ELSEVIER

Systems & Control Letters 43 (2001) 43–52

SYSTEMS & CONTROL LETTERS

www.elsevier.com/locate/sysconle

An application of Lie groups in distributed control networks

George A. Kantor[a, *, 1], P.S. Krishnaprasad[b]

[a] *The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA*
[b] *Institute for Systems Research, University of Maryland, College Park, MD 20742, USA*

Abstract

Here we introduce a class of linear operators called recursive orthogonal transforms (ROTs) that allow a natural implementation on a distributed control network. We derive conditions under which ROTs can be used to represent SO(n) for $n \geqslant 4$. We propose a paradigm for distributed feedback control based on plant matrix diagonalization. To find an ROT suitable for this task, we derive a gradient flow on the appropriate underlying Lie group. A numerical example is presented.

Keywords: Gradient flow; Distributed control network; Singular value decomposition

1. Introduction

Distributed control networks are rapidly emerging as a viable and important alternative to centralized control. In a typical distributed control network, a number of spatially distributed nodes composed of "smart" sensors and actuators are used to take measurements and apply control inputs to some physical plant. The nodes have embedded processors and the ability to communicate with the other nodes via a network. The challenge is to compute and implement a feedback law for the resulting MIMO system in a distributed manner while respecting the bandwidth limitations of the network.

There is a growing body of work regarding control networks. Many authors have investigated the problem of implementing a centralized controller where the communication link between the sensors, actuators, and controller is a single shared channel. Brockett has investigated the stabilization of a network of intelligent motors [2]. Wong and Brockett have studied the problems of state estimation and feedback control for control networks with limited communication bandwidth [23,24]. Hristu [11] has addressed the problem of finding stabilizing feedback laws for linear systems with limited communication. Wang and Mau [21] and Ooi et al. [17] present results regarding systems with feedback that is subject to a communication delay. Walsh et al. [19,18] and Walsh et al. [20] analyzed and provided control algorithms for single channel networks where some of the control is distributed among the network nodes.

The research on shared single channel control networks is important because it is applicable to existing control network architectures such as CAN. However, we feel that more flexible communication architectures are necessary to take full advantage of the

* Corresponding author.
E-mail address: kantor@ri.cmu.edu (G.A. Kantor).

[1] This research was supported in part by a grant from the Army Research Office under the ODDR&E MURI97 Program Grant No. DAAG55-97-1-0114 to the Center for Dynamics and Control of Smart Structures (through Harvard University) and also by grants from the National Science Foundation's Engineering Research Centers Program: NSFD CDR 8803012, and by a Learning and Intelligent Systems Initiative Grant CMS9720334.

0167-6911/01/$ - see front matter
PII: S0167-6911(01)00091-3

capabilities of distributed control networks. Guenther et al. [4–6] applied learning algorithms to develop controllers for distributed networks employing nearest neighbor, hierarchy, multi-hierarchy, and global communication schemes. Chou et al. [3] implemented the discrete wavelet transform on a multi-hierarchy as part of a distributed controller for a class of flexible mechanical systems.

Our approach to this problem is to employ distributed signal processing in an effort to simplify the control problem. Plant matrix diagonalization is one example of this approach. To do this, we search for basis transformations for the vector of outputs coming from the sensors and the vector of inputs applied to the actuators so that, in the new bases, the MIMO system becomes a collection of decoupled SISO systems. This formulation provides a number of advantages for the synthesis and implementation of a feedback control law, particularly for systems where the number of inputs and outputs is large. Of course, in order for this idea to be feasible, the required basis transformations must have properties which allow them to be implemented on a distributed control network. Namely, they must be computed in a distributed manner which respects the spatial distribution of the data (to reduce communication overhead) and takes advantage of the parallel processing capability of the network (to reduce computation time).

In [13] we introduced the idea of plant matrix diagonalization for distributed control networks using Haar–Walsh wavelet packets. This work relied on the work of Wickerhauser, who developed wavelet–based algorithms for approximate principal component analysis [22]. To implement the resulting transforms in a distributed setting, we exploited the fact that a wavelet packet transform is implemented as an alternating series of orthogonal FIR filtering operations and communication steps. Each communication step rearranges the data vector and can be written as a permutation of the identity matrix. In the case of Haar–Walsh packets, each filtering step can be written as a block diagonal matrix where the diagonal blocks are constant 2×2 orthogonal matrices, each of which can be either the identity matrix or a planar rotation of $\pi/4$.

In this paper, we generalize this notion to allow each filtering step to be a block diagonal matrix with general orthogonal matrices along the diagonal. We define a class of transforms called recursive orthogonal transforms (ROTs). Simply stated, any transform that can be written as a product of alternating piecewise rotations and permutation matrices is an ROT. We show by example how the structure of an ROT can be chosen to allow a naturally distributed implementation on a control network. We then derive a gradient flow which can be integrated to find piecewise rotations so that the ROT most nearly diagonalizes a constant, real-valued, symmetric plant matrix. Finally, we demonstrate this idea with a numerical example.

2. Distributed signal processing

The main objective of this paper is to develop distributed signal processing techniques to aid the design and implementation of feedback controllers for plants with distributed control networks. The set of orthogonal transforms contains a large collection of important signal processing tools such as wavelets and principal component analysis. Additionally, the norm preserving property of orthogonal transforms makes them nice candidates for general data transformations. For these reasons, a distributed implementation of general orthogonal transforms would be useful for signal processing on a distributed control network. In this section, we demonstrate that the ROT provides just such an implementation.

Definition 1. A *recursive orthogonal transform* (*ROT*) is a linear operator of the form

$$\tilde{\Theta} = P_1 \Theta_1 P_2 \Theta_2 \cdots P_L \Theta_L, \tag{1}$$

for some L, where for each $i = 1, 2, \ldots, L$,

(1) The matrix Θ_i is of the form

$$\Theta_i = \begin{bmatrix} \theta_1^i & 0 & \cdots & 0 \\ 0 & \theta_2^i & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \theta_{m_i}^i \end{bmatrix}, \tag{2}$$

where $\theta_j^i \in \mathrm{SO}(n_{ij})$, $\sum_{j=1}^{m_i} n_{ij} = n$, and the 0's represent appropriately dimensioned blocks of zeros.

(2) Each P_i is a permutation of the $n \times n$ identity matrix, $\prod_{i=1}^{L} \det(P_i) = 1$.

The integer L is called the *depth* of the ROT. The structure of the ROT is defined by the parameters L, P_i, m_i, and n_{ij} for $i = 1, 2, \ldots, L$, $j = 1, 2, \ldots, m_i$. These quantities are collectively called the ROT *configuration*. The Θ_i, $i = 1, 2, \ldots, L$, are called the ROT *variables*. The set of all possible ROTs of a given configuration,

$\{\tilde{\Theta} \mid \theta_j^i \in \mathrm{SO}(n_{ij}),\ i = 1,2,\ldots,L,\ j = 1,2,\ldots,m_i\}$, is called an ROT *family*.

Because of its block diagonal structure, each Θ_i can be implemented as m_i parallel operations on separate processors. The permutation matrices represent a reordering of the data and can be implemented as communication between the nodes. Hence, an ROT has a natural implementation as a sequence of alternating decentralized computation and communication steps. This notion will be made more concrete in the following sections.

In the definition, we have instituted the constraint that $\prod_i \det(P_i) = 1$. This is to ensure that the ROT is a member of the special orthogonal group. In practice, this constraint is of little consequence. If we have a collection of permutation matrices $\{P_1, P_2, \ldots, P_L\}$ that are compatible with the communication architecture of the network but do not satisfy the constraint, we can simply add another permutation P_{L+1} to the end of the product on the RHS of Eq. (1) to ensure that the ROT has positive determinant.

2.1. ROTs for linear arrays

Here we present an ROT that is configured to be implemented on a linear array of smart sensor/actuator pairs using only nearest neighbor communication. This specific example is intended to demonstrate the naturally distributed implementation admitted by an ROT. Configuration parameters can be also chosen to address a more general set of network connections and communication patterns. The results in this paper apply to general ROTs as well as the specific subclass presented in this section.

Consider a transform $\tilde{\Theta}$ of the form given by Eq. (1) where L is a fixed odd integer, n is even, $n_{ij}=2$ for each $i = 1,2,\ldots,L,\ j = 1,2,\ldots,n/2$, and the permutation matrices are given as

$$P_i = \begin{cases} \mathbb{I}_{n\times n} & \text{if } i = 1, \\ P_{\mathrm{e}} & \text{if } i \text{ is even}, \\ P_{\mathrm{o}} & \text{if } i \text{ is odd},\ i \neq 1, \end{cases}$$

where

$$P_{\mathrm{e}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

$$P_{\mathrm{o}} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}.$$

This configuration can be used to transform the output vector of a distributed control network composed of a linear array of smart sensors. The sensors, or nodes, are indexed from left to right. The value of the ith sensor (i.e. the ith element of the output vector) is known only to the ith node. Each node has a communication link to its left and right neighbors. The nth node is the left neighbor of the first node. We compute the transformation $\tilde{y} = \tilde{\Theta}^{\mathrm{T}} y$ in a sequence of levels. For convenience, we define $\tilde{\Phi} \triangleq \tilde{\Theta}^{\mathrm{T}}$ and note that $\tilde{\Phi}$ can be written as $\tilde{\Phi} = \Phi_L P_{\mathrm{e}} \cdots \Phi_3 P_{\mathrm{e}} \Phi_2 P_{\mathrm{o}} \Phi_1$, where $\Phi_i = \Theta_i^{\mathrm{T}},\ i = 1,2,\ldots,L$.

In the first level, the intermediate vector $a = \Phi_1 y$ is computed. To accomplish this, each odd node sends the value to the even node on its right. Since the matrix Φ_1 is block diagonal with 2×2 blocks, a can be computed in a decentralized way on the processors on the even nodes.

In the second level, the intermediate vector $c = \Phi_2 P_{\mathrm{o}} a$ is computed. This level is composed of two steps: communication and piecewise rotation. After the first level is completed, the ith node, even i, contains the quantities a_{i-1} and a_i. To start the second level, each even node i sends the value a_{i-1} to the node on its left and it sends a_i to the node on its right. This communication step creates the intermediate vector b, which is just a right circular shift of the vector a. Hence, the communication step implements the permutation operation $b = P_{\mathrm{o}} a$. Now each ith node, i odd, contains the quantities b_i and b_{i+1}. Since Φ_2 is block diagonal, the vector $c = \Phi_2 b = \Phi_2 P_{\mathrm{o}} \Phi_1 y$ can be computed in a decentralized way on the odd nodes. At the end of the second level, each ith node, i odd, contains the values c_i and c_{i+1}.

The third level also begins with a communication step. Each ith node, i odd, passes the values c_i and c_{i+1} to the nodes on its left and right, respectively. This is equivalent to a left circular shift of the vector c, which results in the vector $d = P_{\mathrm{e}} c$. The elements of d are distributed among the even nodes of the array, where the intermediate vector $e = \Phi_3 d = \Phi_3 P_{\mathrm{e}} \Phi_2 P_{\mathrm{o}} \Phi_1 y$ is computed in a decentralized way. This process of

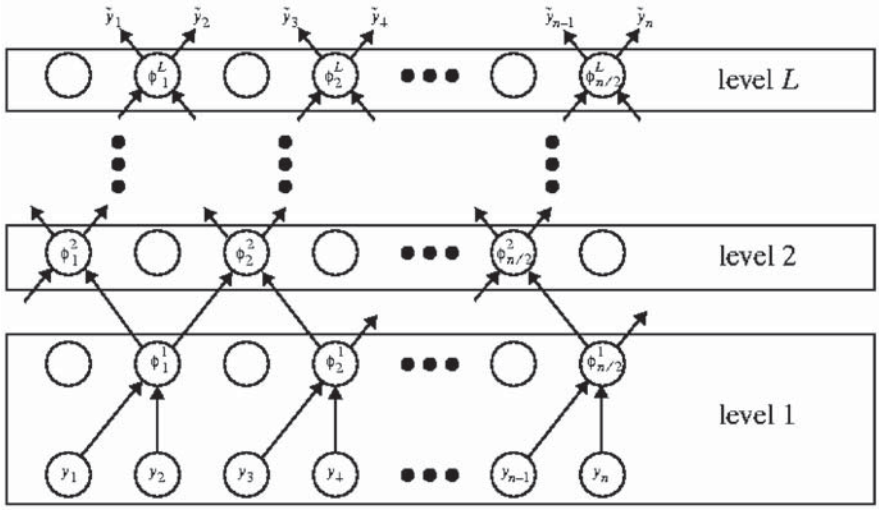

Fig. 1. Graphical description of the implementation of an ROT on a linear array of smart sensors. The raw data vector $y=[y_1, y_2, \ldots, y_n]^{\mathrm{T}}$ starts at the bottom of the diagram and is processed in levels moving upwards.

permutations and piecewise rotations is repeated for L levels. The transformed vector $\tilde{y}$ is the output of the Lth level. This process is depicted in Fig. 1.

2.2. Plant diagonalization for control

Plant diagonalization represents one way in which distributed signal processing with ROTs can be used to simplify the problem of designing and implementing feedback controllers for large distributed control networks. Consider a system given by the input–output description $y=H_0u$, where $y \in \mathbb{R}^n$ is the output vector, $u \in \mathbb{R}^n$ is the input vector, and $H_0 = H_0^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ is the plant matrix. Assume for now that an ROT $\tilde{\Theta}$ exactly diagonalizes H_0, i.e. $H = \tilde{\Theta}^{\mathrm{T}} H_0 \tilde{\Theta}$ is a diagonal matrix. Define the transformed input and output vectors, $\tilde{y} = \tilde{\Theta}^{\mathrm{T}} y$ and $\tilde{u} = \tilde{\Theta}^{\mathrm{T}} u$, respectively. Viewing the system in these new co-ordinates, we have $\tilde{y} = H\tilde{u}$. Since H is diagonal, the problem of synthesizing a MIMO controller is greatly simplified in the new co-ordinates. Since the co-ordinate transforms are ROTs, the resulting controller is easily implemented on a distributed control network: an ROT transforms the data vector into the new co-ordinates in a distributed manner; the control $\tilde{u}_i$ is chosen for each element y_i of the transformed output vector; and another ROT transforms $\tilde{u}$ into the actual control vector u in a distributed manner such that each element of u resides on the node containing the actuator to which it is to be applied.

The concept of the distributed feedback control that results from the use of ROTs for plant diagonalization is appealing, but we do not aim to overemphasize the importance of this idea in the context of this paper. Clearly, systems that can be modeled as symmetric, real-valued plant matrices are of limited interest. Still, the results generated here are important as a "first step" that can be extended to plant diagonalization for more general systems. In fact, we have developed an extension that can be used to approximately diagonalize a complex-valued, non-symmetric matrix and demonstrated how it can be used to control the resonances of flexible mechanical systems [12]. In the same work, we also developed an ROT capable of approximate diagonalization of dynamic plants that possess a spatial invariance property.

2.3. Theoretical intuition

For a fixed configuration, the ROT family gives a set of candidate representations of the members of SO(n). Obviously, if the number of degrees of freedom in the underlying variable space is less than the dimension of SO(n), the family cannot represent all of SO(n). In these cases, a member of SO(n) is approximated by the "nearest" member of SO(n) which can be exactly represented by a member of the family.

It is not obvious that an ROT configuration family that has a variable space with dimension greater than or equal to that of SO(n) can be used to represent all of SO(n). Our intuition is that it can be done for a wisely chosen set of permutation matrices. Here we back up this intuition with some theoretical results that

can be used to choose ROT configurations capable of representing all of SO(n).

Theorem 2. *Let* $\tilde{\Theta} = \Theta_1 P_2 \Theta_2 P_3 \Theta_3$ *be an* $(n+m)\times(n+m)$ *depth-3 ROT, where*

$$\Theta_i = \begin{bmatrix} \theta_1^i & 0_{n\times m} \\ 0_{m\times n} & \theta_2^i \end{bmatrix}, \tag{3}$$

where $\theta_1^i \in \mathrm{SO}(n)$ *and* $\theta_2^i \in \mathrm{SO}(m)$, *for* $i \in \{1,2,3\}$, *where both m and n are greater than or equal to* 2. *Define* $\mathfrak{k} \subset \mathfrak{so}(n+m)$ *as*

$$\mathfrak{k} = \left\{ \begin{bmatrix} \Omega_1 & 0_{n\times m} \\ 0_{m\times n} & \Omega_2 \end{bmatrix} \,\middle|\, \Omega_1 \in \mathfrak{so}(n), \Omega_2 \in \mathfrak{so}(m) \right\}. \tag{4}$$

Let $\mathfrak{p}$ *be the complement of* $\mathfrak{k}$ *in* $\mathfrak{so}(n+m)$ *and let* $\mathfrak{a}$ *be a maximal Abelian subalgebra of* $\mathfrak{p}$. *Let permutation matrices* $P_2 = P_3^{\mathrm{T}}$ *be such that*

$$\mathfrak{a} \subset P_2 \mathfrak{k} P_2^{\mathrm{T}}. \tag{5}$$

Then given any $g \in \mathrm{SO}(n+m)$, *there exist* Θ_1, Θ_2, *and* Θ_3 *of the form given by Eq.* (3) *such that* $\tilde{\Theta} = g$.

To prove this theorem, we use the theory of symmetric subalgebras. We summarize the result we need and state it as a theorem without proof. A complete discussion can be found in Hermann [9].

Theorem 3. *Let G be a connected, semisimple Lie group with finite center and Lie algebra* $\mathfrak{g}$. *Let* $\mathfrak{k}$ *be a symmetric subalgebra and let* $\mathfrak{p}$ *be the complement of* $\mathfrak{k}$ *in* $\mathfrak{g}$, *i.e.* $[\mathfrak{k},\mathfrak{k}] \subset \mathfrak{k}$, $[\mathfrak{k},\mathfrak{p}] \subset \mathfrak{p}$, *and* $[\mathfrak{p},\mathfrak{p}] \subset \mathfrak{k}$. *Let* $\mathfrak{a}$ *be a maximal Abelian subalgebra of* $\mathfrak{p}$. *Then any* $g \in G$ *can be written*

$$g = \exp(X_{k1}) \exp(X_a) \exp(X_{k2}), \tag{6}$$

where $X_{k1}, X_{k2} \in \mathfrak{k}$ *and* $X_a \in \mathfrak{a}$.

Proof of Theorem 2. We first note that SO(p), $p = n+m$, is a semisimple Lie group with finite center. It is tedious but not difficult to check the bracket conditions listed in Theorem 3 to see that $\mathfrak{k}$, as defined in Eq. (4), is a symmetric subalgebra of $\mathfrak{so}(p)$. The Θ_i reside in the compact Lie subgroup of SO(p) whose Lie algebra is $\mathfrak{k}$. The dimension of $\mathfrak{k}$ is equal to the dimension of $\mathfrak{so}(n) \oplus \mathfrak{so}(m)$, which is $d_k = (n(n-1)+m(m-1))/2$. Hence, we can write

$$\Theta_i = \exp\left(\sum_{j=1}^{d_k} \alpha_j^i A_j \right),$$

where $\{A_i \mid i = \{1,2,\ldots,d_k\}\}$ forms an orthogonal basis of $\mathfrak{k}$. Substituting into the original expression for $\tilde{\Theta}$ we have

$$\begin{aligned} \tilde{\Theta} &= \exp\left(\sum_{j=1}^{d_k} \alpha_j^1 A_j \right) P_2 \exp\left(\sum_{j=1}^{d_k} \alpha_j^2 A_j \right) P_2^{\mathrm{T}} \\ &\quad \times \exp\left(\sum_{j=1}^{d_k} \alpha_j^3 A_j \right) \\ &= \exp\left(\sum_{j=1}^{d_k} \alpha_j^1 A_j \right) \exp\left(\sum_{j=1}^{d_k} \alpha_j^2 P_2 A_j P_2^{\mathrm{T}} \right) \\ &\quad \times \exp\left(\sum_{j=1}^{d_k} \alpha_j^3 A_j \right). \end{aligned} \tag{7}$$

We know that $\mathrm{span}\{P_2 A_j P_2^{\mathrm{T}} \mid j \in \{1,2,\ldots,d_k\}\}$ contains a maximal Abelian subalgebra $\mathfrak{a} \subset \mathfrak{p}$. So for any $X_a \in \mathfrak{a}$, we can choose $\{\alpha_j^2 \mid j \in \{1,2,\ldots,d_k\}\}$ so that

$$\sum_{j=1}^{d_k} \alpha_j^2 P_2 A_j P_2^{\mathrm{T}} = X_a.$$

Hence, Eq. (7) is equivalent to Eq. (6) and Theorem 3 can be invoked to finish the proof. □

In order to apply Theorem 2, it is necessary to find permutation matrices such that Eq. (5) is satisfied. Sufficient conditions for the existence of such a P are given by the following Lemma, which is stated here without proof:

Lemma 4. *Let* $\mathfrak{k} \subset \mathfrak{so}(n+m)$ *be*

$$\mathfrak{k} = \left\{ \begin{bmatrix} \Omega_1 & 0_{n\times m} \\ 0_{m\times n} & \Omega_2 \end{bmatrix} \,\middle|\, \Omega_1 \in \mathfrak{so}(n), \Omega_2 \in \mathfrak{so}(m) \right\}, \tag{8}$$

where m and n are both greater than or equal to 2 *and* $\min(m,n)$ *is even. Let* $\mathfrak{p}$ *be the complement of* $\mathfrak{k}$ *in* $\mathfrak{so}(n+m)$ *and let* $\mathfrak{a}$ *be a maximal Abelian subalgebra of* $\mathfrak{p}$. *Then there exists a permutation matrix P such that* $\mathfrak{a} \subset P \mathfrak{k} P^{\mathrm{T}}$.

Theorem 2 can be applied to show that a 4×4 depth-3 ROT with two-dimensional piecewise rotations and 4×4 permutation matrices $P_2 = P_{\mathrm{e}}$ and $P_3 = P_{\mathrm{o}} = P_{\mathrm{e}}^{\mathrm{T}}$ as defined in Section 2.1 can be used to

represent any element of SO(4). The basis vectors of $\mathfrak{k}$ are defined as

$$A_1 \triangleq \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad A_2 \triangleq \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

Further define A_3 and A_4 to be

$$A_3 \triangleq P_e A_1 P_e^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

and

$$A_4 \triangleq P_e A_2 P_e^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{11}$$

It is easy to verify that $\mathfrak{a} \triangleq \operatorname{span}\{A_3, A_4\}$ is a maximal Abelian subalgebra of $\mathfrak{p}$, so Theorem 2 can be applied.

Theorem 2 can be applied in a recursive manner to obtain results for ROTs with more than 3 levels. For example, we can use Theorem 2 once to get full representation of SO(8) with an 8×8 depth-3 ROT $\tilde{\Theta} = \Phi_1 P_1 \Phi_2 P_1^{\mathrm{T}} \Phi_3$ where each Φ_i is of the form

$$\Phi_i = \begin{bmatrix} \phi_1^i & 0_{4\times 4} \\ 0_{4\times 4} & \phi_2^i \end{bmatrix}, \tag{12}$$

with $\phi_j^i \in \mathrm{SO}(4)$. Then Theorem 2 can be invoked again to allow 4×4 depth-3 ROTs to represent each of the ϕ_j^is. The result in this example is that an 8×8 depth-9 ROT with two-dimensional piecewise rotations can be used to represent all of SO(8).

3. A flow for distributed diagonalization

In the previous section we introduced and defined the ROT, showed that a transform in the form of an ROT has a natural implementation on a distributed control network, and provided some theoretical intuition arguing that ROTs can be used to represent or approximate general orthogonal transforms. In this section we show how to find an ROT to accomplish the task of approximate plant matrix diagonalization. Given a fixed ROT configuration, we show how to find the ROT variables that most nearly diagonalize a given real-valued, symmetric plant matrix.

The objective of this section is stated as follows: Given a symmetric $n \times n$ matrix H_0 and configuration parameters for the ROT $\tilde{\Theta} = P_1 \Theta_1 P_2 \Theta_2 \cdots P_L \Theta_L$, find $\Theta_1, \Theta_2, \ldots, \Theta_L$ such that the matrix

$$H = \tilde{\Theta}^{\mathrm{T}} H_0 \tilde{\Theta}, \tag{13}$$

is, in some sense, most nearly diagonalized. Our first step toward solving this problem is to find a "diagonalness" functional $\phi(H)$. Then we search for the $(\Theta_1, \Theta_2, \ldots, \Theta_L)$ which minimize ϕ by flowing along the gradient vector field $\nabla\phi$ on the configuration space of the Θ_i's. This idea is motivated by Brockett [1], who showed that the matrix diagonalization problem can be solved by integrating an ODE which evolves on the orthogonal group.

Let Θ_k, $k = 1, 2, \ldots, L$, be as described in Eq. (2) Each Θ_k is a block diagonal matrix where the blocks on the diagonal are orthogonal matrices. Let $M_k \triangleq \mathrm{SO}(n_{k1}) \times \mathrm{SO}(n_{k2}) \times \cdots \times \mathrm{SO}(n_{km_k})$. Then Θ_k belongs to the Lie subgroup $M_k \subset \mathrm{SO}(n)$.

The Lie algebra of $\mathrm{SO}(\ell)$ is $\mathfrak{so}(\ell) = \{\Omega \in \mathbb{R}^{\ell\times\ell} \mid \Omega^{\mathrm{T}} = -\Omega\}$. The Lie algebra of M_k is the tangent space at the identity $e \in M_k$,

$$T_e M_k = \mathfrak{so}(n_{k1}) \oplus \mathfrak{so}(n_{k2}) \oplus \cdots \oplus \mathfrak{so}(n_{km_k}) \tag{14}$$

and a vector in $T_e M_k$ can then be written

$$\Omega_k = \begin{bmatrix} \omega_1^k & 0 & \cdots & 0 \\ 0 & \omega_2^k & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \omega_{m_k}^k \end{bmatrix}, \tag{15}$$

where $\omega_j^k \in \mathfrak{so}(n_{kj})$ for $j = 1, 2, \ldots, m_k$. Hence the tangent space to M_k at the point Θ_k is $T_{\Theta_k} M_k = \{\Theta_k \Omega_k \mid \Omega_k \in T_e M_k\}$.

Define $\Psi \in M \triangleq M_1 \times M_2 \times \cdots \times M_L$ to be the ordered L-tuple of Θ_ks,

$$\Psi = (\Theta_1, \Theta_2, \ldots, \Theta_L). \tag{16}$$

Using this notation, the matrix $H = \tilde{\Theta}^{\mathrm{T}} H_0 \tilde{\Theta}$ is a function of Ψ and will be written in sequel as $H(\Psi)$. Let X denote a vector in $T_\Psi M$,

$$X = (\Theta_1 \Omega_1, \Theta_2 \Omega_2, \ldots, \Theta_L \Omega_L). \tag{17}$$

We also define two sequences of recursive orthogonal transforms:

$$\tilde{\Theta}_k \triangleq \prod_{\ell=1}^{k} P_\ell \Theta_\ell, \qquad \check{\Theta}_k \triangleq \prod_{\ell=k+1}^{n} P_\ell \Theta_\ell, \tag{18}$$

$k = 1, 2, \ldots, L$. Here the product symbol $\prod$ denotes multiplication with indices ascending from left to right. Also, $\prod_{k=a}^{b}(\cdot) = \mathbb{I}_{n\times n}$ for $b < a$.

Let N be a fixed diagonal $n \times n$ matrix with distinct values along the diagonal. We can now define a cost function using the distance between $H(\Psi)$ and N given by the Frobenius norm, $\|A\|^2 = \mathrm{tr}(A^{\mathrm{T}}A)$. Simple matrix manipulation reveals

$$\|N - H(\Psi)\|^2 = \|N\|^2 + \|H(\Psi)\|^2 - 2\mathrm{tr}(NH(\Psi)). \tag{19}$$

The norms $\|N\|^2$ and $\|H(\Psi)\|^2 = \|\tilde{\Theta}^{\mathrm{T}} H_0 \tilde{\Theta}\|^2$ are both constant. The Frobenius distance between H and N is minimized when

$$\phi(\Psi) = \mathrm{tr}(NH(\Psi)) \tag{20}$$

is maximized. The function $\phi : M \to \mathbb{R}$ can be thought of as a measure of the "diagonalness" of $H(\Psi)$.

A Riemannian metric for TM can be inherited from the space of $n \times n$ real matrices. For each $\Psi \in M$ we define $\langle \cdot, \cdot \rangle_\Psi : T_\Psi M \times T_\Psi M \to \mathbb{R}$ to be

$$\begin{aligned} &\langle X_1, X_2 \rangle_\Psi \\ &\quad = \langle (\Theta_1 \Omega_1^1, \ldots, \Theta_L \Omega_L^1), (\Theta_1 \Omega_1^2, \ldots, \Theta_L \Omega_L^2) \rangle_{\mathbb{R}^{n\times n}} \\ &\quad = -\sum_{k=1}^{L} \mathrm{tr}(\Omega_k^1 \Omega_k^2). \end{aligned} \tag{21}$$

Definition 5 (*Projection operators*). For each $k = 1, 2, \ldots, L$, the operator $\Pi_k : T_e\mathrm{O}(n) \to T_e(M_k)$ is defined to project from the set of skew-symmetric matrices to the set of block diagonal skew-symmetric matrices by setting all off-diagonal blocks to zero.

Theorem 6. *The ascent direction gradient flow of the diagonalness function ϕ defined in Eq.* (20) *using the Riemannian metric defined in Eq.* (21) *is*

$$\begin{aligned} &\dot{\Theta}_k = -\Theta_k \Pi_k [\check{\Theta}_k N \check{\Theta}_k^{\mathrm{T}}, \tilde{\Theta}_k^{\mathrm{T}} H_0 \tilde{\Theta}_k], \\ &\Theta_k(0) = \Theta_{k0}, \end{aligned} \tag{22}$$

for $k = 1, 2, \ldots, L$, where $[\cdot, \cdot]$ denotes the matrix Lie bracket, $[A, B] = AB - BA$.

The proof of this theorem consists of the straightforward but tedious verification that $\nabla\phi(\Psi) = -(\dot{\Theta}_1, \dot{\Theta}_2, \ldots, \dot{\Theta}_L)$ has the following properties:

(1) $\nabla\phi(\Psi) \in T_\Psi M \quad \forall \Psi \in M$.
(2) $D\phi_\Psi(X) = \langle \nabla\phi(\Psi), X \rangle \quad \forall X \in T_\Psi M$.

For a given metric, the $\nabla\phi(\Psi)$ that satisfies these properties is the unique gradient vector field. The complete proof can be found in [12].

From the properties of gradient flows on compact manifolds we know that the solution to Eq. (22) exists for all time and converges to the set of equilibria for the flow. One can say more, noting that the function defining the gradient flow is analytic. In this case a result of Lojasiewicz [14] can be used to prove that the gradient flow converges to a specific equilibrium and not just the set. While this may be known to some as a type of "folk theorem" [10], there does not seem to have been a proof written down along these lines until the work of Mahony [15,16]. The question of which equilibrium point the flow converges to remains open. The numerical results we have obtained are promising, however, as is demonstrated by the following example.

4. Numerical example

Here we show the results of applying this technique to approximately diagonalize a 16×16 symmetric matrix. This can be thought of as the plant matrix of a linear array composed of 16 sensor/actuator pairs where the coupling between any two pairs is equal to the inverse of the distance between them. We use the ROT configuration presented in Section 2.1. The original plant matrix is shown in Fig. 2(a), where the intensity of the (i, j)th pixel corresponds to the value of the (i, j)th element of H_0. The approximately diagonalized plant matrix for an ROT with depth 11 is shown in Fig. 2(b).

Fig. 3 plots approximation error as a function of ROT depth. Here, the approximation error is defined to be

$$E(\tilde{\Theta}) = \frac{\|Q^{\mathrm{T}} H_0 Q - \tilde{\Theta}^{\mathrm{T}} H_0 \tilde{\Theta}\|}{\|H_0\|}, \tag{23}$$

where Q is a matrix whose columns are the unit eigenvectors of H_0.

The dimension of $\mathrm{SO}(n)$ is $n(n-1)/2$. The number of degrees of freedom in the ROTs being considered is $Ln/2$, where L is the depth of the transform. Intuitively,

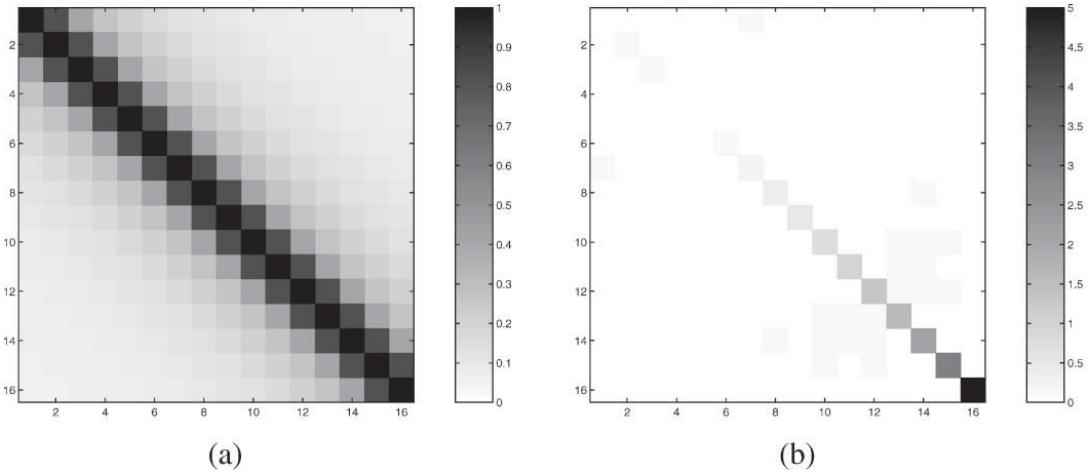

Fig. 2. (a) 16×16 plant matrix, H_0, corresponds to a linear array of sensor/actuator pairs with one-over-distance coupling. (b) Approximately diagonalized matrix $H=\tilde{\Theta}^{\mathrm{T}}H_0\tilde{\Theta}$, where $\tilde{\Theta}$ is an ROT with depth 11.

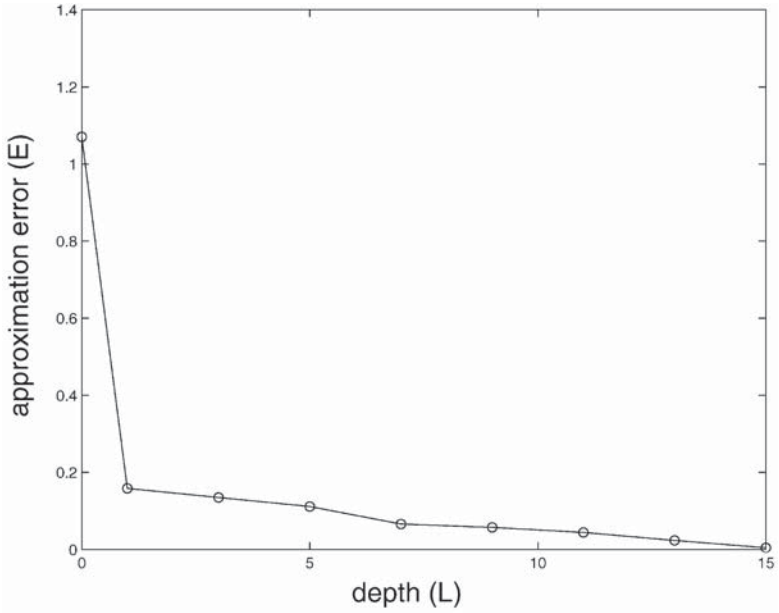

Fig. 3. Plot of approximation error versus ROT depth.

when $L=n-1$ the ROT "should" have enough degrees of freedom to represent any Θ in SO(n). These results seem to support this notion since the approximation error gets very close to zero for $L=15$.

We have conducted similar numerical studies for a variety of other symmetric matrices. Specific examples included plant matrices derived from flexible cantilever beams and thin flexible membranes, as well as symmetric matrices with random entries. In every case, the ROT variables that resulted from integrating the gradient flow seemed to produce good answers. And as the number of degrees of freedom in the ROT approached the dimension of the corresponding orthogonal group, the approximation error approached zero.

5. Conclusions

We have introduced the ROT, demonstrated that it admits a natural implementation on a distributed control network, and derived a gradient flow to accomplish approximate diagonalization of a real symmetric matrix. We conclude the paper by discussing interesting possibilities for future research.

One problem which has already been addressed is approximate singular value decomposition using ROTs [12]. This extension to the work in Section 3 is motivated by the work of Helmke and Moore [7,8], who extended Brockett's work on symmetric matrix diagonalization to address SVD.

Important questions remain regarding the selection of the parameters of the ROT and the flow. The results in this Section 2.3 provide a means of configuring ROTs that are guaranteed to provide representations of the special orthogonal group. In our experience these results are more conservative than necessary. For example, the ROT configuration developed for linear arrays in Section 2.1 does not satisfy the conditions of Theorem 2. However, our numerical studies seem to show that this ROT configuration can represent $SO(n)$ when the underlying variable space is high enough. A better understanding of this relationship is required so that the permutation matrices can be chosen intelligently. Intuitively, as the dimension of the variable space is increased by increasing the ROT depth and the size of the diagonal blocks, approximation error for a general member of $SO(n)$ should get smaller. The development of error bounds as a function of depth and block size would be a useful tool for the design of ROTs. The role of the diagonal matrix N used in the cost function is another important aspect of this work that is currently not well understood.

Techniques that improve the convergence properties of the gradient flow need to be developed. Clever numerical integration schemes and iterative methods may be able to improve the rate of convergence. And the adaptation of optimization techniques such as simulated annealing may help the gradient flow converge to better local maxima.

In the currently envisioned application of ROTs, the variables are found off-line and resulting ROT is implemented in a manner similar to the example in Section 2.1. An algorithm for the integration of the ODEs given in Eq. (22) on a distributed control network would enable on-line computation of the ROT variables. This would allow the network to continuously adapt to a changing environment.

References

[1] R.W. Brockett, Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems, in: Proceedings of the 1988 IEEE Conference on Decision and Control, Linear Algebra and Its Applications 1988, pp. 799–803.

[2] R.W. Brockett, Stabilization of motor networks, in: Proceedings of the 34th Conference on Decision and Control, December 1995, pp. 1484–1488.

[3] K. Chou, G. Guthart, D. Flamm, A multiscale approach to the control of smart materials, in: Proceedings of the SPIE Conference on Smart Structures and Materials, Vol. 2447, 1995, pp. 249–263.

[4] O. Guenther, T. Hogg, B.A. Huberman, Controls for unstable structures, in: Proceedings of SPIE Conference on Mathematics and Control in Smart Structures, March 1997, pp. 754–763.

[5] O. Guenther, T. Hogg, B.A. Huberman, Learning in multiagent control of smart matter, in: AAAI Workshop on Multiagent Learning, 1997.

[6] O. Guenther, T. Hogg, B.A. Huberman, Market organizations for controlling smart matter, Technical Report, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, 1997.

[7] U. Helmke, J.B. Moore, Singular-value decomposition via gradient and self-equivalent flows, Linear Algebra Appl. 169 (1992) 223–248.

[8] U. Helmke, J.B. Moore, Optimization and Dynamical Systems, Springer, Berlin, 1994.

[9] R. Hermann, Lie Groups for Physicists, W.A. Benjamin, Inc., New York, 1966.

[10] M. Hirsch, Private Communication with P.S. Krishnaprasad, 1996.

[11] D. Hristu, Optimal control with limited communication, Ph.D. Thesis, Harvard University, Cambridge, MA, 1999.

[12] G.A. Kantor, Approximate matrix diagonalization for use in distributed control networks, Ph.D. Thesis, University of Maryland at College Park, 1999 (available as Institute for Systems Research Technical Report Ph.D. 99-7 at http://www.isr.umd.edu).

[13] G.A. Kantor, P.S. Krishnaprasad, Efficient implementation of controllers for large scale linear systems via wavelet packet transforms, in: Proceedings of the 32nd Conference on Information Sciences and Systems, 1998, pp. 52–56.

[14] S. Lojasiewicz, Ensembles semi-analytiques, Technical Report, Institut des Hautes Etudes Scientifiques, Bures-sur-Yvette (Sein-et-Oise), France, 1965.

[15] R. Mahony, Convergence of gradient flows and gradient descent algorithms for analytic cost functions, in: A. Beghi, L. Finesso, G. Picci (Eds.), Proceedings of the International Symposium of Mathematical Theory of Networks and Systems, 1998, pp. 653–656.

[16] R. Mahony, B. Andrews, Convergence of the iterates of descent methods for analytic cost function, SIAM Journal of Control and Optimization, to appear.

[17] J.M. Ooi, S.M. Verbout, J.T. Ludwig, G.W. Wornell, A separation theorem for periodic sharing information patterns in decentralized control, IEEE Trans. Automat. Control 42 (11) (1997) 1546–1550.

[18] G.C. Walsh, O. Beldiman, L. Bushnell, Asymptotic behavior of networked control systems, in: Proceedings of the 1999 Conference on Control Applications, 1999, pp. 1448–1453.

[19] G.C. Walsh, O. Beldiman, L. Bushnell, Error encoding algorithms for networked control systems, in: Proceedings of the 1999 IEEE Conference on Decision and Control, 1999, pp. 4933–4938.

[20] G.C. Walsh, H. Ye, L. Bushnell, Stability analysis of networked control systems, in: Proceedings of the American Control Conference, June 1999, pp. 2876–2880.
[21] W.J. Wang, L.G. Mau, Stabilization and estimation for perturbed discrete time-delay large-scale systems, IEEE Trans. Automat. Control 9 (1997) 1277–1282.
[22] M.V. Wickerhauser, Large-rank approximate principal component analysis with wavelets for signal feature discrimination and the inversion of complicated maps, J. Chem. Inform. Comput. Sci. 34 (1993) 1036–1046.
[23] W.S. Wong, R.W. Brockett, Systems with finite communication bandwidth constraints I: state estimation problems, IEEE Trans. Automat. Control 42 (1997) 1294–1299.
[24] W.S. Wong, R.W. Brockett, Systems with finite communication bandwidth constraints II: stabilization with limited information feedback, IEEE Trans. Automat. Control 44 (1999) 1049–1053.