The Degree of Nonholonomy in Distributed Computations

Zak Costello and Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology
{zak.costello,magnus}@gatech.edu

Abstract— **A network of locally interacting agents can be thought of as performing a distributed computation. But not all computations can be faithfully distributed. This paper discusses which global linear transformations can be computed in finite time using local weighting rules, i.e., rules which rely solely on information from adjacent nodes in a network. Additionally, it is shown that the degree of nonholonomy of the computation can be related to the underlying information exchange graph. The main result states that the degree of nonholonomy of the system dynamics is equal to $D - 1$ where D is the diameter of the information exchange graph. An optimal control problem is solved for finding the local interaction rules, and a simulation is performed to elucidate how optimal solutions can be obtained.**

I. Introduction

One common theme when designing control and coordination mechanisms for distributed, multi-agent systems is that the information, on which decisions are based, is restricted to be shared among agents that are adjacent in the underlying information-exchange network, e.g., [1], [2], [3], [4]. As a result, local rules are needed for processing the information and coordinating the agents in the network in such a way that some global objective is achieved. Problems that fit this description can be found in a variety of applications, including power systems [5], [6], [7], formation control [8], [9], [10], [11], [12], distributed sensor networks [13], [14], smart textiles [15], and distributed optimization [16], [17]. In this paper we take initial steps towards developing a general theory of local implementability/computability of such global behaviors.

One key aspect of distributed algorithm design is the definition of local interaction rules that produce desired global behaviors. An example of this are consensus algorithms for computing averages in a distributed manner. In fact, consensus plays a role in many different applications, including multi-agent robotics, distributed sensor fusion, and power network control, e.g., [3], [6], [18]. To this end, let the scalar state of each node in a network be $x_i \in \mathbb{R}$, with initial condition $x_i(t_0) = \xi_i, i = 1, \ldots, n$, where n is the number of nodes in the network. By stacking the states together in $x \in \mathbb{R}^n$, we implicitly perform an asymptotic, global computation through the so-called consensus equation

$$\dot{x}_i = -\sum_{j \in N_i} (x_i - x_j), \tag{1}$$

where N_i encodes a neighborhood relationship in the underlying information-exchange network. And, if the network is connected and undirected, all node values will converge to the same value, namely the average of the initial conditions, e.g., [2]. In other words,

$$\lim_{t\to\infty} x(t) = \frac{1}{n} \begin{bmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{bmatrix} \xi, \tag{2}$$

where ξ is the vector containing all the initial node values. As such, the consensus equation is asymptotically computing the average, which is a global property since it relies on the state of every node in the network.

In this work, we are interested in problems where networks are tasked with computing arbitrary linear transformations of the initial node states. In particular, we address two fundamental questions: *What global, linear transformations can be computed using local rules? How do we find the local rules that would compute a given linear transformation?*

There are two independent issues at play when investigating decentralized computations. First, the set of decentralized rules must be created. Then, the rules must be executed. In this paper, the focus is on characterizing the set of decentralized rules. The creation of those rules need not be decentralized in general, the problem of decentralizing the creation of decentralized rules is left as a separate issue for further study and is not addressed in this paper.

Some work has been done in the general area of obtaining global information with local interactions. In [19], a fixed weighting scheme was used to compute linear transformations on networks. That work has a similar aim and takes a different discrete time approach. In a certain sense, the investigation in [20] follows this line of inquiry as well. There, quadratic invariance was used to establish whether or not a convex optimization problem exists whose solution is a decentralized implementation of a centralized feedback controller. [21] further expounds on this idea and provides a practical, graph theoretic method for finding this distributed controller. Our work distinguishes itself from this body of work by using a time varying weighting method, which admits the computation of global, linear transformations in finite time. In this paper, we consider computations that are to be performed in continuous time, using local rules over a static information-exchange network. The local rules, once obtained, admits a decentralized implementation, where "decentralized" in this context means that each node in the network only needs to communicate state information among adjacent nodes in the network. In [22] necessary and sufficient conditions were derived for the existence of time varying weights which can compute some given linear transformation. We expand on the work in [22] and analyze how the network topology is related to how complicated the computation is, in the sense that we provide a theorem relating the topology of the underlying information exchange graph to the degree of nonholonomy of the system.

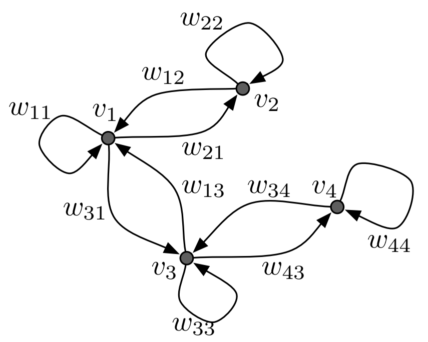
II. Problem Definition

To formalize what is meant by local interactions, we first need to discuss the information-exchange network over which the interactions are defined. To this end, let V be a vertex set with cardinality n, and $E \subset V \times V$ be an edge set of ordered pairs, with cardinality m, where we insist on $(i,i) \in E,\ \forall i \in V$, as well as $(i,j) \in E \Leftrightarrow (j,i) \in E$. We say that $\mathcal{G}$ is a *computation graph* if $\mathcal{G}$ is a strongly connected graph $\mathcal{G} = (V,E)$, where the structural assumptions on E imply that $\mathcal{G}$ is directed and contains self-loops. We moreover assume that $\mathcal{G}$ is connected. As the main purpose with $\mathcal{G}$ is to encode adjacency information in the information-exchange network, we introduce the operator sparse($\mathcal{G}$) to capture these adjacencies, and we say that an $n \times n$ matrix $M \in$ sparse($\mathcal{G}$) if $(i,j) \notin E \Rightarrow M_{ij} = 0$.

There are a number of different ways in which local interactions can be defined. In this paper, we assume that they are given by time-varying, piecewise continuous weights associated with the edges in the network. If $x_i \in \mathbb{R}$ is the scalar state associated with node $i \in V$, we define a local interaction as a continuous-time process

$$\dot{x}_i(t) = \sum_{j|(i,j)\in E} w_{ij}(t)x_j(t). \tag{3}$$

Note that we do not insist on $w_{ij} = w_{ji}$, as seen in Figure 1.



$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & 0 \\ w_{21} & w_{22} & 0 & 0 \\ w_{31} & 0 & w_{33} & w_{34} \\ 0 & 0 & w_{43} & w_{44} \end{bmatrix} \in sparse(\mathcal{G})$$

Fig. 1. An example of the sparsity structure and the local interaction rules used in this paper.

If we stack the states together in $x = [x_1, \dots, x_n]^T \in \mathbb{R}^n$, what we mean by *local interactions* is thus

$$\dot{x}(t) = W(t)x(t), \quad W(t) \in \text{sparse}(\mathcal{G}), \tag{4}$$

with solution

$$x(t) = \Phi(t,t_0)x(t_0), \tag{5}$$

where Φ is the state transition matrix associated with the system in (4), e.g., [23].

Now, the purpose of the local interactions is to perform a global linear computation. In other words, given the $n \times n$ matrix T and the initial condition $x(t_0) = \xi$, what we would like to do is find $W(t) \in \text{sparse}(\mathcal{G}),\ t \in [t_0, t_f]$, such that

$$x(t_f) = T\xi. \tag{6}$$

By comparing this expression to (5), this simply means that what we would like is

$$\Phi(t_f, t_0) = T. \tag{7}$$

If this was the case, then the local interactions, as defined through $W(t)$, would indeed compute $T\xi$ over the interval $[t_0, t_f]$ for all possible values of ξ, i.e., one can think of the network as a black box that takes ξ as the input at time t_0 and, at time t_f, returns $T\xi$ as the output.

As a final observation before we can formulate the general problem of performing global, linear computations using local interactions, we note that state transition matrix satisfies the same dynamics as (4), i.e.,

$$\frac{d\Phi(t,t_0)}{dt} = W(t)\Phi(t,t_0), \tag{8}$$

with initial condition $\Phi(t_0,t_0) = I$, where I is the $n \times n$ identity matrix.

Problem 1 [Local Computation]
Given a linear transformation T and a computation graph $\mathcal{G}$, find $W(t) \in sparse(\mathcal{G}),\ t \in [t_0, t_f]$, such that

$$\dot{\mathbf{X}}(t) = W(t)\mathbf{X}(t), \tag{9}$$

with boundary conditions $\mathbf{X}(t_0) = I,\ \mathbf{X}(t_f) = T$.

III. On the Implications of Possible Solutions

In order for a solution to Problem 1 to exist we previously showed in [22] that the desired transformation T must be in the set of $n \times n$ positive determinant matrices. This set constitutes a group together with the standard matrix product. Here this group is denoted $GL_+^n(\mathbb{R})$. This condition turns out to be both necessary and sufficient and can be stated succinctly as follows:

Theorem 1. *[22] A solution to Problem 1 exists if and only if $T \in GL_+^n(\mathbb{R})$.*

This result can be interpreted to mean that whenever $\det(T) > 0$, there is a $W(t) \in$ sparse($\mathcal{G}$) that drives $\mathbf{X}$ from I to T. The remainder of this section is devoted to the implications of this fact.

One consequence of Theorem 1 is that it is impossible to use local rules, as understood in this paper, to achieve consensus in finite time. This follows directly from the fact that the consensus computation is given by the linear map

$$T_{cons} = \frac{1}{n}\mathbf{1}\mathbf{1}^T, \tag{10}$$

where $\mathbf{1}$ is a vector of length n, with all entries equal to one. And,

$$\text{rank}(T_{cons}) = 1,$$

i.e., $\det(T_{cons}) = 0$. Since any valid transformation, $T \in \text{GL}_+^n(\mathbb{R})$, by definition, $T \in \{X \in \mathbb{R}^{n\times n} \mid \det(X) > 0\}$. So, T_{cons} is not in the feasible set. We state this fact as a corollary:

Corollary 1. *There is no solution to Problem 1 which admits finite time consensus.*[1,2]

However, consider instead the transformation

$$T_{cons2} = \begin{bmatrix} 1/n & 1/n & \cdots & 1/n \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{11}$$

We have

$$\det(T_{cons2}) = \frac{1}{n} \tag{12}$$

and, as such, it is computable using local rules. In this case, the network average is only computed by a single node (node 1 in this case), while the remaining nodes return to their initial values at the end of the computation. This can in fact be generalized to any scalar, non-zero, linear map $\ell : \mathbb{R}^n \to \mathbb{R}$ through

$$T_\ell \xi = \begin{bmatrix} \ell(\xi) \\ \xi_2 \\ \vdots \\ \xi_n \end{bmatrix},$$

where we have assumed that $\ell(\xi)$ depends on ξ_1.[3] The point with this is that *it is possible to compute any scalar, non-zero, linear map as long as the computation only has to take place at a single node.*

Another observation is that if $W(t)$ produces the linear computation T, then T^{-1} is computable as well. In fact, noting that $\mathbf{X}(t_f) = T$ then $\mathbf{X}^{-1}(t_f) = T^{-1}$. Since $\mathbf{X}(t)$ is a state transition matrix $\mathbf{X}(t) = \Phi(t, t_0)$ we have $\mathbf{X}^{-1}(t_f) = \Phi(t_0, t_f)$. As such, if $\mathbf{Y}(t_0) = I$ and

$$\dot{\mathbf{Y}}(t) = -W(t_f - t)\mathbf{Y}(t) \tag{13}$$

we obtain, $\mathbf{Y}(t_f) = T^{-1}$.

[1]Note that this applies to any agreement across the nodes, i.e., not only to average consensus.

[2]Consensus in continuous time can , of course, be achieved asymptotically since $\det(T) = 0$ can be approached as time tends to infinity.

[3]If not, simply pick another node in the network that ξ does depend on, as the node where the computation takes place.

IV. Nonholonomy of Distributed Computation

Here we discuss how the connectivity of the computation graph $\mathcal{G}$ impacts the complexity of the computation. Specifically, we look at the relationship between graph structure and degree of nonholonomy of the drift free system expressed in (9). In order to develop these results, the system in (9) is reworked into a more standard, drift free form, and several supporting results are developed.

The standard format of a drift free system, e.g., [24], is

$$\dot{x} = \sum_{i=1}^{q} g_i(x)u_i, \tag{14}$$

where x is the state of the system, and $u_1, \ldots, u_q \in \mathbb{R}$ are the control inputs. To express the system in (9) on this form, the *index matrix* $\mathbb{I}_{ij} \in \mathbb{R}^{n\times n}$ is needed. This matrix is defined as having a one at the ith row and jth column, and zeros everywhere else. The index matrix allows us to rewrite

$$\dot{\mathbf{X}} = W\mathbf{X}$$

as

$$\dot{\mathbf{X}} = \left(\sum_{i=1}^{n}\sum_{j=1}^{n} W \odot \mathbb{I}_{ij}\right)\mathbf{X}, \tag{15}$$

where the $\odot$ symbol represents element-wise matrix product, i.e.,

$$\dot{\mathbf{X}} = \left(\begin{bmatrix} w_{11} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{bmatrix} + \ldots + \begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & w_{nn} \end{bmatrix}\right)\mathbf{X}, \tag{16}$$

where we have suppressed the explicit dependence on t for the sake of notational ease. Rearranging the terms and letting

$$g_{ij}(\mathbf{X}) = \mathbb{I}_{ij}\mathbf{X}, \tag{17}$$

we get the drift-free matrix formulation

$$\dot{\mathbf{X}} = \sum_{i=1}^{n}\sum_{j\mid(i,j)\in E} g_{ij}(\mathbf{X})w_{ij}. \tag{18}$$

To clarify, $g_{ij}(\mathbf{X})$ is a matrix whose ith row contains the jth row of $\mathbf{X}$, with the rest of the elements in the matrix equal to 0,

$$g_{ij}(\mathbf{X}) = \begin{array}{c} 1 \\ \vdots \\ i-1 \\ i \\ i+1 \\ \vdots \\ n \end{array}\begin{bmatrix} 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \\ \mathbf{X}_{j1} & \ldots & \mathbf{X}_{jn} \\ 0 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & 0 \end{bmatrix}. \tag{19}$$

Vectorizing this equation is useful in the analysis of this system as Lie brackets play a key role in the analysis of the drift free system. The vectorized version of g_{ij} is given by $\vec{g}_{ij} = \text{vec}(g_{ij})$, resulting in the vectorized version of (18),

$$\text{vec}(\dot{\mathbf{X}}) = \sum_{i=1}^{n} \sum_{j|(i,j)\in E} \vec{g}_{ij}(\mathbf{X}) w_{ij}. \tag{20}$$

The first order of business towards establishing the relationship between the degree of nonholonomy of this system and the underlying communication graph is the derivation of the Lie brackets for the system in (20).

Lemma 1.

$$[\vec{g}_{ij}(\mathbf{X}), \vec{g}_{kl}(\mathbf{X})] = \begin{cases} -\vec{g}_{il}(\mathbf{X}) & \textit{if } j = k,\ i \neq l \\ \vec{g}_{kj}(\mathbf{X}) & \textit{if } i = l,\ j \neq k \\ \mathbf{0} & \textit{otherwise} \end{cases} \tag{21}$$

Proof. The Lie bracket $[\vec{g}_{ij}, \vec{g}_{kl}]$ is given by

$$[\vec{g}_{ij}, \vec{g}_{kl}] = \frac{\partial \vec{g}_{kl}}{\partial \text{vec}(\mathbf{X})} \vec{g}_{ij} - \frac{\partial \vec{g}_{ij}}{\partial \text{vec}(\mathbf{X})} \vec{g}_{kl}, \tag{22}$$

where we have suppressed the explicit dependence on $\mathbf{X}$.

Substitution of (17) into (22), the above expression yields

$$\frac{\partial(\text{vec}(\mathbb{I}_{kl}\mathbf{X}))}{\partial \text{vec}(\mathbf{X})} \text{vec}(\mathbb{I}_{ij}\mathbf{X}) - \frac{\partial(\text{vec}(\mathbb{I}_{ij}\mathbf{X}))}{\partial \text{vec}(\mathbf{X})} \text{vec}(\mathbb{I}_{kl}\mathbf{X}), \tag{23}$$

which can be rewritten, using the Kronecker product, as

$$\begin{aligned} &\frac{\partial((I \otimes \mathbb{I}_{kl})\text{vec}(\mathbf{X}))}{\partial \text{vec}(\mathbf{X})} (I \otimes \mathbb{I}_{ij})\text{vec}(\mathbf{X}) \\ -&\frac{\partial((I \otimes \mathbb{I}_{ij})\text{vec}(\mathbf{X}))}{\partial \text{vec}(\mathbf{X})} (I \otimes \mathbb{I}_{kl})\text{vec}(\mathbf{X}) \end{aligned} \tag{24}$$

Taking the above derivatives yields

$$(I \otimes \mathbb{I}_{kl})(I \otimes \mathbb{I}_{ij})\text{vec}(\mathbf{X}) - (I \otimes \mathbb{I}_{ij})(I \otimes \mathbb{I}_{kl})\text{vec}(\mathbf{X}). \tag{25}$$

Using the mixed product property of the Kronecker product, (22) can be further simplified as

$$(I \otimes \mathbb{I}_{kl}\mathbb{I}_{ij})\text{vec}(\mathbf{X}) - (I \otimes \mathbb{I}_{ij}\mathbb{I}_{kl})\text{vec}(\mathbf{X}), \tag{26}$$

i.e., the Lie bracket in (22) becomes

$$[\vec{g}_{ij}(\mathbf{X}), \vec{g}_{kl}(\mathbf{X})] = \text{vec}(\mathbb{I}_{kl}\mathbb{I}_{ij}\mathbf{X}) - \text{vec}(\mathbb{I}_{ij}\mathbb{I}_{kl}\mathbf{X}). \tag{27}$$

Now, using the fact that, $\mathbb{I}_{ij}\mathbb{I}_{kl} = \mathbb{I}_{il}$ if $j = k$ and $\mathbb{I}_{ij}\mathbb{I}_{kl} = 0$ otherwise, we can break down (27) into 3 cases:

First if $j = k$ and $i \neq j$ we get

$$[\vec{g}_{ij}, \vec{g}_{kl}] = -\text{vec}(\mathbb{I}_{il}\mathbf{X}) = -\vec{g}_{il}.$$

The second case occurs when $i = l$ and $j \neq k$, in which case

$$[\vec{g}_{ij}, \vec{g}_{kl}] = \text{vec}(\mathbb{I}_{kj}\mathbf{X}) = \vec{g}_{kj}.$$

Otherwise, the Lie bracket is $\mathbf{0}$, and the lemma follows. □

To better understand what Lemma 1 really states, note that for every edge $(i,j) \in E$ there exists a corresponding vector field $\vec{g}_{ij}(\mathbf{X})$. One way to interpret the Lie bracketing operation that follows from Lemma 1 is that when applied to vector fields corresponding to adjacent edges, it creates new vector field corresponding to a new edge.

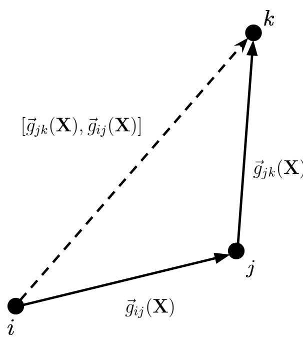
For example, consider the directed three node line graph shown in Figure 2. The directed edges (i,j) and (j,k) each have an associated vector field, $\vec{g}_{ij}(\mathbf{X})$ and $\vec{g}_{jk}(\mathbf{X})$ respectively. Applying Lemma 1, the Lie bracket is $[\vec{g}_{jk}(\mathbf{X}), \vec{g}_{ij}(\mathbf{X})] = \vec{g}_{ik}(\mathbf{X})$. This resulting vector field is associated with adding the edge (i,k) to the graph. So, by modulating vector fields associated with edges in a particular information-exchange graph we can make the graph behave as if it has additional edges. Consider the nested Lie bracket operation

$$[\vec{g}_{kl}(\mathbf{X}), [\vec{g}_{jk}(\mathbf{X}), \vec{g}_{ij}(\mathbf{X})]] \tag{28}$$

which can be simplified to

$$[\vec{g}_{kl}(\mathbf{X}), \vec{g}_{ik}(\mathbf{X})] = \vec{g}_{il}(\mathbf{X}) \tag{29}$$

This application shows that by nesting Lie brackets, virtual edges connecting nodes of increasing distances away can be created. In this example, a vector field associated with an edge between nodes i and l was created. These nodes were separated with a distance of 3 and were virtually connected with two nested lie brackets. *In general, through repeated applications of Lemma 1, any connected graph can be made to behave like a complete graph.*



k
$[\vec{g}_{jk}(\mathbf{X}), \vec{g}_{ij}(\mathbf{X})]$
$\vec{g}_{jk}(\mathbf{X})$
j
$\vec{g}_{ij}(\mathbf{X})$
i

Fig. 2. For every edge in the information exchange graph $\mathcal{G}$ connecting nodes i and j, there is a corresponding vector field $\vec{g}_{ij}(\mathbf{X})$. The Lie bracket operation when applied to the vector fields corresponding two two adjacent edges is equivalent to the vector field of the edge created by this sets closure.

Now, that a Lie bracketing relationship has been established for the drift free system, several definitions are needed in order to connect Lie brackets to the properties of the computation graph $\mathcal{G}$.

Definition 1 (Distance)**.** *The distance between two nodes in a graph u, v is denoted $d(u,v)$. The value of $d(u,v)$ is the cardinality of the edge set that comprises the shortest path connecting the two nodes.*

Definition 2 (Graph Diameter)**.** *The* Graph Diameter *D of a graph $\mathcal{G} = (V,E)$ is $\max\limits_{u,v} d(u,v)$ for any $u, v \in V$.*

Given a distribution of a drift free system Δ, A filtration is defined as follows: $G_i = G_{i-1} + [G_1, G_{i-1}]$. Where

$G_1 = \Delta$ and $[G_1, G_{i-1}] = span\{[g,h] | g \in G_1, h \in G_{i-1}\}$. If a filtration does not increase the dimension of the space spanned by G_i then that distribution is said to be *involutive*.

Definition 3 (Degree of Nonholonomy)**.** *The degree of nonholonomy is given by the least number of filtrations p required to reach an involutive distribution. (See for example [24].)*

Using the above definitions, the following lemma is presented in order to relate a particular vector field to the number of filtration required to form it.

Lemma 2. *For any $u, v \in V$, $\vec{g}_{uv}(\mathbf{X}) \in G_{d(u,v)-1}$.*

Proof. Because $\mathcal{G}$ is strongly connected, every node pair $u, v \in V$ is path-connected. Path-connected means that there is a path through adjacent nodes in the graph $\mathcal{G}$ that starts at node u and ends at node v. The distance between the two nodes, $d(u,v)$ gives the number of edges in the shortest path connecting u and v. Using that fact, assume that the path goes through the nodes $N_1, \ldots, N_{d(u,v)+1}$, i.e., N_1 is adjacent to N_2, N_2 is adjacent to N_3, and so forth, while $N_1 = u$ and $N_{d(u,v)+1} = v$. Since these nodes are adjacent, we, by definition, have that $\vec{g}_{N_1 N_2}, \vec{g}_{N_2 N_3}, \ldots, \vec{g}_{N_{d(u,v)} N_{d(u,v)+1}} \in \Delta(\mathbf{X})$.

Since the involutive closure contains every possible Lie bracket that can be recursively created from elements $\Delta(\mathbf{X})$, the problem is to create $\vec{g}_{uv}$ from some combination of Lie brackets from elements in $\Delta(\mathbf{X})$. And, from Lemma 1, we know that $[\vec{g}_{N_2 N_3}, \vec{g}_{N_1 N_2}]$ is equal to $\vec{g}_{N_1 N_3}$. Applying Lemma 1 again gives $[\vec{g}_{N_3 N_4}, \vec{g}_{N_1 N_3}] = \vec{g}_{N_1 N_4}$. If this procedure is recursively applied $d(u,v) - 1$ times, we arrive at $[\vec{g}_{N_{d(u,v)} N_{d(u,v)+1}}, \vec{g}_{N_1 N_{d(u,v)}}] = \vec{g}_{N_1 N_{d(u,v)+1}}$. So, we are able to construct $\vec{g}_{N_1 N_{d(u,v)+1}}$ from previous Lie brackets. And, as $N_1 = u$ and $N_{d(u,v)+1} = v$, $d(u,v) - 1$ lie bracketing operations are required and therefore $d(u,v) - 1$ filtrations are required. So, we arrive at $\vec{g}_{uv} \in G_{d(u,v)}$.

□

The previous result can be used to relate the distance between any two nodes in the computation graph to the degree of nonholonomy of the system. This theorem can be intuitively understood because the number of filtrations needed to form a particular vector field is equal to the distance between a pair of nodes. To link the information-exchange graph to the degree of nonholonomy of (18), one final intermediate result is needed.

Lemma 3. *$\{\vec{g}_{uv}(\mathbf{X}) \mid u, v \in V\}$ is a set of linearly independent vector fields.*

Proof. By definition, $\vec{g}_{uv}(\mathbf{X}) = \text{vec}(\mathbb{I}_{uv}\mathbf{X})$. This definition can be expanded to $\text{vec}(\mathbb{I}_{uv}\mathbf{X}) = (\mathbf{X}^T \otimes I_n)\text{vec}(\mathbb{I}_{uv})$. Concatenating all possible vectors resulting from combinations of vertices,

$$(\mathbf{X}^T \otimes I_n) \begin{bmatrix} \text{vec}(\mathbb{I}_{11}) & \text{vec}(\mathbb{I}_{21}) & \ldots & \text{vec}(\mathbb{I}_{nn})) \end{bmatrix} \tag{30}$$

which can be further simplified to

$$(\mathbf{X}^T \otimes I_n) I_{n^2} \tag{31}$$

Taking the determinant of this expression yields

$$det(\mathbf{X}^T \otimes I_n) = det(\mathbf{X}^T)^n \tag{32}$$

Because $\mathbf{X} \in GL_n^+(\mathbb{R})$ the determinant of X is always positive and therefore we can write

$$det(\mathbf{X}^T)^n \neq 0 \tag{33}$$

This implies that the set of vectors $\{\vec{g}_{uv}(\mathbf{X}) \forall u, v \in V\}$ is linearly independent. □

Using the lemmas developed above we can prove the main result of this paper:

Theorem 2. *The Degree of Nonholonomy of (18) is equal to $D - 1$ where D is the diameter of the computation graph $\mathcal{G}$.*

Proof. By definition the diameter of $\mathcal{G}$ is $D = \max_{u,v} d(u,v)$. Because, $G_i = G_{i-1} + [G_1, G_{i-1}]$, for any $i < D - 1$, $G_i \subset G_{D-1}$. Since D is the maximum distance of any path between two nodes in $\mathcal{G}$ it follows from Lemma 2 that any $u, v \in V$ $\vec{g}_{uv}(\mathbf{X}) \in G_{D-1}$. Lemma 2 also gives that G_{D-1} is the smallest possible filtration that contains every $\vec{g}_{uv}(\mathbf{X})$ because there is atleast one pair of nodes u, v which has a shortest path length of D. By Lemma 3 this set of vectors is linearly independent. Since we know that there are n^2 such vectors it can be concluded that they span the space of $\mathbb{R}^{n^2}$. Therefore, G_{D-1} spans the entire state space of (18) and is involutive. It follow that $D-1$ is the degree of nonholonomy of (18). □

Theorem 2 provides a connection between graph structure and dynamic constraints. The Lie bracket operation on adjacent edges can be interpreted as information flow along edges of a system. If information is to flow between node i and j, the system can travel along the vector field $\vec{g}_{ij}(\mathbf{X})$. Equivalently, information flow along $\vec{g}_{jk}(\mathbf{X})$ occurs when information flows between node j and k. The Lie bracket corresponding to $[\vec{g}_{ij}(\mathbf{X}), \vec{g}_{jk}(\mathbf{X})]$ creates a vector field corresponding to information flow between nodes i and k. In this way, the result in Theorem 2 becomes intuitive. In order to have information flowing between every node in the graph for a global computation, there must be a vector field corresponding to information flow between each node.

Considering that the diameter of a graph is the maximum path length between any two nodes, the number of informational hops required to connect each node to every other node is equal to the diameter. Additionally, the Lie bracket operation increases in order for each required hop. So, Information flow and the Lie bracket operation are inherently tied together.

V. Optimal Local Interactions

Just because we know that a computation $T\xi$ can be done using local rules it does not follow that we can (easily) find these rules, encoded through $W(t) \in \text{sparse}(\mathcal{G})$, such that $\dot{\mathbf{X}} = W\mathbf{X},\ \mathbf{X}(t_0) = I, \mathbf{X}(t_f) = T$. In this section, we address this problem in the context of optimal control.

Let the cost be given by

$$J(W) = \int_0^{t_f} \frac{1}{2}\|W(t)\|_F^2 dt, \tag{34}$$

where $\|\cdot\|_F$ is the Frobenius norm. The resulting constrained minimization problem becomes

Problem 2 [Optimal Local Interactions]

$$\min_W J(W) = \int_0^{t_f} \frac{1}{2}\|W(t)\|_F^2 dt \tag{35}$$

such that

$$\begin{array}{l} \dot{\mathbf{X}} = W\mathbf{X} \\ W(t) \in \textit{sparse}(\mathcal{G}),\ \forall t \in [t_0, t_f] \\ \mathbf{X}(t_0) = I,\ \mathbf{X}(t_f) = T. \end{array} \tag{36}$$

The Hamiltonian associated with Problem 2 (e.g., [25]), with costate matrix λ, is given by

$$H = \text{vec}(\lambda)^T \text{vec}(W\mathbf{X}) + \frac{1}{2}\|W\|_F^2. \tag{37}$$

We can rewrite the Hamiltonian as

$$H = \sum_{i=1}^n \sum_{j|(i,j)\in E} \sum_{k=1}^n \lambda_{ik} w_{ij} \mathbf{X}_{jk} + \frac{1}{2} \sum_{i=1}^n \sum_{j|(i,j)\in E} w_{ij}^2. \tag{38}$$

The optimality conditions are

$$0 = \frac{\partial H}{\partial w_{ij}} = \sum_{k=1}^n \lambda_{ik} \mathbf{X}_{jk} + w_{ij}, \tag{39}$$

i.e., the optimal weights are given by

$$w_{ij} = -\sum_{k=1}^n \lambda_{ik} \mathbf{X}_{jk}, \tag{40}$$

which yields $m + n$ optimality conditions. This is also the number of nonzero values in the W matrix.

We get the costate equations from the derivative of the Hamiltonian with respect to $\mathbf{X}$:

$$\dot{\lambda}_{ij} = -\frac{\partial H}{\partial \mathbf{X}_{ij}} = -\sum_{k|(i,k)\in E} w_{ki} \lambda_{kj}. \tag{41}$$

By substituting the optimality conditions into both the state and costate equations, we get $2n$ equations with initial and final conditions on $\mathbf{X}_{ij}$. The resulting, two-point boundary problem becomes

$$\begin{array}{rcl} \dot{\mathbf{X}}_{ij} & = & -\sum_{k|(i,k)\in E} \mathbf{X}_{kj} \sum_{l=1}^n \lambda_{il} \mathbf{X}_{kl} \\ \mathbf{X}(t_0) & = & I,\ \mathbf{X}(t_f) = T \\ \dot{\lambda}_{ij} & = & \sum_{k|(i,k)\in E} \lambda_{kj} \sum_{l=1}^n \lambda_{kl} \mathbf{X}_{il}, \end{array} \tag{42}$$

which can be solved numerically, as will be seen in the next section.

VI. Simulation

Computing linear transformation of states can be useful in a variety of network applications. In this section we one such example. We consider the case of distributed information exchange among non-local agents. Consider the situation when the linear transformation represents a reordering (or swapping) of states. For a 4 node case, where agents 1 and 2 and agents 3 and 4 are to "swap" state values, the transformation matrix becomes

$$T_{swap} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{43}$$

However, the linear interpolation between I and T_{swap} contains a singular matrix, which makes the two-point boundary problem numerically ill-conditioned when using shooting methods, e.g., [26]. A way around this problem is to avoid this singular matrix by solving two sequential two-point boundary problems.

As an example, in the first iteration, we let the boundary conditions be $\mathbf{X}(t_0) = I,\ \mathbf{X}((t_f - t_0)/2) = T_1$. For the second iteration, they are $\mathbf{X}((t_f - t_0)/2) = T_1,\ \mathbf{X}(t_f) = T_{swap}$, where

$$T_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{44}$$

This sequential approach avoids the numerical ill-conditioning, and the solution is shown in Figures 3 - 5.

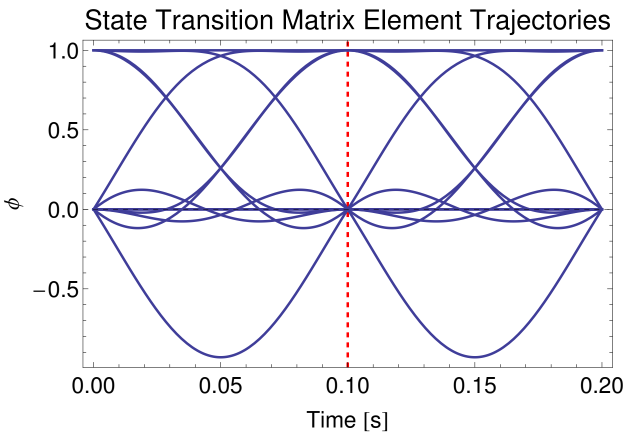


Fig. 3. The evolution of the state transition matrix for the 4-node swapping problem is shown above, with $\Phi(t_0, t_0) = I,\ \Phi((t_f - t_0)/2, t_0) = T_1$, and $\Phi(t_f, t_0) = T_{swap}$. Each line in this plot shows the evolution of one element in the state transition matrix while computing a position swap of 4 nodes. Notice that at t_0, every element starts at either 1 or 0, indicating that the state transition matrix starts at the identity matrix.

VII. Conclusions

In this paper, a step was taken towards computing global functions on networks with local interaction rules. In particular, it presented a method which allows a networked system to compute global, linear transformations using only

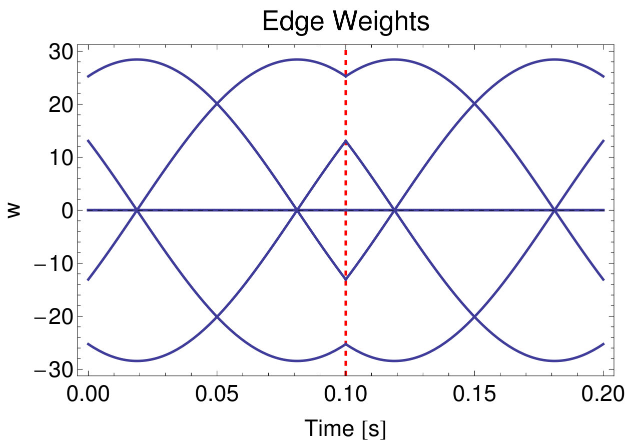


Fig. 4. The edge weight functions define the local interactions needed to achieve the swap in the 4-node case. Each line represents an edge weight over the time horizon of interest.

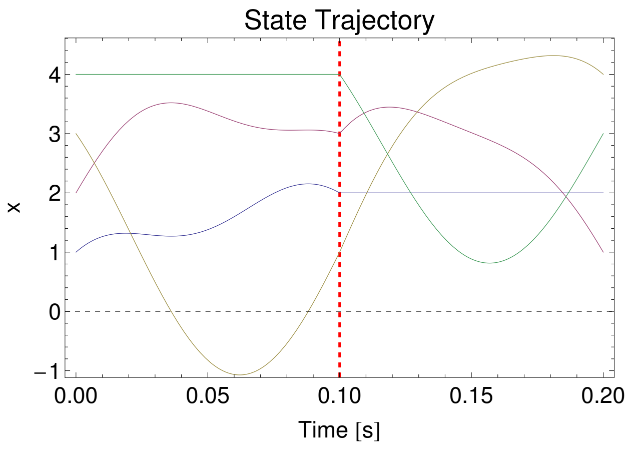


Fig. 5. The evolution of the node states for the swap problem. The initial state is $x(t_0) = [1, 2, 3, 4]^T$ and the final state is $x(t_f) = [2, 1, 4, 3]^T$, i.e., the first and second states swapped values and the third and fourth state swapped values.

local rules. We provided necessary and sufficient conditions under which it is possible to use a distributed, time-varying weighting scheme to compute the transformation T for undirected, connected networks with fixed topology. The implications of those conditions were discussed and the relationship between distributed computation and degree of nonholonomy was developed. Specifically, we showed that the degree of nonholonomy of distributed computation under this scheme was equal to $D - 1$ where D is the diameter of the underlying information-exchange graph.

Acknowledgment

This work was sponsored in part by a grant from the US Air Force Office for Sponsored Research. The authors would like to thank Professor Mark Costello at the Georgia Institute of Technology for his advice regarding this work.

References

[1] F. Bullo, J. Cortes, and S. Martnez, *Distributed Control of Robotic Networks. A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.

[2] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[4] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control*. Springer-Verlag, 2008.

[5] A. L. Dimeas and N. D. Hatziargyriou, “Operation of a multiagent system for microgrid control,” *Power Systems, IEEE Transactions on*, vol. 20, no. 3, pp. 1447–1455, 2005.

[6] T. Ramachandran, Z. Costello, P. Kingston, S. Grijalva, and M. Egerstedt, “Distributed power allocation in prosumer networks,” in *IFAC Necsys*, 2012.

[7] S. Grijalva, M. Costley, and N. Ainsworth, “Prosumer-based control architecture for the future electricity grid,” in *IEEE Multi-Conference on Systems and Control*, 2011.

[8] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 6, pp. 926–939, 1998.

[9] M. Ji and M. Egerstedt, “Distributed coordination control of multiagent systems while preserving connectedness,” *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, 2007.

[10] A. Jadbabaie, J. Lin, and A. S. Morse, “Coordination of groups of mobile autonomous agents using nearest neighbor rules,” *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[11] H. Tanner, A. Jadbabaie, and G. Pappas, “Stable flocking of mobile agents, part II : Dynamic topology,” in *Proc. 42nd IEEE Conf. Decision Control*, 2003.

[12] N. Michael and V. Kumar, “Controlling shapes of ensembles of robots of finite size with nonholonomic constraints,” in *RSS*, 2008.

[13] K. Romer and F. Mattern, “The design space of wireless sensor networks,” *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 54–61, 2004.

[14] F. Zhang and N. Leonard, “Coordinated patterns of unit speed particles on a closed curve,” *Systems and Control Letters*, vol. 56, no. 6, pp. 397–407, 2007.

[15] D. Marculescu, R. Marculescu, N. H. Zamora, P. Stanley-Marbell, P. K. Khosla, S. Park, S. Jayaraman, S. Jung, C. Lauterbach, W. Weber, *et al.*, “Electronic textiles: A platform for pervasive computing,” *Proceedings of the IEEE*, vol. 91, no. 12, pp. 1995–2018, 2003.

[16] J. Cortés and F. Bullo, “Coordination and geometric optimization via distributed dynamical systems,” *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[17] A. Nedic, A. Ozdaglar, and A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[18] R. Olfati-Saber and J. S. Shamma, “Consensus filters for sensor networks and distributed sensor fusion,” in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*. IEEE, 2005, pp. 6698–6703.

[19] S. Sundaram and C. N. Hadjicostis, “Distributed function calculation and consensus using linear iterative strategies,” *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 4, pp. 650–660, 2008.

[20] M. Rotkowitz and S. Lall, “A characterization of convex problems in decentralized control,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 2, pp. 274–286, 2006.

[21] J. Swigart and S. Lall, “A graph-theoretic approach to distributed control over networks,” in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. IEEE, 2009, pp. 5409–5414.

[22] Z. Costello and M. Egerstedt, “From global linear computations to local interaction rules,” *IEEE Transactions on Automatic Control*, 2013 (Under Submission), a preprint is available at arXiv.

[23] R. Brockett, *Finite Dimensional Linear Systems*. John Wiley & Sons, Inc., 1970.

[24] S. Sastry, *Nonlinear systems: analysis, stability, and control*. Springer New York, 1999, vol. 10.

[25] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

[26] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.