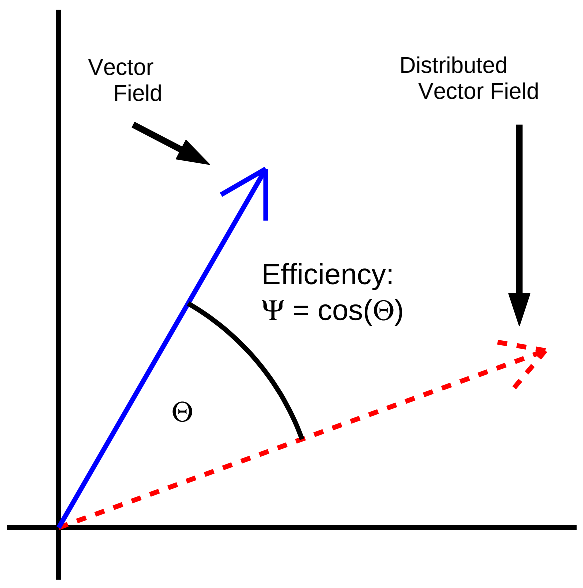


Figure 3.14: Distributed efficiency measures the extent to which the performance of a global controller is lost due to the distributed implementation; it is characterized geometrically by the angle between the input vector field and the distributed vector field, and algebraically by the condition number of the distribution operator.

vector has unit norm. Then, we have the following expression:

$$\frac{\langle V(\mathbf{x}), \mathcal{D}_{\mathcal{I}}(\mathbf{x})V(\mathbf{x})\rangle}{\sqrt{\langle V(\mathbf{x}), \mathcal{D}_{\mathcal{I}}^2(\mathbf{x})V(\mathbf{x})\rangle}}$$

Now, let $\lambda_{\min}(\mathbf{x})$ and $\lambda_{\max}(\mathbf{x})$ denote the smallest and largest (in absolute value) *non-zero* eigenvalues of $\mathcal{D}_{\mathcal{I}}(\mathbf{x})$; since we have assumed that $V(\mathbf{x})$ is in the range of $\mathcal{D}_{\mathcal{I}}(\mathbf{x})$, the numerator is bounded below by $\lambda_{\min}(\mathbf{x})\|V(\mathbf{x})\|^2$, and the denominator is bounded above by $\lambda_{\max}(\mathbf{x})\|V(\mathbf{x})\|$; since we have assumed $\|V(\mathbf{x})\| = 1$, we now have

$$\Psi_{\mathcal{I}}(V, \mathbf{x}) \geq \frac{1}{\kappa_P(\mathcal{D}_{\mathcal{I}}(\mathbf{x}))}$$

as desired. □

So, we now have a systematic method by which to bound the efficiency of a distributed implementation; this characterization is quite interesting in that it is not coupled to the vector field to be distributed. Admittedly, this is only a bound, but it provides us with some notion of "intrinsic efficiency" associated with a particular interaction set $\mathcal{I}$; indeed, as we saw before that interaction sets have an associated coordination capacity, we see also that they have associated efficiency properties that are, at least in terms of this bound, independent of the task they implement.

3.4 Linear Dynamic Coordination

In the preceding sections, we have introduced a design mechanism for distributed coordination; we specify coordination tasks as optimization problems *independent* of implementation, and then systematically convert these specifications into implementations based on a given distributed interaction. One tacit assumption in this development was that the coordination task to be performed was *static*; the specified task encoded in the optimization problem did not include a mechanism for time dependence. In this section we will develop a special case of coordination dynamics which we call "linear coordination" for reasons which will become obvious; this class of designs will naturally generalize to tracking dynamically specified coordinated states. We begin by introducing linear coordination problems in the static case.

3.4.1 The Linear Coordination Problem

Consider again the standard distributed manifold $M_{\mathbf{R}}(\cdot)$ assigning a real-valued state $x(i)$ to each node in a connected graph G. Linear coordination for this scenario is specified by the following optimization problem:

$$\begin{array}{cc} \min\limits_{\mathbf{x}} & \mathbf{x}^T Q \mathbf{x} \\ \text{s.t.} & \mathbf{b}^T \mathbf{x} = \mathbf{b}^T \mathbf{u} \end{array}$$

where $\mathbf{x}, \mathbf{b}, \mathbf{u} \in \mathbf{R}^n$ and Q is a positive-definite symmetric matrix in $\mathbf{R}^{n\times n}$. As before, we interpret $\mathbf{u}$ as a vector of local inputs, and $\mathbf{x}$ as the concatenated state vector with $\mathbf{x}_i = x(i)$. We will also assume that each component of $\mathbf{b}_i$ is non-zero; we make this assumption for well-posedness, but it is a sensible requirement in that it couples the states of all the members of the network in the constraint equation.

The mapping specified by this optimization problem can be calculated to be:

$$\mathbf{x}^* = \frac{\mathbf{b}^T \mathbf{u}}{\mathbf{b}^T Q^{-1} \mathbf{b}} Q^{-1} \mathbf{b}.$$

Let us note two aspects of this structure. First, note that the coordinated state vector lies along $Q^{-1}\mathbf{b}$; we can thus interpret this as solving a certain linear system of equations distributed over the set of agents. For $Q = I$, this in fact effects a projection; the asymptotic state is precisely the projection of the inputs $\mathbf{u}$ onto the subspace spanned by $\mathbf{b}$. This brings us to the second aspect of the mapping, the leading coefficient. Not only does the mapping specify a subspace of coordinated states, but it also executes a computation; again, if $Q = I$ then this leading coefficient is the projection coefficient of the input vector on the span of $\mathbf{b}$.

We refer to this specification as "linear coordination" for two reasons: its gradient $2Q\mathbf{x}$ is a linear function of the state vector, and its constraint equation defines the "same" linear subspace of the tangent space over all of $\mathbf{R}^n$. Applying the associated coordination to the pairwise interaction $\mathcal{P}_E(G)$, we obtain the following coordinated interaction set:

$$\left\{ \frac{1}{b_i^2 + b_j^2} \left(b_j \mathbf{e}_i \otimes \mathbf{e}_i + b_i \mathbf{e}_j \otimes \mathbf{e}_j - b_i b_j \mathbf{e}_i \otimes \mathbf{e}_j - b_i b_j \mathbf{e}_j \otimes \mathbf{e}_i \right) \;\middle|\; (i,j) \in G \right\}.$$

Now, assuming that each $b_i \neq 0$ and that G is connected, we can show that the associated distribution operator is has rank $n-1$. To see this, note that each term in the interaction set above is an orthogonal projection onto the span of vectors of the form:

$$-b_j \mathbf{e}_i + b_i \mathbf{e}_j.$$

Since G is connected, it has a spanning tree with exactly $n-1$ edges. Each edge in the tree induces an independent vector field independent, and so the resulting distribution operator $\mathcal{D}_{\mathcal{I}}$ has rank $n-1$. This, in turn, implies that the coordinated interaction set *strongly solves* the optimization problem specifying the coordination problem; this is because the constraint defines a codimension-1 affine subspace, and so one needs a set of $n-1$ linearly independent vectors to span it.

So, we now know that pairwise interaction on a connected graph is sufficient to implement linear coordination as described above. The distribution operator is in general complicated to write explicitly, but for each choice of $\mathbf{b}$ it is a fixed matrix $A(\mathbf{b})$; the resulting distributed gradient dynamics is:

$$\dot{\mathbf{x}} = -A(\mathbf{b})Q\mathbf{x}.$$

For $Q = I$ and $\mathbf{b} = \mathbf{1}_n$, we recover the distributed averaging dynamics $\Phi_L(G)$ as a special case. One should note an interesting point about this equation; depending on the nature of Q, the product $A(\mathbf{b})Q$ need not be symmetric. Nonetheless, we have *a priori* stability analysis for this equation, because of the distributed gradient design mechanism.

As can be seen, linear coordination also yields particularly simple efficiency analysis; the efficiency is globally determined by the projected condition number of the distribution operator $A(\mathbf{b})$. Similar relationships between performance and spectral properties of matrices associated with graphs have been previously explored (see []) for the special case of the Laplacian dynamics.

3.4.2 Dynamic Coordination

We have now familiarized ourselves with the basic linear coordination problem, and shown the associated distributed gradient dynamics. For fixed inputs $\mathbf{u}$ encoded in the initial values $\mathbf{x}(0)$, the distributed gradient dynamics carries out the projection operation discussed in the previous section.

It is natural to ask whether one can effect *tracking* of this coordinated state; that is, if instead of a *fixed* input $\mathbf{u}$ we had a *dynamic* input $\mathbf{u}(t)$, we would like to design a mechanism which assures that the system is "dynamically coordinated". A natural answer to this question is available from our "Stability - Invariance - Equilibrium" architecture.

Thus, let us consider the time-varying specification:

$$\begin{array}{cc} \min\limits_{\mathbf{x}} & \mathbf{x}^T Q\mathbf{x} \\ \text{s.t.} & \mathbf{b}^T\mathbf{x} = \mathbf{b}^T\mathbf{u}(t). \end{array}$$

It is not immediately clear what one should do to implement this kind of coordination, but our notation in fact suggests a good candidate. Naively writing out the specification for the distributed gradient dynamics as per our usual pattern (except explicitly writing the input $\mathbf{u}(t)$ in place of the initial values which encode them)

$$\begin{array}{cc} \min & \lim_{t\to\infty} \mathbf{x}^T Q\mathbf{x} \\ \text{s.t.} & \mathbf{b}^T\mathbf{x}(t) = \mathbf{b}^T\mathbf{u}(t) \end{array}$$

the solution becomes fairly clear: one needs to *dynamically compensate* for the evolution of the desired invariant surface. Consider then, the following simple modification of our design:

$$\dot{\mathbf{x}} = -A(\mathbf{b})Q\mathbf{x} + \dot{\mathbf{u}}.$$

This equation is quite simple, but within it lie two important points. First, let us consider the evolution of the constraint equation:

$$\begin{aligned} \frac{d}{dt}\left(\mathbf{b}^T\mathbf{x}\right) &= \mathbf{b}^T\dot{\mathbf{x}} \\ &= -\mathbf{b}^T A(\mathbf{b})Q\mathbf{x} + \mathbf{b}^T\dot{\mathbf{u}} \\ &= \mathbf{b}^T\dot{\mathbf{u}}, \end{aligned}$$

where the second line follows from the fact that the distributed gradient dynamics leaves the *static constraint* invariant. Clearly, if we ensure (as per our usual model) that the initial values of the system encode the initial inputs, then we have that the constraint equation is satisfied for all t. Thus, for example, if the inputs ever stopped changing and remained constant after some time T, the system would track the appropriate invariant manifold and then asymptotically settle to the coordinated state specified by $\mathbf{u}(T)$. More generally, we now have an explicit input-output representation for the tracking of this time-varying input; we can apply the standard sorts of analysis typical to linear systems (transfer functions, frequency domain performance characterizations, and so on). We will

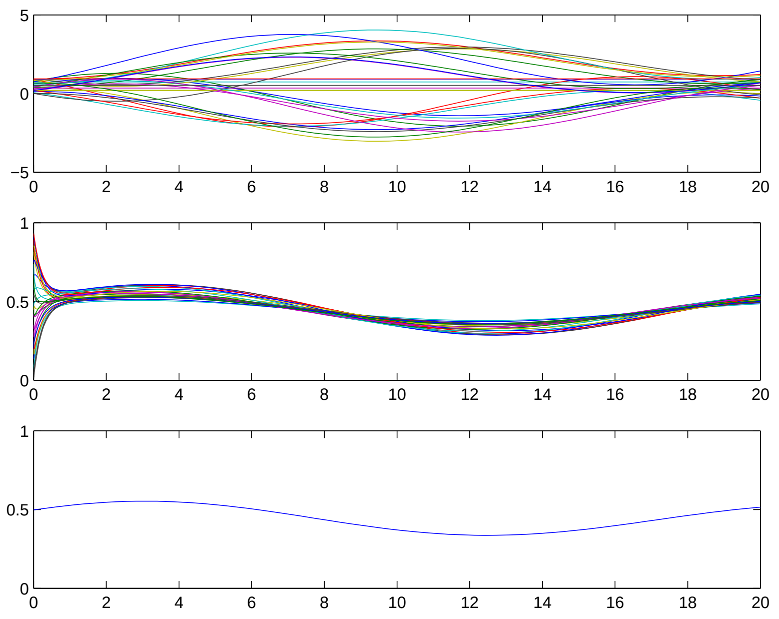


Figure 3.15: Tracking the average of a slowly varying set of inputs. The top plot shows trajectories of the local inputs $u(i)$ on some connected graph G. The center plot shows the evolution of the state variables $x(i)$, and the bottom plot shows the evolution of the average of the $u(i)$ terms. Clearly, this slowly varying input is well tracked by all the members of the network.

not carry out this analysis, but we will return to it in Section 4.4, where it will provide a natural interpretation for a distributed filtering mechanism. Some hints of this behavior can be seen in the two Figures 3.15 and 3.16.

So, we have presented a mechanism that naturally accomodates dynamic tracking of coordinated states. In closing, we should note that the principle of dynamically tracking the "right" invariant manifold is one that, in principle, generalizes to nonlinear problems as well; the main difficulty in these cases (and the reason for which these problems are not treated in this work) is that one may not be able to effect tracking in general if the matrix $\mathbf{Dg}(\mathbf{x})$ (the derivative of the constraint equation) loses rank over the course of the dynamics.

3.4.3 Robust Coordination

In this section we will present a simple mechanism for providing dynamic coordination in a way that is robust, in the sense that it will reject isolated disturbances in the implementation of the dynamics. Further, it will not require differentiation of the dynamic inputs, and so should be regarded as more robust in that regard as well. Finally, it will make the system robust to any imprecision in the initial values. Because we will have greater need to write explicit formulae in this section, we will focus solely on the dynamic version of the distributed averaging system $\Phi_L(\cdot)$; there is nothing special about this choice, and the following development can be carried out for an arbitrary linear

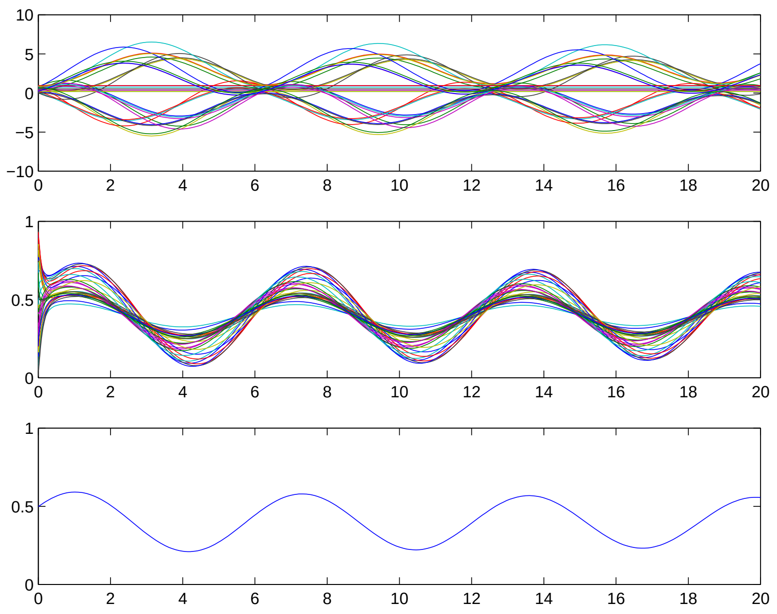


Figure 3.16: Tracking the average of a rapidly varying set of inputs. Again, the top plot shows the local inputs $u(i)$, the center shows the local state variables, and the bottom shows the average of the local inputs. Here we see that although the target quantity is tracked well averaged across the nodes, individual nodes can have significantly larger transient errors than for the slower input.

coordination problem.

Let us consider the individual node dynamics for the distributed averaging system, with the dynamic modification discussed in the previous section:

$$\dot{x}(i) = \sum_{j \in N_i} (x(j) - x(i)) + \dot{u}(i)$$

where the $u(i)$ term denotes the local input at node i. Let us rewrite this in an alternative integral form:

$$\begin{aligned} x(i,t) &= u(i) + \sum_{j \in N_i} \delta(i,j,t) \\ \dot{\delta}(i,j) &= (x(j) - x(i)). \end{aligned}$$

It is important to note here that although the time-evolution of the $x(i)$ variables has not been changed, *we have changed the underlying distributed dynamical system.* Before exploring the implications of this change, let us make the specification of the new system clear.

The new distributed manifold will be denoted $M_{\mathbf{R},E}(\cdot)$ indicating that the assigned states will be induced by the edge structure. For each i in a graph G, we assign the local state space:

$$M_i = \mathbf{R} \times \mathbf{R}^{|N_i|}.$$

Thus, each node is assigned a local state independent of its edge structure, which we will denote $x(i,t)$, and a state associated with each of its neighbors in the graph, which we denote $\delta(i,j,t)$. We will refer to these quantities as "flux" terms; the reason for this naming will become clear in the following section.

Now, returning to the integral formulation of the dynamics, we see that the local inputs are no longer differentiated, nor are the $x(i)$ variables subject to any imprecision in initial values; they are not true "states", and instead are induced by a static mapping from the flux terms and the current value of the local input (we will modify this in a moment). For now, one only need worry about perturbations or imprecisions in the evolution of the flux variables. As one might expect, one can apply standard feedback concepts to this system, and we will present a simple solution:

$$\begin{aligned} x(i,t) &= u(i) + \sum_{j \in N_i} \delta(i,j,t) \\ \dot{\delta}(i,j) &= (x(j) - x(i)) - (\delta(i,j,t) + \delta(j,i,t)). \end{aligned}$$

Here we have added a "coordination feedback" term on the flux variables; let us examine what is going on here in some detail. First, note that

$$\dot{\delta}(i,j) + \dot{\delta}(j,i) = -2\left(\delta(i,j) + \delta(j,i)\right)$$

and so the flux terms on each link exponentially converge to be equal and opposite. Now, summing over the $x(i)$ variables, we see that

$$\sum_{i \in G} x(i,t) = \sum_{i \in G} u(i,t) + \sum_{(i,j) \in G} \delta(i,j,t) + \delta(j,i,t)$$

where the flux terms in the latter sum decay to zero exponentially per our previous argument. This implies that this coordination feedback *rejects perturbations to the desired invariant manifold from the coordination specification.* We have thus made this system "robust" to a variety of imperfections that may arise in implementation. The only possible misgiving one might have about the system we have designed here is that it has secretly "violated causality"; in order to implement this system, each node must be able to instantaneously deliver their value of $u(i,t)$ to their neighbors, since the neighbor's dynamics depends directly on $x(i,t)$, which in turn is specified by $u(i,t)$. The following specification remedies this problem:

$$\begin{aligned} \dot{x}(i) &= u(i) - x(i) + \sum_{j \in N_i} \delta(i,j,t) \\ \dot{\delta}(i,j) &= (x(j) - x(i)) - (\delta(i,j,t) + \delta(j,i,t)). \end{aligned}$$

Here we have simply added a first-order lag between the local inputs and the local state $x(i)$; this is now a "strictly causal" system, in the sense that no node requires *instantaneous access* to its neighbors states.

This linear system can be shown to be asymptotically stable, but the proof is uninformative so we will not present it. The important point to note is that this system must go to equilibrium if the $u(i)$ terms go to equilibrium; thus asymptotically we must have:

$$\sum_{i \in G} x(i) = \sum_{i \in G} u(i).$$

So, we see that the specification for the desired invariant manifold is upheld asymptotically. This system thus provides an asymptotically stable and causal mechanism for robustly executing the distributed averaging operation.

3.4.4 Reconfigurable Coordination

Throughout our discussion so far, we have tacitly assumed that a single group of nodes was attempting to coordinate its behavior; this was primarily to avoid clutter in the presentation, but one can certainly imagine scenarios in which there are multiple groups coordinating their behavior, and that membership in the groups is dynamic. One would like to have some mechanism by which to induce coordination that respects this property; we call this property *reconfigurability.* In this section, we will show a reconfigurable version of the distributed averaging system. Again, this development is not intrinsically tied to the averaging specification, but making a concrete choice for the specification significantly clarifies the exposition. The functionality we wish to accomplish is the same distributed averaging behavior of $\Phi_L(G)$, except we want to ensure this on every connected component of G. This is simple enough for static graphs; since nodes which are not in the same subgroup do not influence each others' dynamics, this property is naturally accomplished. If, however, the group membership varies over time, as depicted in Figure 3.17, the standard approach does not solve the problem. In order to see this, consider beginning with two subgroups, which each execute their averaging coordination, and are subsequently merged. Again, the dynamics will drive the nodes to the coordinated state of the *merged group.* If the groups were to split again, then all the nodes would be thereafter "contaminated" by the coordinated state of the merged subgroup.

Interestingly, we have already "accidentally" presented the solution to this problem. Consider again the integral realization of the averaging system presented in the previous section (these results

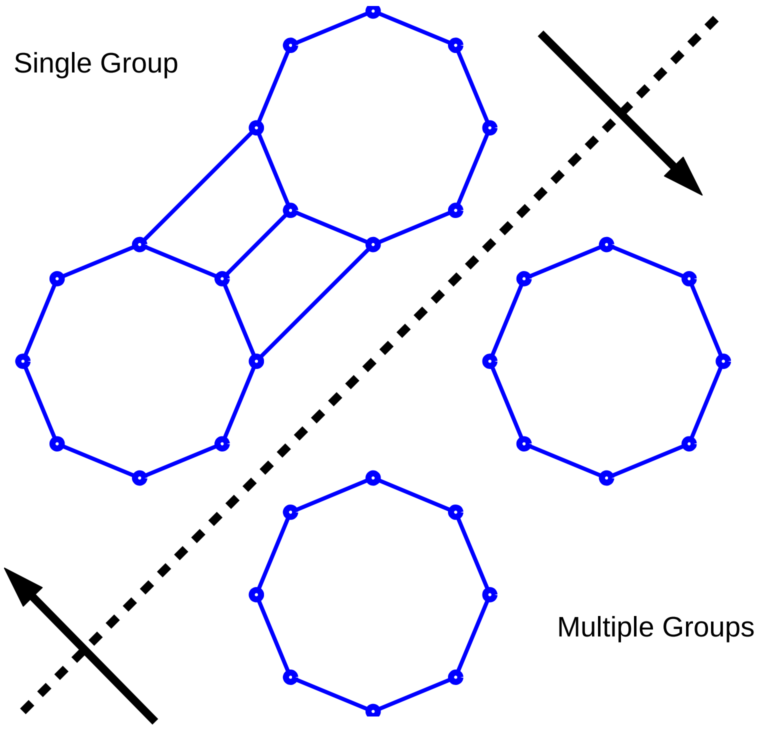


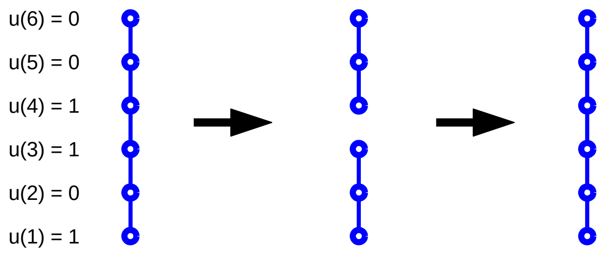
Figure 3.17: *Reconfigurable* coordination refers to the ability to adapt to dynamic changes in group membership. In the case of the averaging system $\Phi_L(\cdot)$, we wish each connected component to track its own average quantity, and for the dynamics to naturally adapt to merging and splitting of groups.

can be recovered for the "robust" versions):

$$\begin{aligned} x(i,t) &= u(i) + \sum_{j \in N_i} \delta(i,j,t) \\ \dot{\delta}(i,j) &= (x(j) - x(i)). \end{aligned}$$

Now we will see the significance of the "flux" terminology. Considering the sum of the $x(i)$ terms as a conserved quantity corresponding to an invariant manifold, we see that each $\delta(i,j)$ term is precisely the amount that has flowed from node i to node j; clearly, $\delta(i,j) + \delta(j,i) = 0$. Upon losing a neighbor in the time-evolution of the network, one can simply "undo" the flow of the conserved quantity; this is accomplished by the following dynamics:

$$\begin{aligned} x(i,t) &= u(i) + \sum_{j \in N_i(t)} \delta(i,j,t) \\ \dot{\delta}(i,j) &= \begin{cases} (x(j) - x(i)) \; j \in N_i \\ 0 \text{ else.} \end{cases} \end{aligned}$$



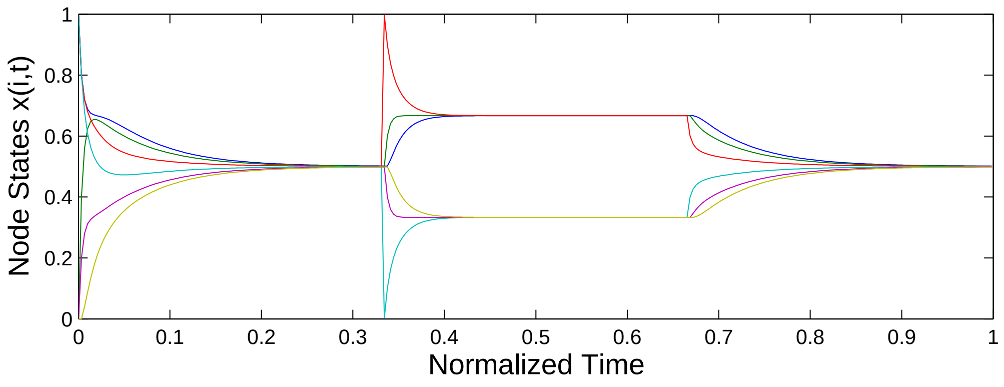


Figure 3.18: Illustration of reconfiguration dynamics for the averaging system $\Phi_L(\cdot)$. The network is initially a connected linear chain, with the local inputs $u(i)$ listed next to the associated nodes. The network splits, and then merges again; the dynamics of the $x(i,t)$ variables is illustrated in the lower plot. We see that the network first approaches the coordinated state of the merged network, then undergoes a non-smooth jump to reach the coordinated states of the two sub-networks. Finally, when the network merges again, it recovers the global coordinated state.

Summing over a connected component of G, say G',

$$\sum_{i \in G'} x(i,t) = \sum_{i \in G'} u(i,t) + \sum_{(i,j) \in G'} \delta(i,j,t) + \delta(j,i,t) = \sum_{i \in G'} u(i,t).$$

This equation shows that in each connected component, we have the desired invariant quantity. We have essentially tracked the "transport" of a conserved quantity in our system, and reversed this transport as necessary to ensure that we are only enforcing coordination relative to the desired connected component. This is situation is depicted in Figure 3.18.

Note again the implications of the "Stability - Invariance Equilibrium" architecture; the stability and equilibrium properties ensure that *a* coordinated state is achieved in each connected component, and the invariance has provided a natural mechanism for selecting the *desired* coordinated state after a splitting of the network.

Chapter 4

Dynamic Coordination: Applications

The preceding chapter developed a class of tools for designing and analyzing distributed systems in a systematic way; this chapter applies these ideas to a number of application areas, broken down by category. Sections 4.1 and 4.2 provide several applications based on the distributed averaging system with the "dynamic coordination" extensions from Section 3.4, while Section 4.3 presents some new applications that can be explored with the new tools developed in the preceding chapters. Each application is intended to illustrate some particular aspect of designing distributed systems.

Section 4.1 examines dynamic coordination for systems which need access to some time-varying global quantity. The basic idea underlying each of these applications is that of a "distributed observer", which is a system that dynamically tracks some function of the global states of a system.

Section 4.2 considers a class of distributed data processing problems, in which each member of some network has access to an element of some larger data set, and the group wishes to perform various types of data analysis on this set; these applications range from statistical computations to distributed model building.

Section 4.3 presents a class of problems we call "topological coordination", in which the coordinated state is itself induced by the structure of the network. We present some standard graph-theoretic computations, as well as a treatment of the "distributed accounting" problem presented in the introductory text.

Throughout this section, the reader is assumed to be familiar with the design and analysis techniques presented in Chapter 3; while the dynamical systems and associated applications presented should be intuitively understandable without any special knowledge, we will make use of previous results for arguing that the designs presented execute the desired behaviors.

4.1 Distributed Observer-Based Designs

This section deals with applications of "distributed observers". An observer is simply a dynamic mechanism whose purpose is to track the evolution of some dynamic process and report it to some other functional unit which makes use of this information.

The two examples here demonstrate distributed observer design interacting with two disparate classes of dynamical analysis. In Section 4.1.1, we show "feedback control design" applied to the distributed averaging tracking system, in order to ensure desirable steady-state error properties when applied to a special class of inputs; this design results in a mechanism for (among other things) synchronization of multiple non-uniform clocks. The contribution of the distributed observer design lies in *quantifiable distributed tracking performance*, which enables the proof of the desired property.

Section 4.1.2 shows distributed observer design in the context of a classical nonlinear dynamical system, the Lotka-Volterra dynamics. Here the distributed observer is used to estimate the "competition term" entering the Lotka-Volterra equations without requiring global access to the variables of every member of the network. The contribution of the distributed coordination mechanism in this case is that, through the structure of its dynamical invariant, it provably preserves a certain coarse-level aspect of the centralized dynamics. This enables us to obtain a distributed mechanism which, although not identical to the centralized mechanism, has the same qualitative behavior. The resulting system executes three useful computations: it elects a leader, determines the maximum of a set of positive numbers, and counts the number of nodes on the network.

4.1.1 Clock Synchronization

In this section we examine the problem of determining a single global time among a network of nodes with locally available (but distinct) time signals; specifically, suppose we have a connected graph G, and at each node i the following local time signal:

$$\tau(i,t) = \alpha(i)t + \beta(i).$$

Note that this is merely a parametrization of the local time signals; we do not mean to imply that the $\alpha(i)$ or $\beta(i)$ terms are known locally.

We will apply a variant of $\Phi_L(\cdot)$ with the tracking behavior described in Section 3.4.2, and an additional feedback term. Consider the following dynamics:

$$\begin{aligned} x(i,t) &= \tau(i,t) + \sum_{j\in N_i} \delta(i,j,t), \\ \dot{\delta}(i,j,t) &= (x(j) - x(i)) + v(i,t), \\ \dot{v}(i,t) &= (x(j) - x(i)). \end{aligned}$$

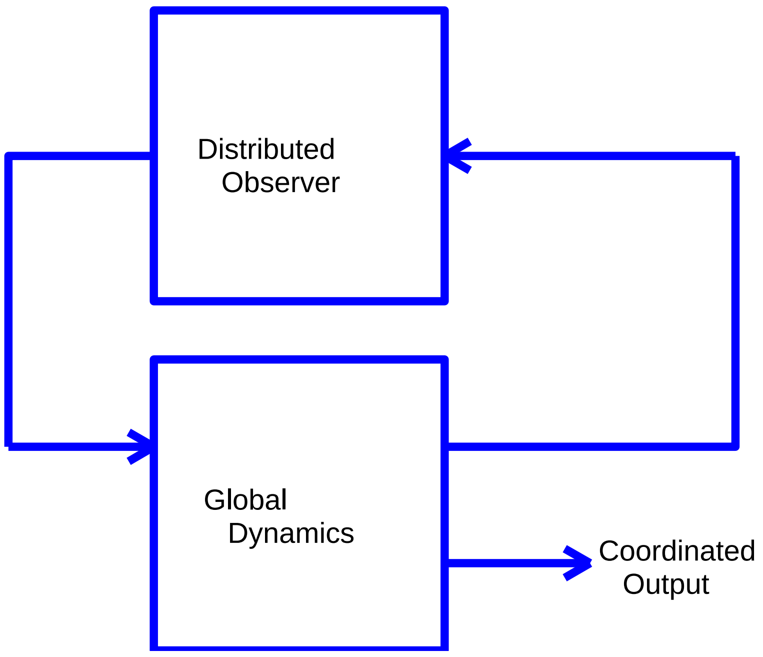


Figure 4.1: Schematic representation of a distributed observer design. A distributed coordination mechanism is used to estimate the evolution of some global variables; when this information is fed back into the global dynamics, a coordinated output arises.

Here we have added an integral feedback term to the dynamics of the flux variables. Since this is a linear time-invariant system, we can analyze its behavior using a matrix transfer function. Computing the transfer function from the $\tau(i,t)$ terms to the $x(i,t)$ variables, we obtain

$$s^2\left(s^2+sL+L\right)^{-1}.$$

Now, since L is symmetric, it can be expressed as a spectral decomposition:

$$L=\sum_i \lambda_i Q_i$$

where the λ_i terms are non-negative real eigenvalues, the Q_i are orthogonal projections onto the associated eigenspaces, and

$$\lambda_0=0, \quad Q_0=\frac{\mathbf{1}_n\mathbf{1}_n^T}{\mathbf{1}_n^T\mathbf{1}_n}. \tag{4.1}$$

Applying this expression, and some algebraic manipulation, we can write the preceding transfer function as:

$$\frac{\mathbf{1}_n\mathbf{1}_n^T}{\mathbf{1}_n^T\mathbf{1}_n}+s^2\left(\sum_{i>0}\frac{1}{s^2+\lambda_i s+\lambda_i}Q_i\right).$$

Recall from our discussion of Laplacian matrices in Chapter 2, that the Laplacian of a connected graph has a nullspace of dimension one, and is positive semi-definite. This implies that the remaining

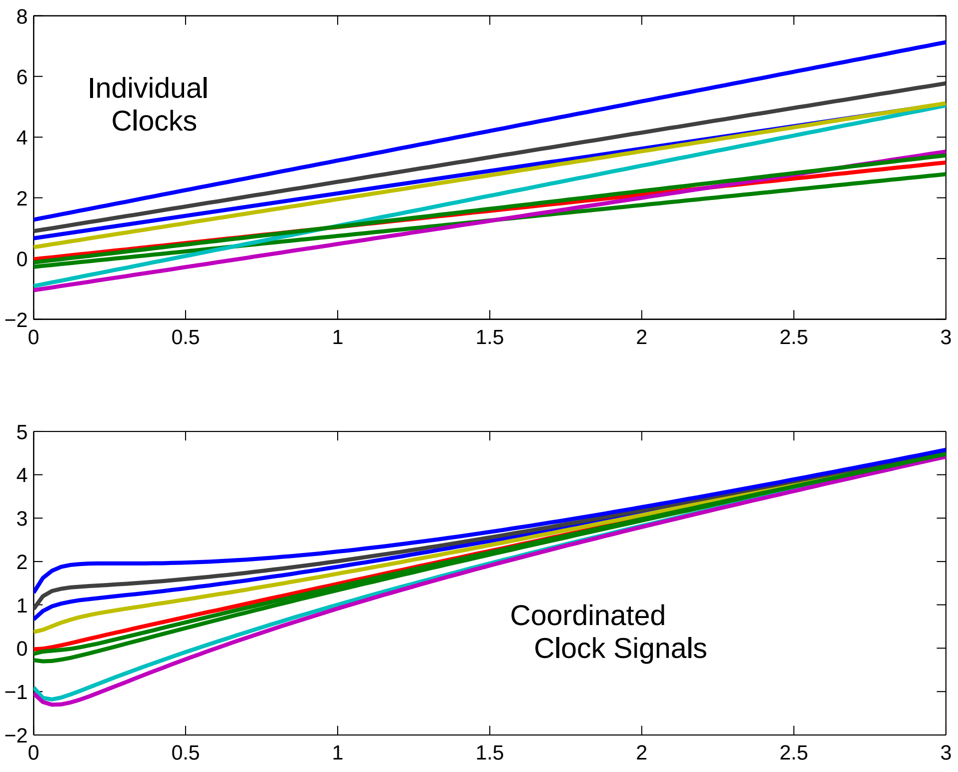


Figure 4.2: Sample trajectories of the clock synchronizing dynamics. Each node has a local time that is independent of all the other clocks, and only constrained to grow linearly in time; the coordinating dynamics drives all of the members of the network to a single global time, independent of the underlying network topology.

λ_i terms are strictly positive, and that the transfer functions in the sums describe *asymptotically stable systems.* We see that the terms in the sum each have two zeros at $s = 0$; from the Final Value Theorem, we know that when applied to a signal with up to two poles at zero, these systems produce outputs that converge to zero exponentially. Now, the Laplace transform of the inputs is:

$$\frac{\alpha(i)s + \beta(i)}{s^2}$$

which has two poles at zero. We thus have:

$$x(i,t) \rightarrow \frac{1}{N} \sum_{i \in G} \left(\alpha(i)t + \beta(i) \right).$$

The implication of this statement is that each node asymptotically achieves an *identical* signal $x(i,t)$; further, this is a “plausible” global clock signal, in that it too is a linearly growing function of time (its parameters are exactly the average parameters of the various local inputs). We have thus exhibited an application of a dynamic coordination mechanism to provide global access to a single clock on an arbitrary unstructured network.

4.1.2 Leader Election, Maximum-Tracking, and Counting

In this section we present a distributed dynamical system that (empirically) accomplishes three useful tasks:

- It selects from among the N nodes in a network a single "leader"; each node unambiguously determines whether or not it is the leader, and also learns the identifier of the leader.
- It selects the largest value from a set of positive numbers, and makes this value available to every member of the network; further *this quantity can be tracked* as a function of time.
- It computes the number of nodes on the network, *without a priori requirements* on network structure, and makes this information available to all members of the network.

The basic mechanism that will power these results is a special case of the Lotka-Volterra dynamics, which is as follows:

$$\dot{x}(i) = x(i)\left(\alpha(i) - \frac{1}{N}\sum_{j\in G} x(i)\right).$$

Here the $\alpha(i)$ terms are distinct positive values; within the context of the Lotka-Volterra population model, they correspond to carrying capacities. These equations have a very special property: for arbitrary initial conditions in which all the $x(i)$ are non-zero, all of the $x(i)$ terms go to zero save the one associated with the largest $\alpha(i)$ term (see Hofbauer and Sigmund [11] for a proof). Intuitively, this indicates that only the "fittest" species survives.

Let us denote the maximum of the $\alpha(i)$ by α_m, and the associated node i_m. Now, from the above property, we see that $x(i_m)$ must converge to

$$x(i_m) = N\alpha_m.$$

Implicitly, the node with the largest α_i indirectly computes the number of nodes on the network, and discovers that its value is indeed the largest.

The system described above suffers from two drawbacks:

- It requires global interconnection for each node to implement its local dynamics.
- It only provides the maximum value and the size of the network at one node.

We will remedy both of these failings by using a distributed mechanism for dynamically estimating the relevant average quantity required in the Lotka-Volterra equations.

Consider then the following dynamics:

$$\begin{aligned}
\dot{x}(i) &= x(i)\left(\alpha(i) - y(i)\right), \\
\dot{y}(i) &= \sum_{j \in N_i} \left(y(j) - y(i)\right) + \dot{x}(i), \\
\dot{z}(i) &= \sum_{j \in N_i} \left(z(j) - z(i)\right) + \frac{d}{dt}\left(\frac{x(i)}{\alpha(i)}\right)^2.
\end{aligned}$$

The first equation is the Lotka-Volterra dynamics; the second and third equations are the average-tracking dynamics for $\frac{1}{N}\sum x(i)$ and $\frac{1}{N}\sum\left(\frac{x(i)}{\alpha(i)}\right)^2$ (we assume that the $y(i)$ and $z(i)$ variables are initialized accordingly). Clearly, if the $y(i)$ terms provided an exact estimate of the average of the $x(i)$ terms, we would reproduce the previous dynamics exactly. While we do not have exact estimates, we do reproduce the following quantity exactly:

$$\begin{aligned}
\frac{d}{dt}\sum_{i \in G} \log x(i) &= \sum_{i \in G} \left(\alpha(i) + y(i)\right) \\
&= \sum_{i \in G} \left(\alpha(i) + x(i)\right)
\end{aligned}$$

where in the second line we have used the invariant $\sum_{i \in G} x(i,t) = \sum_{i \in G} y(i,t)$. Hence, the invariant quantity ensures that the dynamics of the sum of the logarithms *is unchanged by the addition of the distributed observer.* Intuitively, the distributed observer mechanism is sufficiently accurate to ensure that "on the average" the coarse-scale dynamics is the same as in the centralized implementation.

The global stability of the proposed dynamics is a complex question in non-linear dynamics; similar systems are treated in detail in [11], and it is not our goal to examine such questions. Simulation studies seem to bear out its global stability, but we have been unable to find an analytical tool that can prove this to be true.

However, we will now show that *given* that there is a globally-asymptotically-stable non-zero equilibrium, then it is the desired one.

Lemma 9 (Leader Election and Counting). *Suppose there exists a non-zero globally-asymptotically-stable equilibrium of the dynamics with the distributed observer; then it has the following properties:*

1. *The $y(i)$ variables converge to the maximum of the $\alpha(i)$ terms.*

2. *The $z(i)$ terms converge to the number of nodes on the network.*

Proof. In equilibrium, all the $y(i)$ variables must be identical. Stability of the equilibrium implies that this value must be bounded above by the maximum of the $\alpha(i)$ variables, else all the $x(i)$ variables would only be stable at zero, and hence the $y(i)$ and $z(i)$ variables would also de facto be zero because they track averages of the $x(i)$ and $x^2(i)$ terms. However, the $y(i)$ variables must also

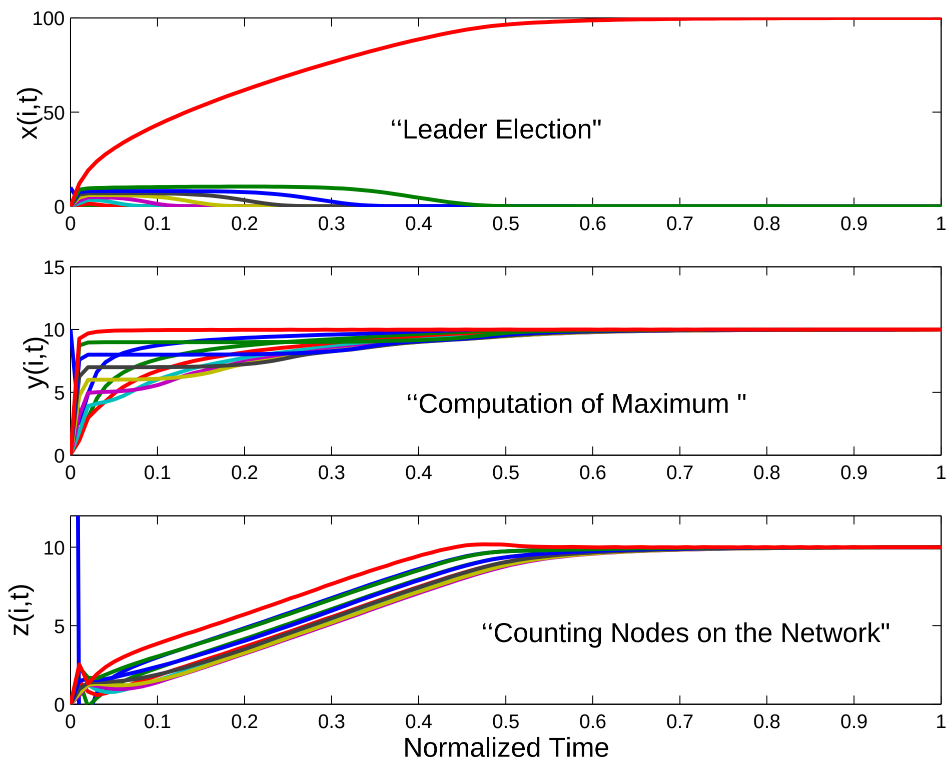


Figure 4.3: Illustration of the leader-election/counting dynamics with the distributed observer. The top plot shows the dynamics of the “activity” states; asymptotically exactly one node has a nonzero value. The middle plot shows the estimates of the “leader’s” unique identifier (the nodes are numbered from one to ten in this example). The bottom plot shows the estimates of the number of nodes on the network.

be bounded below by the same maximum $\alpha(i)$ value, else the $x(i)$ with the highest $\alpha(i)$ would not be stable at any non-zero value. Thus, the $y(i)$ variables must all be equal to the largest $\alpha(i)$. This in turn implies that the corresponding $x(i)$ must be non-zero, else it would be unstable.

We thus have a single non-zero $x(i)$, and we know that the average of the $x(i)$ terms is equal to the maximum $\alpha(i)$. This implies that the remaining non-zero $x(i)$ is equal to the product of its own $\alpha(i)$, and the number of nodes on the network N. Now, all we need to note is that the $z(i)$ variables track the average of the terms $(x(i)\alpha(i))^2$. Knowing that exactly one $x(i)$ is non-zero, and having previously derived its value, we now know that this average is exactly N. □

One final point to note is that this system naturally tracks variations in the $\alpha(i)$ terms without any modification.

So, we have presented a system using the distributed averaging dynamics as an observer for the global Lotka-Volterra dynamics, and shown (assuming stability) that this system recovers the coordinated behavior of the centralized implementation.

4.2 Distributed Data Processing

In this section we will present two examples of *distributed data processing*, by which we mean a situation in which data from some large data set are spread over multiple computational units connected over an unstructured network. Our goal is to perform computations that yield information about the global data set, without ever requiring access to non-local information in the network. We should emphasize that this is in itself not a novel pursuit; see for example Haykin [9] for an introductory text from the artificial intelligence community. Our goal is to show designs based on our distributed-gradient formalism that execute these useful computations.

The two "design principles" we wish to demonstrate here are parallel interconnections and cascades of a distributed coordination mechanism. Section 4.2.2 will show how to use multiple parallel coordination mechanisms to carry out principal component analysis and solve least-squares problems. Section 4.2.1 will show a cascade mechanism for decorrelating statistically dependent data (which also produces a mechanism for distributed QR factorization).

Throughout this section we will focus on data vectors from $\mathbf{R}^2$ and $\mathbf{R}^3$ to make the exposition as concrete as possible, but the mechanisms provided easily generalize to higher-order systems that execute the associated operations for data vectors of arbitrary dimension.

4.2.1 Principal Components and Least Squares Models

In this section we will show a very simple mechanism for principal components analysis and for linear least-squares problems. We will use three parallel executions of the distributed averaging system to locally synthesize the global principal matrix (or normal equations), which will then allow each node to perform a local computation on a problem of *fixed size* to obtain the globally optimal solution, independent of the size of the network.

Let us suppose that we have a connected graph G with N nodes, and each node i has access to some vector

$$\begin{pmatrix} \alpha(i) \\ \beta(i) \end{pmatrix} = \mathbf{w}(i) \in \mathbf{R}^2.$$

Let us suppose that the $\alpha(i)$ and $\beta(i)$ terms have zero mean for simplicity (the mean can clearly be subtracted using the distributed averaging system). Organizing all the $\mathbf{w}(i)$ terms as *rows* in the following data matrix:

$$A = \begin{pmatrix} \mathbf{w}(1)^T \\ \mathbf{w}(2)^T \\ \vdots \\ \mathbf{w}(N)^T \end{pmatrix}.$$

The *principal matrix*, or sample covariance matrix (see any textbook on statistical data processing,

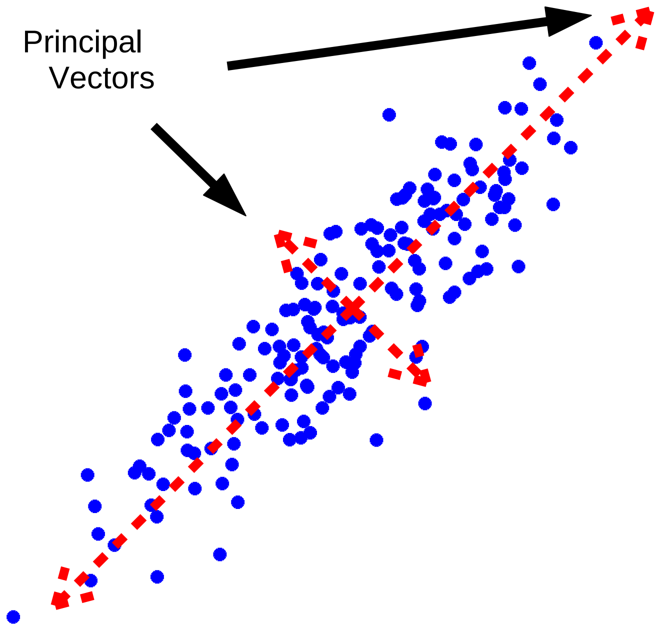


Figure 4.4: Illustration of principal vectors for multivariable Gaussian data. The equations for the principal vectors are of constant size as a function of data set size, and the algorithm proposed obtains at each node a copy of the principal matrix from which to compute the principal vectors.

e.g. Haykin [9]), is defined by:

$$A^T A = \sum_{i \in G} \begin{pmatrix} \alpha(i)^2 & \alpha(i)\beta(i) \\ \alpha(i)\beta(i) & \beta(i)^2 \end{pmatrix}.$$

This is a symmetric positive-semidefinite matrix, and its eigenvectors are known as *principal vectors.* This matrix is used, for example, in techniques for low-dimensional approximation of data. Intuitively, the principal vector with the *largest* associated eigenvalue is the one which captures "most" of the information about a "typical" data vector. It is very easy to obtain a distributed solution to this problem; three parallel copies of the distributed averaging system are run for the quantities $\alpha(i)^2$, $\beta(i)^2$, and $\alpha(i)\beta(i)$. This allows each node to obtain a local copy of

$$\frac{1}{N} \begin{pmatrix} \alpha(i)^2 & \alpha(i)\beta(i) \\ \alpha(i)\beta(i) & \beta(i)^2 \end{pmatrix}$$

Since the eigenvectors of a matrix are invariant under rescaling of the matrix, this information suffices to compute the principal vectors at each node.

As an immediate extension of this application, we can compute least-squares solutions to systems of linear equations. To do so, we begin with a system of linear equations:

$$A\mathbf{c} = \mathbf{b}.$$

Suppose that each row is of the form

$$c_1\alpha(i) + c_2\beta(i) = b(i),$$

which simply means that each node has some observation $b(i)$ which it wishes to explain with a linear combination of its explanatory variables $\alpha(i)$ and $\beta(i)$. In order to obtain the best possible estimate, we wish to compute the least-squares solution to the (in general) overdetermined system of equations above. To do so, we form the normal equations (this formulation is convenient but there are of course many others):

$$A^T A\mathbf{c} = A^T\mathbf{b}.$$

Now, expanding $A^T\mathbf{b}$, we find it is just:

$$\begin{pmatrix} \sum_{i\in G} \alpha(i)b(i) \\ \sum_{i\in G} \beta(i)b(i) \end{pmatrix}.$$

Applying the same technique as for the principal matrix, we can obtain both sides of the following equation:

$$\frac{1}{N}A^T A\mathbf{c} = \frac{1}{N}A^T\mathbf{b}.$$

Clearly, the solution to this set of equations is the same as that of the normal equations, and we can assemble these equations locally at each node; all that remains to be done is a local solution to a linear system of equations. It should be clear that this generalizes systematically to arbitrary dimensions; each term in the desired matrices and vectors is a scaled inner product, which can easily be computed by appropriate application of the averaging system.

The two techniques we have presented are admittedly very simple, but serve to demonstrate two points. First, we see that a distributed coordination tool can be applied to produce general linear-algebraic model building in a distributed implementation; each node contributes its local data, and gets a copy of the globally optimal answer. Second, these designs demonstrate a design mechanism in which multiple *parallel* coordination processes which are then *coupled* at output by an appropriate nonlinear mapping (eigenvector computation or matrix inversion). This situation is illustrated in Figure 4.5.

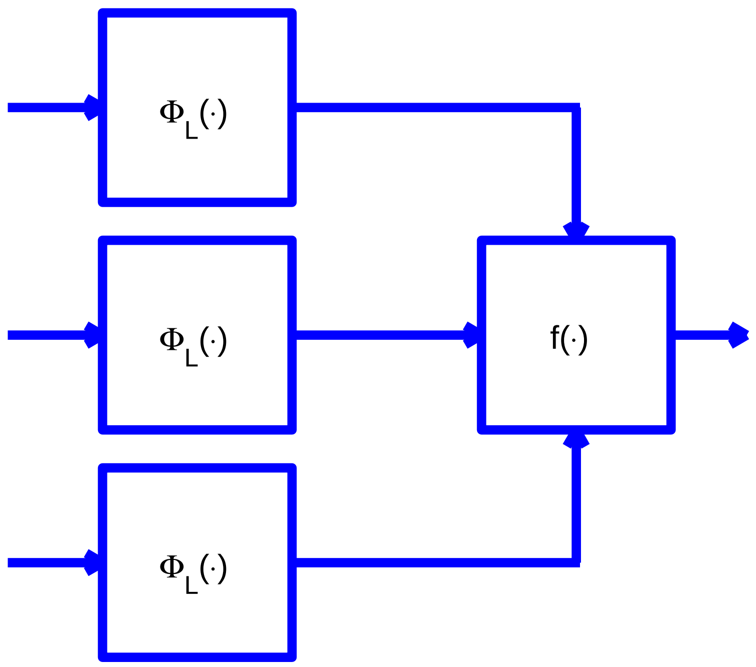


Figure 4.5: A parallel coordination architecture with a nonlinear output on the coordinated variables; this system accomplishes distributed principal components analysis and least-squares model building.

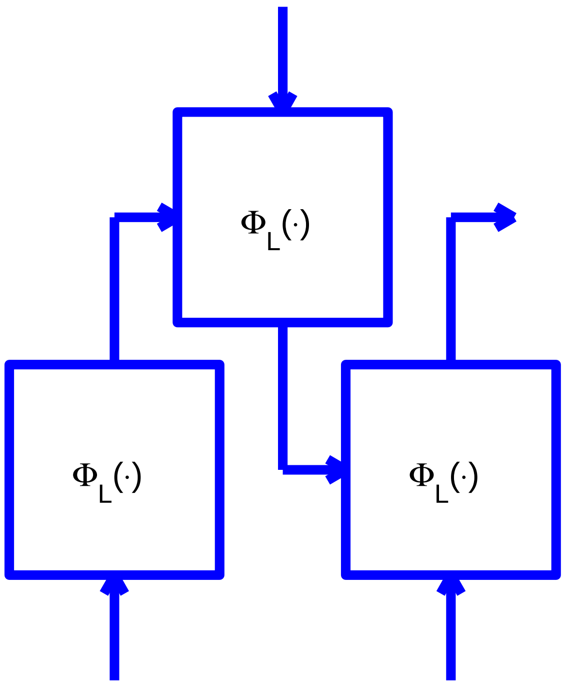


Figure 4.6: A cascade coordination architecture representing a recursively decomposed computation; this system accomplishes distributed decorrelation and QR factorization.

4.2.2 Decorrelation and QR Factorization

In this section we will present a Gram-Schmidt mechanism for orthogonalizing a collection of data vectors; in a statistical interpretation, this will "decorrelate" the components of the data vectors so that in the transformed sample their sample correlations will be zero. In a linear algebraic interpretation, this is precisely QR factorization of the matrix whose rows are represented by the data vectors. The mechanism by which we will achieve this is a cascade of two copies of the distributed averaging mechanism $\Phi_L(\cdot)$.

Let us suppose that we have a connected graph G, and each node i has access to some vector of observations of zero-mean variables,

$$\begin{pmatrix} \alpha(i) \\ \beta(i) \\ \gamma(i) \end{pmatrix} = \mathbf{w}(i) \in \mathbf{R}^3.$$

We would like to find a transformation matrix $R \in \mathbf{R}^{3\times 3}$ such that the three components of $R\mathbf{w}(i)$ are empirically uncorrelated across the data set. The standard solution, of course, is to project one of the variables (say $\beta(i)$) onto the other, and subtract the resulting "correlation term". Specifically, for two random variables p and q we can obtain a pair of uncorrelated variables as follows:

$$\left\{ p, q - \frac{E(pq)}{E^2(p)} p \right\}$$

This corresponds to an R matrix of:

$$R = \begin{pmatrix} 1 & -\frac{E(pq)}{E^2(p)} \\ 0 & 1 \end{pmatrix}$$

Applying this idea, we can first empirically decorrelate the $\alpha(i)$ and $\beta(i)$ by computing

$$R(1,2) = \left(\frac{\sum \alpha(i)\beta(i)}{N} \right) \left(\frac{N}{\sum \alpha(i)^2} \right),$$

and subtracting $R(1,2)\alpha(i)$ from $\beta(i)$; let us call this new decorrelated variable $\tilde{\beta}(i)$. As is shown in the equation, we can implement this computation with parallel averaging systems. In order to decorrelate $\gamma(i)$, we must repeat the above operation on $\gamma(i)$ for *both* $\alpha(i)$ and $\tilde{\beta}(i)$; this represents a second layer of averaging computations. This recursive decomposition of the decorrelation process leads to a natural *cascade* implementation with the distributed averaging dynamics; this is schematically represented in Figure 4.6.

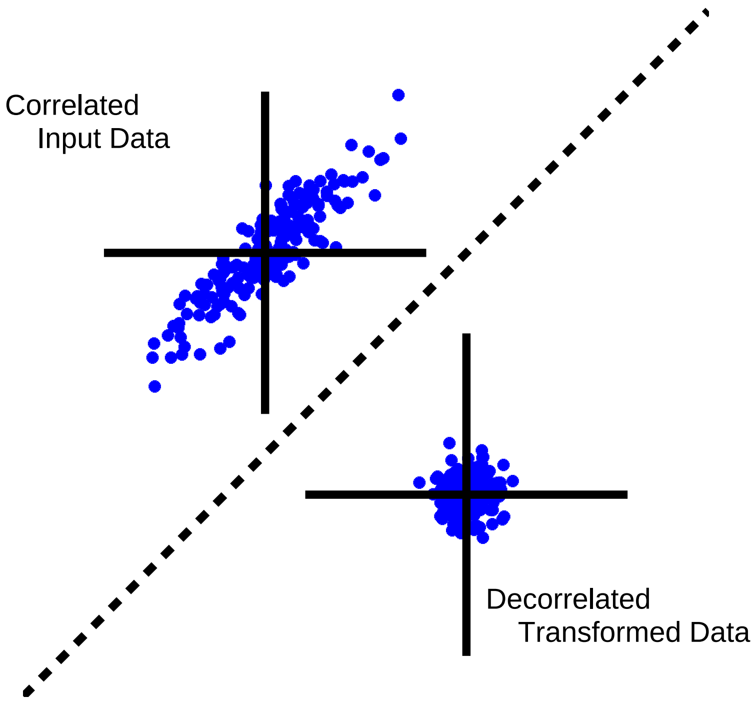


Figure 4.7: Decorrelation of Sample Data

4.3 Topological Coordination

In this section we present some new applications of dynamic coordination which are associated with the interconnection structure of the network on which they are implemented. These mechanisms exercise the new tools developed for designing distributed coordination mechanisms.

Section 4.3.1 demonstrates a distributed gradient system for an unconstrained problem, but imposes constraints in the interaction set that model "small groups of nodes acting together". The resulting dynamics will provide algorithms for graph coloring (exact for 2-coloring, heuristic for 3-coloring). Interestingly, the only difference between the 2-coloring and 3-coloring dynamics will be a change in the interaction sets.

Section 4.3.2 shows a distributed gradient system with a "nonlinear" coordination, and applies this to determining the extremal eigenvectors of various matrices associated with a graph. The resulting nonlinear dynamics is quite complex, and without the distributed gradient mechanism a convergence analysis would be difficult (as would characterization of equilibrium). This demonstrates the formal power of the method for synthesizing complicated nonlinear dynamics, and providing a succinct and powerful analysis tool for these systems.

Finally, Section 4.3.3 revisits our motivational "distributed accounting" problem, in which a group with multiple debts attempts to restructure its obligations so that the total outstanding payment is reduced, while ensuring "fairness", in the sense that no one is required to pay more or less (in total) than they owe. This illustrates the use of our modeling formalism in a less traditional scenario, and hopefully indicates some of its flexibility for future applications.

4.3.1 Graph Two-Coloring and Three-Coloring

Here we consider the problem of "graph-coloring", which is a prototypical algorithmic problem involving graphs. We will present an "exact" solution for the case of two-coloring (which is well known to be a tractable problem), and a heuristic mechanism for three-coloring (which is a classic NP-complete computation, see Papadimitriou and Steiglitz [20]).

The problem of graph coloring is as follows: given a graph G, and a fixed set of "color" labels, say "red" and "blue", assign to each node a color such that *every* node's color is distinct from its neighbors' colors. For any particular graph G, such an assignment may or may not exist; if one does exist, we would like to find it, and if no coloring exists we would like to obtain proof that it does not.

The two-coloring mechanism we will present will encode the coloring information in the *sign* of a real-valued state variable. Let us assume that G is a connected graph, with node states $x(i)$. The optimization which will specify our problem is simply the unconstrained problem

$$\min \sum_{i \in G} x(i)^2,$$

but we will impose a special interaction structure. Specifically, we will consider the following interaction set:

$$\left\{ \frac{1}{2} \left(\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_i \otimes \mathbf{e}_j + \mathbf{e}_j \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j \right) \;\middle|\; (i,j) \in G \right\}.$$

Intuitively, this interaction set allows *pairs* of neighbors to move identically; in the $\{\mathbf{e}_i, \mathbf{e}_j\}$ basis, the matrix representing this interaction is

$$\frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Assembling the distributed gradient dynamics, we obtain

$$\dot{x}(i) = -\sum_{j \in N_i} (x(i) + x(j)).$$

Now, since this is a distributed gradient system, we know *a priori* that it goes to a limit, and that in the limit every interaction produces a zero tangent vector. In particular this means that in equilibrium

$$x(i) = -x(j) \text{ for all } (i,j) \in G.$$

This implies that there are at most two possible values of the various $x(i)$ on the network; if they are non-zero then we have produced an encoding of a two-coloring in the sign of each $x(i)$. The only question that remains is whether the equilibrium is non-zero.

If the graph G is not two-colorable, then it is impossible to assign two opposite values to the nodes of the graph such that the above is satisfied; hence the equilibrium in this case must be zero. If the graph *is* two colorable, then there exists an assignment of values from $\{-1, 1\}$ to each node in the graph, call it $c(i)$, such that $c(i) = -c(j)$ for every edge in G. Consider then, the following:

$$\frac{d}{dt}\left(\sum_{i\in G} c(i)x(i)\right) = 0.$$

This can readily be deduced from the fact that each interaction produces a tangent vector that respects this invariant. This implies that there is a *non-zero linear invariant* in our graph-coloring system, and so the system will converge to the intersection of this manifold with the equilibrium manifold described above, producing a non-zero equilibrium state in which the color of each node is encoded in the sign of the corresponding $x(i)$. This is guaranteed to be non-zero "generically"; it is of course possible to pick an initial value for which the resulting equilibrium state will be at the origin, but the set of values which cause this has measure zero in $\mathbf{R}^n$.

Peripherally, we should now mention that this system can be interpreted as a potential system. Specifically, it arises from the pairwise potential

$$f(x, y) = (x + y)^2\,.$$

We indeed see that this accomplishes "reflection alignment", as indicated in Section 3.1; the only possible non-zero equilibria correspond to neighbors' states reflected across the point $x = 0$.

Now, to examine three-coloring, let us consider a new interaction set; in order to define it, we introduce the following set $T(G)$:

$$T(G) = \{(i, j, k) \mid (i, j),\ (j, k),\ (k, i) \in G\}\,.$$

This is the set of all "triangles" in T. The interaction set we wish to define is as follows:

$$\left\{ \sum_{p,q=i,j,k} \mathbf{e}_p \otimes \mathbf{e}_q \,\middle|\, (i, j, k) \in T(G) \right\}.$$

each interaction represents the collective action of a group of three mutually interconnected neigh-

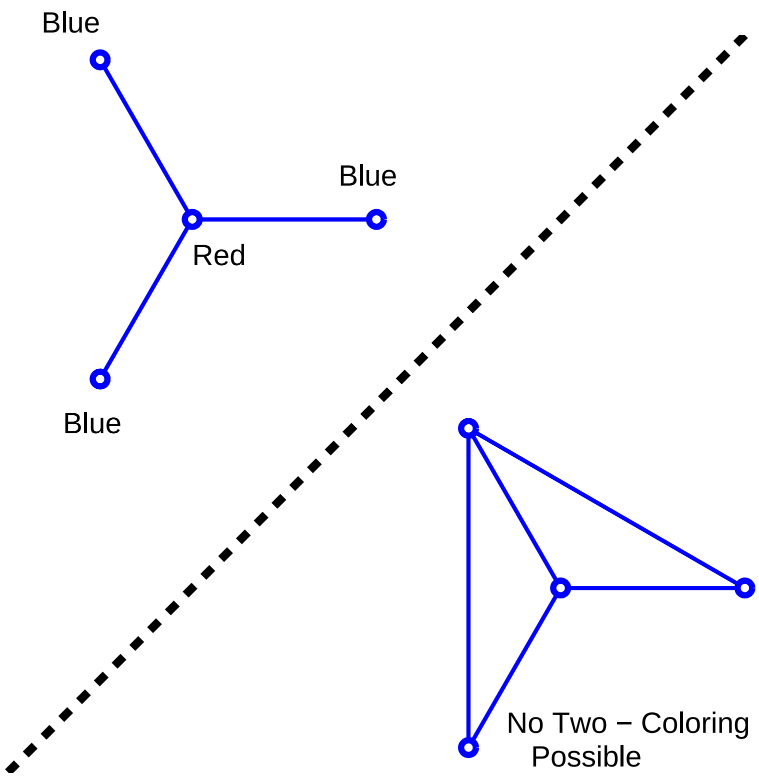


Figure 4.8: Two coloring for a pair of graphs. One graph is two-colorable, and this coloring is indicated by the color labels next to the nodes. The other graph does not admit any two coloring. The coordination mechanism proposed provides a solution when one exists, and converges globally to zero when no two-coloring exists (see Figure 4.9).

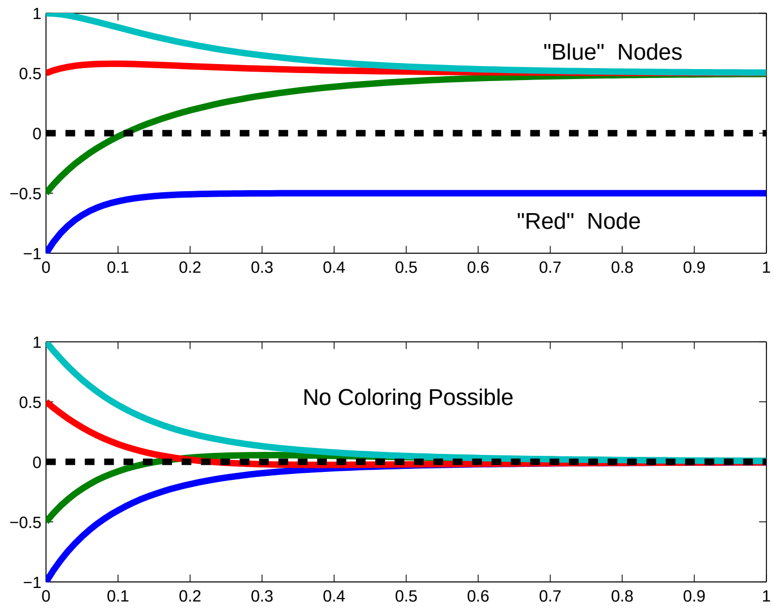


Figure 4.9: The dynamics of our two coloring mechanism for the graphs shown in Figure 4.8; for the two colorable graph, the “red” node converges to a negative value, and the “blue” nodes converge to the opposite value. For the graph with no admissible coloring, all states converge to zero.

bors. To see this, consider the matrix representation of the above interaction in the $\{\mathbf{e}_i, \mathbf{e}_j\mathbf{e}_k\}$,

$$\frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Clearly, every tangent vector produced by this interaction modifies the states of the three members of the triangle equally.

At this point let us introduce an additional assumption about G, that it is "triangularly connected".

Definition 30 (Triangular Connectedness). *Let G be a connected undirected graph. We will say that G is* triangularly connected *if it has the following three properties:*

1. *G is connected.*
2. *For every edge $(i, j) \in G$, i and j occur as neighbors in some triangle (i, j, k) in $T(G)$.*
3. *For every two triangles (i, j, k) and (i, p, q) in $T(G)$ sharing a node i, then (i, j, k) and (i, p, q) also share an edge, e.g. $(i, j) = (i, p)$.*

Lemma 10 (Triangular Connectedness Implies "Uniqueness" of Three Coloring). *Let G be a graph that is triangularly connected. Then, up to renaming of colors, there is at most one three coloring of G.*

The proof is a straightforward exercise and omitted. Now, considering the distributed gradient dynamics, and again applying our characterization of equilibrium from Chapter 3, we see that for each triangle the equilibrium states satisfy:

$$x(i) + x(j) + x(k) = 0.$$

This means that for each triangle there exists some non-positive value α, some non-negative value β, and some third value γ satisfying $\alpha \leq \gamma \leq \beta$, such that the $x(i)$, $x(j)$, and $x(k)$ variables each take one of these values. Moreover, the triangular connectedness requirement implies that the values α, β, γ are the *same* in each triangle on the network. To see this, consider assigning any two in some triangle T_1; this clearly specifies the third value. Now, every triangle adjacent to T_1 shares an edge, and hence two node values with T_1; this in turn sets the third value in each of the adjacent triangles. The triangular connectedness requirement implies that this in fact specifies all the values on the entire network. The only question that remains is whether these values can be non-zero.

Now, there can be a non-zero solution to these equations if and only if G is three-colorable; clearly, if the graph is not three-colorable, no such solution can exist. If G *is* three-colorable, then there exists an assignment of the values $\{-1, 0, 1\}$, call it $c(i)$, satisfying the above conditions. Further, if such an assignment exists, then we have the following invariant quantity in the dynamics:

$$\frac{d}{dt}\left(\sum_{i \in G} c(i)x(i)\right) = 0.$$

To verify this, simply note that every interaction satisfies this property. Again, this invariant implies that for generic initial values, the states converge to a non-zero equilibrium. Encoding the color of the node by it being the largest, smallest, or intermediate value among the three values in the triangle, we can convert these equilibrium states into a three-coloring.

We have thus shown that for triangularly connected graphs, our algorithm solves the three-coloring problem from generic initial values. This is "unsurprising", as a greedy algorithm easily solves such instances of three coloring. For general graphs, we do not have a good characterization of the system's performance; it is not our goal to treat intricacies of graph coloring, and so we will omit further discussion of this topic. The point we wish to emphasize is the simplicity of the analysis provided by the distributed gradient formalism, and the "naturalness" of the transition between the two-coloring and three-coloring formulations (parametrized only by the allowable set of interactions).

4.3.2 Graph Eigenvectors

In this section we consider a significantly more difficult problem with a nonlinear coordination; specifically, we will constrain our system to evolve on a sphere.

Let us suppose we have some symmetric matrix associated with the edges of a connected graph G, say $M(G)$; the adjacency matrix or the Laplacian matrix could be considered, for example. Specifically, we assume that $M_{ij}(G)$ is some real-valued weight on the edge (i, j) in G, and that $M_{ij}(G) = 0$ if (i, j) is not in G. This matrix has some set of eigenvectors, with corresponding eigenvalues; suppose now that we wish to compute an extremal eigenvector of this matrix (say the one corresponding to the largest eigenvalue). This corresponds to the following optimization:

$$\max_{\|\mathbf{x}\|=c} \frac{\mathbf{x}^T M(G)\mathbf{x}}{\mathbf{x}^T\mathbf{x}}.$$

Let us consider the standard pairwise interaction set $\mathcal{P}_E(G)$

$$\{\mathbf{e}_i \otimes \mathbf{e}_i + \mathbf{e}_j \otimes \mathbf{e}_j \mid (i, j) \in G\}$$

Now, we wish to coordinate this interaction set so that it respects the fixed-norm constraint of the eigenvector optimization problem. We know from Chapter 3 that this interaction has coordination

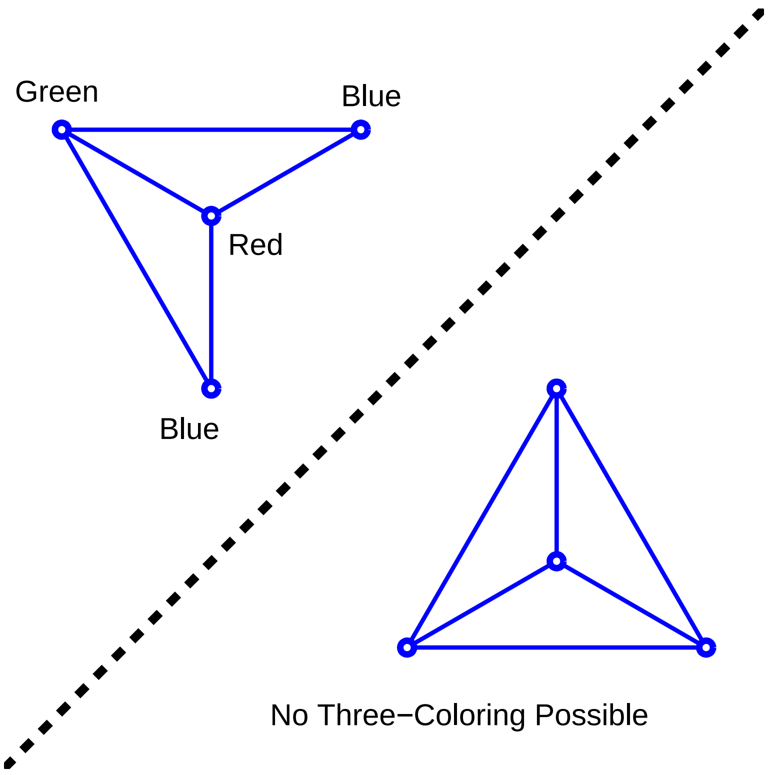


Figure 4.10: Three coloring for a pair of graphs. One graph is three-colorable, and this coloring is indicated by the color labels next to the nodes. The other graph does not admit any three coloring. A small modification to our two-coloring system can serve as a heuristic for three coloring, which solves a small class of problems exactyl (see Figure 4.11).

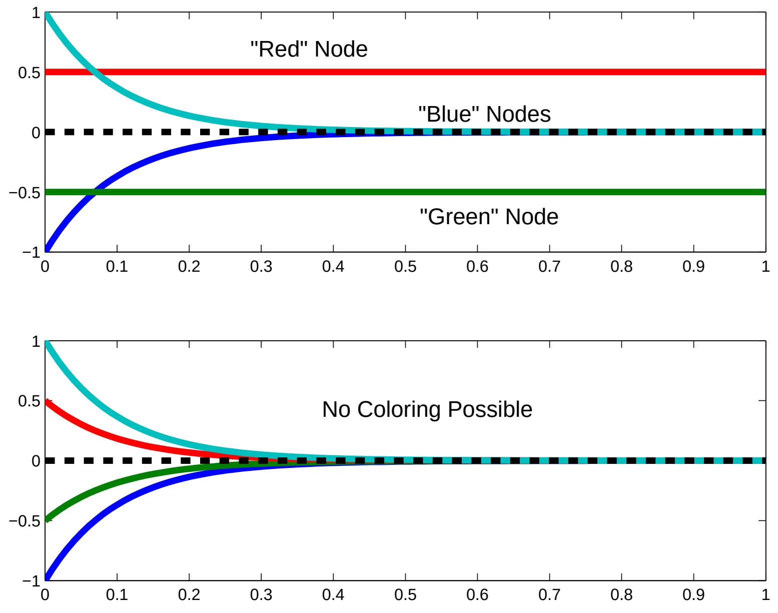


Figure 4.11: The dynamics of our three coloring mechanism for the graphs shown in Figure 4.10; as specified in the text, the three-coloring is encoded in the asymptotic state being positive, negative, or zero. For the graph that admits no coloring, we see that all the states converge exponentially to zero.

capacity 1, and since the norm constraint represents a codimension 1 coordination we know *a priori* that the resulting dynamics will *weakly* solve the system (we will return to strong solvability in a moment).

The coordinated interaction set can be calculated to be:

$$\left\{ \frac{1}{x(i)^2 + x(j)^2} \left(x(j)^2 \mathbf{e}_i \otimes \mathbf{e}_i + x(i)^2 \mathbf{e}_j \otimes \mathbf{e}_j - x(i)x(j) \mathbf{e}_i \otimes \mathbf{e}_j - x(i)x(j) \mathbf{e}_j \otimes \mathbf{e}_i \right) \Big| \, (i,j) \in G \right\}$$

This expression is rather intimidating, but it is somewhat more clear in the $\{\mathbf{e}_i, \mathbf{e}_j\}$ basis:

$$\frac{1}{x(i)^2 + x(j)^2} \left(\begin{array}{cc} x(j)^2 & -x(i)x(j) \\ -x(i)x(j) & x(j)^2 \end{array} \right).$$

Note that each such interaction annihilates the gradient of the constraint equation as desired.

To write the distributed gradient dynamics, let us introduce the notation $y(i)$ for the $i-th$ component of $M(G)\mathbf{x}$. Writing out the dynamics, we obtain:

$$\dot{x}(i) = \sum_{j \in N_i} \frac{y(i)x(j)^2 - y(j)x(i)x(j)}{x(i)^2 + x(j)^2}.$$

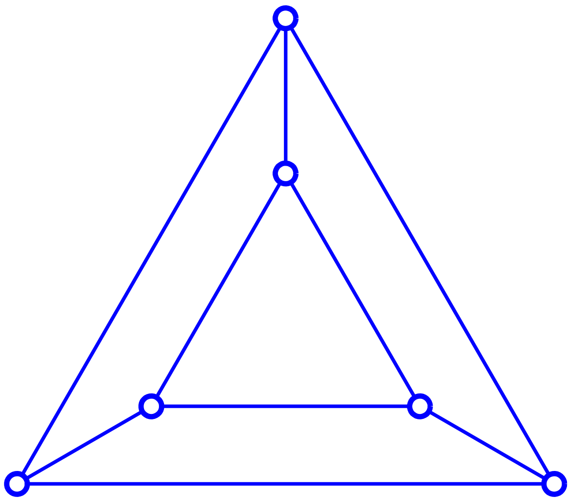
Note the complexity of the dynamics we have synthesized; without knowing the mechanism that generated this system, it seems that it would be extremely difficult to understand its dynamical behavior. For example, it is not obvious from the equations how (or if) the graph structure plays a role in the equilibrium, nor is it clear that it will go to an equilibrium. Nonetheless, we know that this will indeed go to an equilibrium, and that equilibrium will be a distributed extremum relative to the coordinated interaction. Specifically, we will have

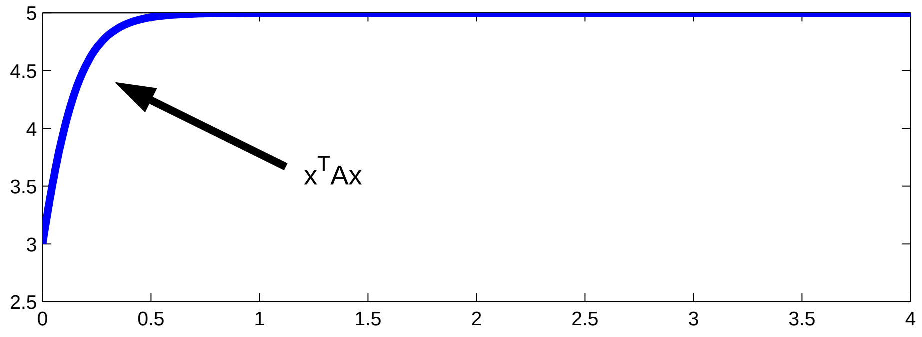
$$y(i)x(j) = y(j)x(i)$$

for all $(i, j) \in G$. It seems we have nearly proven that this does indeed strongly solve the optimization (recall that strongly solving an optimization means that every distributed extremum is also an extremum of the centralized optimization); to see this, suppose all the $x(i)$ terms are nonzero. Dividing through in the above equation, and writing out $y(i)$ explicitly, we obtain:

$$\frac{\mathbf{e}_i^T M(G)\mathbf{x}}{x(i)} = \frac{\mathbf{e}_j^T M(G)\mathbf{x}}{x(j)}.$$

This equation says that each component of the $\mathbf{x}$ vector is *scaled identically* by the action of $M(G)$; hence, $\mathbf{x}$ must be an eigenvector. Unfortunately, it is possible to generate spurious distributed extrema in which some of the $x(i)$ are zero; this interaction set does *not* strongly solve the general





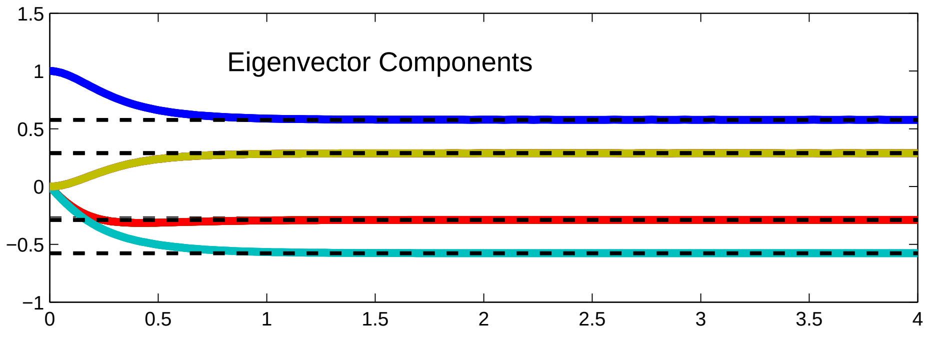


Figure 4.12: Distributed graph eigenvector dynamics. The top figure shows a graph for whose Laplacian matrix we will compute the maximum eigenvalue and associated eigenvector. The center figure shows the trajectory of the objective function dynamics, and the bottom figure shows the evolution of the individual node states; the dashed lines show the exact values of the desired eigenvector (two of the node trajectories are nearly identical and are not distinguishable in the plots). Note that as the limiting values are all non-zero, we have proof that this distributed extremum is in fact a centralized extremum (and hence indeed an eigenvector).

graph eigenvector problem. On the other hand, if one obtains a strictly non-zero solution (which can be easily verified using a distributed mechanism) then it is provably an eigenvector of $M(G)$.

4.3.3 Distributed Accounting

In this section we return to the motivating example with which we began the text, "distributed accounting". We suppose we have a collection of nodes, each owing or being owed some amount of money from its immediate neighbors in some connected graph G. We will encode these debts as a collection of variables $x(i,j)$[1] with $i < j$; each $x(i,j)$ indicates how much is owed *by* i *to* j (this quantity can be negative, indicating that j in fact owes some money to i). Each node then has net payment that it must make,

$$p(i) = \sum_{j>i} x(i,j) - \sum_{j<i} x(j,i).$$

Now, it is possible that the "status quo" is suboptimal, in the sense that debt could be rearranged in a way that maintains everyone's net payment while reducing some metric of total outstanding debt; we will work with the mean-square outstanding debt,

$$\sum_{(i,j)\in G} x(i,j)^2$$

but this choice is mostly inconsequential for the upcoming development. Our formal specification is thus:

$$\begin{array}{cc} \min & \sum_{(i,j)\in G} x(i,j)^2 \\ \text{s.t.} & \sum_{j>i} x(i,j) - \sum_{j<i} x(j,i) = p(i) \\ & \text{for all } i \in G. \end{array}$$

Now, we must ask ourselves what interaction is sensible for this model. Clearly, no interaction involving only one or two edges can even weakly solve this problem; it is simply impossible to respect the fixed net-payment constraint. The simplest interaction that has some hope of solving the problem is the "triangle" interaction: each triangle $(i,j,k) \in T(G)$ has three associated edge variables ("debts") available to it: $x(i,j)$, $x(i,k)$, and $x(j,k)$ (we have assumed that the i,j,k are listed from lowest to highest index). Denoting by $\mathbf{e}_{ij}$ the unit vector modifying the $x(i,j)$ variable in the global state vector $\mathbf{x}$, we have the following interaction set:

$$\{\mathbf{e}_{ij} \otimes \mathbf{e}_{ij} + \mathbf{e}_{ik} \otimes \mathbf{e}_{ik} + \mathbf{e}_{kj} \otimes \mathbf{e}_{kj} \mid i < j < k,\ (i,j,k) \in T(G)\}$$

[1]In this description the state variables are associated with edges and not nodes, which technically breaks with our convention for distributed dynamical systems. It is possible to define another graph in which the states are indeed associated to nodes instead of edges; this is merely notational trickery however, and we will dispense with the formality.

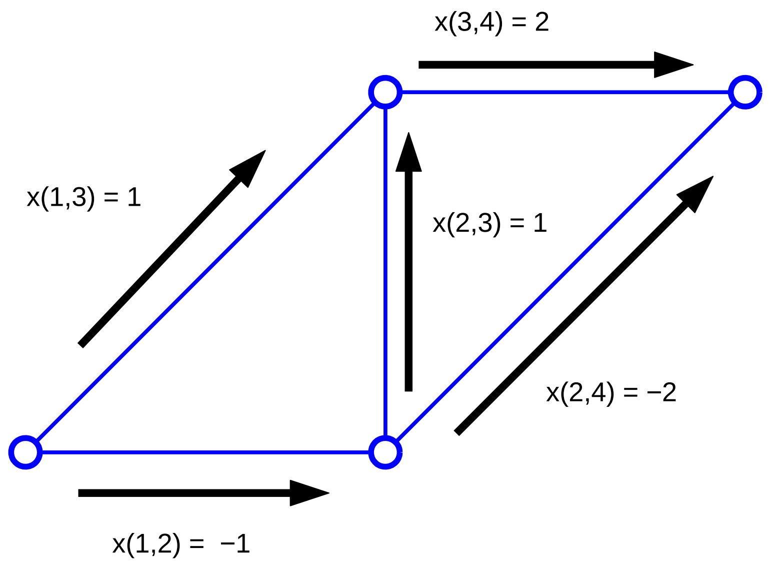


Figure 4.13: Illustration of the Distributed Accounting Problem. Each node owes (or is owed) some amount of money, indicated by the arrows alongside the edges in the network. The goal of the distributed coordination system is to reduce some metric of total outstanding social debt, while maintaining every node's net payment.

The "coordinate-free" expressions for the interaction set are somewhat tedious, so we will not present them. However, in the $\{\mathbf{e}_{ij}, \mathbf{e}_{ik}, \mathbf{e}_{jk}\}$ basis, the coordinated interactions are fairly intuitive and have the following matrix representation:

$$\frac{1}{3}\begin{pmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}.$$

To obtain some intuition about this coordinated interaction, consider an arbitrary vector in its range

$$\begin{pmatrix} \alpha \\ -\alpha \\ \alpha \end{pmatrix}$$

with $\alpha > 0$. This vector indicates an increase in $x(i, j)$, the debt owed from i to j. To compensate this increase $x(i, k)$ is *decreased*, to maintain i's balance of payments. Similarly, $x(j, k)$ is *increased* to maintain the balance of payments at j and k. Now, it is clear that the coordinated interaction weakly solves the optimization specification; the question of conditions under which it strongly solves the specification is somewhat complex and relies on assumptions about the graph structure. For example, the graph in Figure 4.13 has five debt variables, and three linearly independent payment balance constraints (balancing payments for any three nodes implies balanced payments for the

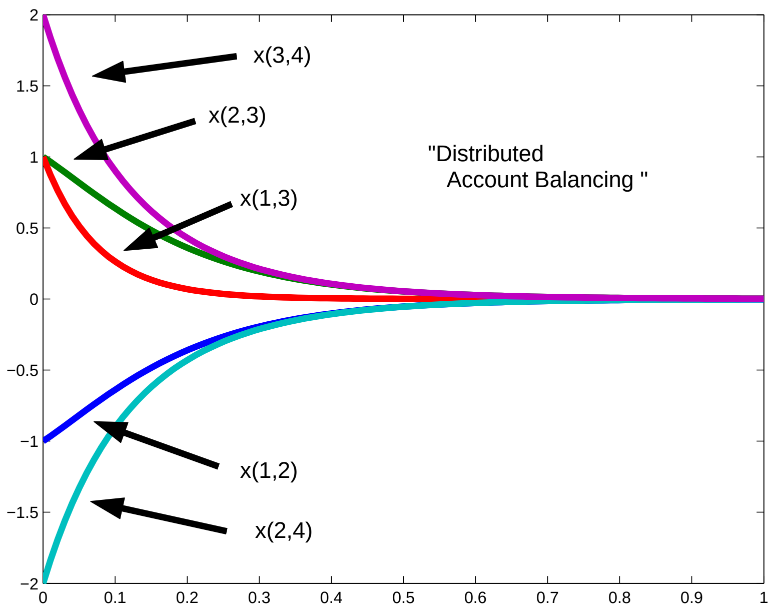


Figure 4.14: Sample trajectories for distributed accounting dynamics; the trajectories shown correspond to the network and initial debts shown in Figure 4.13. Each node's net payment is constant (zero, in this case) across time, but the total outstanding debt is monotonically reduced. In this case, the system recovers the globally optimal solution (in which no payments remain outstanding); for a general graph, the resulting performance depends strongly on how the nodes are interconnected.

fourth). There are two triangles, producing two linearly independent vector fields, and so the distribution operator has rank two. Thus, any vector in the tangent space can be reached by the distribution operator, and so the problem is strongly solved for this special case. We will not attempt a characterization of general conditions under which the problem is strongly solved.

The distributed gradient dynamics for the situation in Figure 4.13 is:

$$\begin{aligned}
\dot{x}(1,2) &= \frac{2}{3}\left(-x(1,2)+x(1,3)-x(2,3)\right), \\
\dot{x}(1,3) &= \frac{2}{3}\left(x(1,2)-x(1,3)+x(2,3)\right), \\
\dot{x}(2,3) &= \frac{2}{3}\left(-x(1,2)+x(1,3)-2x(2,3)+x(2,4)-x(3,4)\right), \\
\dot{x}(2,4) &= \frac{2}{3}\left(x(2,3)-x(2,4)+x(3,4)\right), \\
\dot{x}(3,4) &= \frac{2}{3}\left(-x(2,3)+x(2,4)-x(3,4)\right).
\end{aligned}$$

Sample trajectories for this system are shown in Figure 4.14. As can be seen, the sample trajectories converge exponentially to zero (each nodes net payment was zero for this example); this behavior is guaranteed by our previous deduction that the distribution operator has rank two, and thus strongly solves this problem.

Chapter 5

Summary, Conclusions, and Future Directions

5.1 Discussion and Summary

This work has been a study of a class of distributed coordination problems, and has presented a novel viewpoint in which to analyze and design dynamics for distributed coordination; herein we present a summary of the main issues examined in the preceding text. We will begin our summary of the work by examining its basic principles, and discussion of the technical aspects will follow thereafter.

Several overarching themes have appeared repeatedly in our discussions. The first and most obvious theme of this work has been the idea that coordination is in some way fundamentally tied to optimization; to reiterate the view expressed in the introduction, we have taken the position that coordination is essentially the improvement of global performance metrics while respecting global constraints. A second but equally important theme has been the idea that *distributed* coordination is also fundamentally tied to *dynamics*; the very notion of a distributed system is intrinsically tied to the process by which the global state evolves. Moreover, we have made heavy use of an *interplay* between optimization and dynamics, which has formed the basis for our "design theory".

We have paid considerable attention throughout to *systematic* design of distributed coordination mechanisms; perhaps our most significant break with traditional work on the subject has been in the *separation of functional specification from implementation.* We have used optimization as a general specification language for describing coordination problems, and have discussed the use of optimization models in specifying functional relationships. Simultaneously, we have coupled this idea with an architecture for dynamical systems, which we have referred to as "Stability - Invariance - Equilibrium". We have seen that this architecture allows us to encode functional relationships in a dynamical system by appropriately setting equilibrium and invariant manifolds. Utilizing this dynamical architecture as a means for *implementing* functional relationships *represented* by optimization models, we have arrived at the desired separation of functionality from implementation. However, this viewpoint is incomplete without a tool for *synthesizing* appropriate dynamical systems based on a specification, while ensuring that the resulting system is *appropriately distributed*; this latter goal has been accomplished by a different set of tools, coming from a geometric interpretation of distributed systems.

Our geometric viewpoint on distributed systems has been the "cornerstone" in bridging the gap between specification and implementation; it has accomplished several goals at several levels. First and foremost, it addresses yet another fundamental specification problem, that of the *model for the distributed system*; this too is a significant break from established approaches to this subject, in which *a priori* assumptions are made on a case-by-case basis. For example, it is very common to see models in which only immediate neighbors on a network can affect each others' states, or to see models in which it is *assumed* that there is a link between any two units whose dynamics influence each other. We have had no need for such assumptions, as our approach has been explicitly parametrized over the structure of the distributed system; pairwise interactions, global control, tree-like structures, and other distributed architectures are all treated uniformly by our geometric language. This representation has allowed us to systematically synthesize *different* dynamical systems for the *same* coordination specification, parametrized by the underlying network model. Further, this power of expression has made it possible to formally ask (and partially answer) basic questions regarding what kinds of coordination are achievable under various patterns of distributed interaction.

Another central aspect of our work has been the idea of *distributed proof* of coordination; we have taken as an explicit part of our framework the requirement that "coordinated actions" be locally verifiable. This has motivated both the development of some of our tools, as well as the formulation of basic questions regarding the possibility of obtaining distributed proof of coordination under different models for the networked interaction. This aspect of our philosophical assumptions has had perhaps the greatest impact on the development of our technical tools, which are all geared toward synthesizing dynamics with distributed proof that a certain specification is carried out.

The technical tools presented have reflected our overall viewpoint, and we have made use of our geometric language to provide a unified model for distributed optimization and dynamics. The fundamental concept that has driven this development is the *interaction*: a tangent subbundle of vectors modeling "locally available" tangent directions. This provides a natural language for describing the behavior of locally interacting units, and is extremely flexible; we can uniformly describe purely local control, global control, and various kinds of distributed control by an appropriate choice of subbundle. The power of an interaction is algebraically measured by the dimension of the subspaces it defines, and this corresponds naturally with our intuitive notion of how powerful an interaction should be. Dual to the idea of an interaction is a *coordination*, another tangent subbundle which represents tangent directions that respect some global constraints; again, we can uniformly express various kinds of coordinated behavior by choosing an appropriate subbundle. The complexity of a coordination is algebraically measured by its codimension, and this corresponds with an intuitive expectation that some coordinated behaviors are harder to achieve than others. Coupled with the idea of an interaction, this concept naturally provides the notion of *coordination capacity* for a given distributed interaction; as one would expect from intuition, the capacity of highly localized interactions is lower than that of richly coupled interactions.

We have applied our geometric model to present a framework for optimization under the constraints of distributed interaction. This has allowed us to define a distributed notion of an extremum, as well as the ability of a collection of interactions to solve an optimization problem. Coordination capacity was shown to have a natural connection to distributed optimization problems, and allowed us to make some "intuitive" statements regarding the possibility of implementing general specifications under a give model for the local interactions. Perhaps most important of all from a design perspective, our geometric formalism provided a natural definition for the *distributed gradient*, which has many of the desirable properties of a classical gradient (which is recovered as a special case of the distributed gradient under the "global control" interaction). The dynamics of distributed gradient flow was shown to go to a distributed extremum, and we were able to make use of our differential characterization of distributed extrema to argue that the equilibria of distributed gradient flows must in fact be highly structured. Furthermore, we were able to recast the well-known distributed averaging system $\Phi_L(\cdot)$ as a special case of our design framework, and to recover all of its desirable features as immediate consequences of the design formalism.

The tools we developed for distributed optimization provided precisely the elements we had sought to create in the proposed "Stability - Invariance - Equilibrium" design architecture. Convergence to an equilibrium is guaranteed by the properties of distributed gradient flow, and the differential characterization of distributed extrema provides structural information about the set of possible equilibria. Further, by distributing the gradient over an appropriately coordinated interaction set, we have *a priori* guarantees of whatever invariance property we desire. This allows us to take a general functional specification, which is agnostic of dynamics or a network model, and couple that with an interaction set modeling the underlying network to systematically synthesize dynamics implementing the desired coordination behavior. Moreover, this architecture provided the basis for some useful extensions of the coordinated behavior to dynamic tracking and reconfiguration that were not obvious in its absence.

We have presented a suite of applications of distributed coordination; the collection has been assembled to illustrate a variety of "practical" scenarios in which distributed coordination (and the tools of this thesis) could be utilized to achieve some desired goal. This section has provided some less formal, but nonetheless useful, design concepts for distributed systems. For example, we demonstrated the coupling of a distributed coordination mechanism with another nonlinear dynamical system in which the invariant structure of the coordination dynamics provided a crucial piece of structural information for analyzing the asymptotic behavior of the dynamics. Similarly, we examined the use of parallel and cascade interconnections of coordination mechanisms to execute more complex computational tasks than, say, simple averaging. Finally, we presented a collection of novel applications involving "topological coordination" where we demonstrated the entire collection of our new tools: specification, synthesis, and analysis of distributed coordination dynamics.

Throughout the text, we have attempted to emphasize the unity of the underlying design concept: the use of optimization to specify behaviors, the use of geometry to specify a distributed model, and a combination of the two to synthesize dynamics (as parametrized by a network structure) that implements the desired coordination. This, combined with the design and analysis framework of "Stability - Invariance - Equilibrium" has provided us with a suite of novel and powerful tools for generating and analyzing distributed coordination systems.

5.2 Main Limitations

Although we have been able to obtain a relatively self-contained modeling and design methodology, there are some significant limitations in its applicability as it stands (extensions are discussed in the following sections).

From the perspective of optimization theory, we have not discussed two important constructs: non-gradient (e.g. Newton-like) descent methods, and inequality constraints. We have relied on the gradient rather than other candidate descent directions for one fundamental reason: our mechanism for obtaining *distributed proof* of descent has focused on computing inner-products with the global gradient vector. The Newton descent vector will (in general) not align with the gradient vector, and so it cannot be used as a mechanism for computing distributed proof. It is possible to obtain distributed proof for Newton mechanisms in certain situations, but this is fairly technical and requires bounds on the condition number of the matrix of partial derivatives. To see this in a trivial case, note that the distributed averaging system actually has a Newton direction that aligns exactly with its gradient direction; this system's Hessian matrix is the identity, and so is as well-conditioned as possible.

Inequality constraints have also been excluded from our treatment because of difficulty in obtaining distributed proof that the global constraint is respected. For example, a constraint requiring the sum of all node states to be less than some constant is difficult to treat because it requires every node to know at all times whether the global state vector has reached the boundary of the feasible set. Cast geometrically, this is fundamentally a question about manifolds with boundaries versus manifolds without boundaries. We do not have a good technique to answer such questions, although we should point out that in the case that the inequality constraints happen to be locally observable, all of our current mechanisms apply (possibly generating non-smooth vector fields because of the inequality constraints). Put more plainly, if each node can communicate with every node to which it is coupled by an inequality, the present methods are applicable.

Finally, we have focused our work on coordination problems that are primarily of an informational nature; roughly, we have attempted to provide a dynamic communication infrastructure for executing computations that lead to coordinated states. In order to apply the mechanisms proposed here effectively, one must be able to cast the desired coordination task in the representation language of an optimization (or a cascade of optimizations) that is strongly solvable on the network structure of interest. This is to some extent an art, and it may not be clear how to generate this representation (especially for systems with intrinsic dynamics, as discussed in the following section). Nonetheless, this kind of restriction is common in this area of research (consider, for example, early Lyapunov theory). Further, the diversity of examples from Chapter 4 seems to suggest that with the right cascade, parallel, or feedback structure, a large class of problems can be cast in this framework.

5.3 Extensions and Future Work

The main areas for future development of this work fall into three broad categories: coordination for systems with intrinsic dynamics, technical extensions of the theory itself, and applications of the modeling tools to other types of distributed systems arising in other fields.

Throughout our work, we have tacitly assumed that we were free to design arbitrary dynamics and introduce arbitrary states for each node on our network; we took the liberty of inducing whatever structure was necessary to implement a particular coordinated behavior. In practice, one may have fundamental dynamical constraints preventing the direct application of this theory, and this seems like the most significant limitation of the work in its current form. As our theory currently stands it seems most suited for achieving coordination in some "abstract" sense, and less immediately applicable to situations in which one must control physically (or otherwise) limited dynamical systems. The problems that must be addressed in order to make this theory practical for systems with intrinsic dynamics mostly center around a different kind of tangent space constraint, describing what tangent vectors are *physically realizable* for a given system. It seems that our overall modeling viewpoint, focusing on distributed proof of improvement of some global function (for example, a candidate Lyapunov function), may be of use in designing such a theory even if it does require many of the individual pieces of this work to be recast in a more general framework. Similarly, one must grapple with the notion of specification and implementation of distributed coordination, although in the case where the system's dynamics is intrinsically limited this becomes somewhat problematic to separate. Indeed, at the point that simply *implementing* a desired tangent vector is in itself a design problem, the line between specification and implementation starts to become rather difficult to draw. On a more optimistic note, the distributed gradient seems likely to be useful even for systems with intrinsic dynamics; gradient-based control is common for many dynamical systems, and gradient structures also appear frequently in potential-controlled Hamiltonian systems. One can certainly imagine designing control potentials for Hamiltonian systems and using distributed gradients to synthesize appropriate control forces. This seems like a promising avenue of research, and appears to be the natural next step from the study of gradient-like systems.

The formalism we have presented has made some assumptions that were not strictly necessary from a technical point of view; as an example, the definitions of interaction and coordination could be somewhat more general. Abstractly, all one requires is a mapping from a vector field to another vector field which provably makes a non-negative inner product with the first (pointwise across the configuration manifold). While we have defined this in terms of orthogonal projections, one could certainly imagine other classes of mappings. Similarly, we defined coordinations as idempotent mappings acting on interactions, where we interpreted interactions as subbundles; by expanding our definition of interactions, we necessarily must expand our definition of coordinations to encompass the new classes of mappings allowed. In some sense, the transition we are proposing is analogous to the parallel between optimization theory based on linear algebra and multivariable calculus, and optimization theory based on convex and non-smooth analysis. In order to effect this expansion of the theory in a useful way, one will need a sensible characterization of the class of mappings producing vector fields with "provable progress" towards some goal, and in the general case this seems difficult to achieve; nonetheless, expanding the theory to include more than just the projective definition we have provided seems a likely avenue for future development.

The modeling approach we have presented, although geared towards our specific purposes, seems to be relatively general as a language for discussing spatially distributed interactions; thus, it seems possible that this kind of modeling approach could be applied to other pursuits involving the interaction of multiple decision-making units. It seems likely, for example, that this kind of approach may be of use in the economic sciences for the purpose of analyzing the decision-making behavior of groups having locally available information; as a model for local decision-making, it seems that our geometric framework provides useful tools and poses tractable problems, especially as regards optimization. The role of optimization in economics is, of course, fundamental; it seems possible that market equilibrium models might benefit from analysis as distributed systems, and that this might provide some useful insight into their behavior that classical optimization and game theory do not. Furthermore, it seems that the concept of coordinations might provide some useful insight into the effect of social norms and regulatory interference in market dynamics; presumably, a market model with low coordination capacity pays a steep price in efficiency when subjected to regulation, whereas a market with a rich interaction structure can tolerate significantly more regulation. The dynamics model presented may also yield some insight into equilibrium selection, although in order for this kind of analysis to be carried out one will have to produce a more general version of our model to account for the fact that each economic agent has differing local goals.

Finally, one would like to obtain a suitable discrete-time version of this set of tools; although it is possible to effectively discretize the dynamics of most of the systems we have described, this seems an unsatisfying solution to the problem. However, developing a similar theory for discrete-time systems seems a rather difficult undertaking because so much of our work has exploited the natural geometric structure of continuous models; the notion of interactions seems much more complex to define for systems in which updates occur at discrete time-steps, because we can no longer model the dynamics with tangent vectors. A more natural approach, perhaps, is to find systematic methods for discretizing the systems produced by a continuous model in such a way that we ensure the three underlying principles of our design architecture: stability, invariance, and equilibrium structure. This would be, in some sense, the most satisfying possible solution in that it would continue to exploit the underlying structure that has made this work possible, while systematically generating discrete-time systems implementing a given specification.

Bibliography

[1] I. Akyildiz, W. Su, Y. Sankarasubramniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, pages 102–114, August 2002.

[2] D. Bertsekas and J. Tsitsiklis. *Parallel and Distributed Computing: Numerical Computations.* Athena Scientific, 1989.

[3] K.M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison-Wesley Publications, 1989.

[4] J. Corfman and A. Morse. Decentralized control of linear multivariate systems. *IEEE Transactions on Automatic Control*, 1976.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. *Proc. of Mob. Comp. and Net.*, 1999.

[6] J.A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 2004.

[7] S. Floyd. Tcp and explicit congestion notification. *ACM Computer Communication Review*, 1994.

[8] Y. Hatano and M. Mesbahi. Consensus over random networks. In *Proceedings of the Conference on Decision and Control*, 2004.

[9] S. Haykin. *Neural Networks: A Comprehensive Foundation, 2nd Edition.* Prentice Hall, 1999.

[10] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. *Proc. of Mob. Comp. and Net.*, 1999.

[11] F. Hofbauer and H. Sigmund. *Evolutionary Games and Population Dynamics.* Addison-Wesley Publishers, 2001.

[12] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. In *Proceedings of the Conference on Decision and Control*, 2002.

[13] F.P. Kelly, A. Maulloo, and D. Tan.

[14] N.E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials, and coordinated control of groups.

[15] S. Low and D.E. Lapsley. Optimization flow control i, basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 1999.

[16] N. Lynch. *Distributed Algorithms.* Morgan Kauffman Publishers, 1997.

[17] M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, and R. Murray. Distributed averaging on asynchronous communication networks. In *Proceedings of the Conference on Decision and Control*, 2005.

[18] L. Moreau. Leaderless coordination via bidirectional and unidirectional time-dependent communication. In *Proceedings of the Conference on Decision and Control*, 2003.

[19] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 2004.

[20] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, 1999.

[21] R. Raffard, C. Tomlin, and S. Boyd. Distributed optimization for cooperative agents: Applications to formation flight.

[22] B.S. Rao and H.F. Durrant-Whyte. Fully decentralised algorithm for multisensor kalman filtering. *IEEE Proceedings-D*, 1991.

[23] R.Olfati-Saber. Ultrafast consensus on small-world networks. *Proceedings of ACC*, 2005.

[24] T. Schouwenaars, J. How, and E. Feron. Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees.

[25] D. Spanos and R. M. Murray. Distributed sensor fusion using dynamic consensus. In *IFAC World Congress*, 2005.

[26] D. Spanos, R. Olfati-Saber, and R. M. Murray. Dynamic consensus for mobile networks. In *IFAC World Congress*, 2005.

[27] D. Swaroop and J.K. Hedrick. String stability of interconnected systems. *IEEE Transactions on Automatic Control*, 1996.

[28] S. Wang and E. Davison. On the stabilization of decentralized control systems. *IEEE Transactions on Automatic Control*, 1973.

[29] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. In *Proceedings of the Conference on Decision and Control*, 2003.

[30] L. Xiao, S. Boyd, and S. Lall. A scheme for asynchronous distributed sensor fusion based on average consensus. *Proceedings of IPSN*, 2005.