A Scalable Localized Approach to Filtering and Robust, Adaptive Controller Synthesis for Distributed Networked Systems

SooJean Han and John C. Doyle*

February 13, 2020

Abstract

This manuscript is intended to thoroughly survey a handful important applications and extensions to the recently proposed System Level Synthesis (SLS) framework for synthesizing localized and distributed controllers for large-scale networks of systems. First, the closely-related filtering problem is studied in the context of SLS, giving rise to the localized and distributed Kalman filtering (LDKF) method, which is essentially a scalable implementation of the celebrated Kalman filter. Second, an extension is made to robust and adaptive SLS, which designs controllers which are capable of stabilizing the network despite parametric uncertainties and/or dynamically-changing network topologies is discussed. Moreover, due to the robust nature of the control design scheme, learning these uncertainties precisely (i.e., the exact value of a parametric uncertainty or the exact locations of network connectivity changes) is not necessary to stabilize the overall network. Rather, the uncertainty sets are reduced over time through an adaptation scheme, but they are reduced only until there exists a robust controller which can stabilize the system for all values within the current uncertainty set.

1 Introduction

System Level Synthesis (SLS) is a unified framework for the controller synthesis of linear, discrete-time networked systems subject to realistic communication delay constraints. What makes the SLS approach appealing is its shift of the synthesis task from designing controllers to instead designing the entire closed-loop response, which provides important advantages such as scalability to larger networks.

Many extensions to SLS and its applications have arisen since [28]. Localized and distributed Kalman filtering (LDKF) was presented in [30], as an extension of the traditional centralized Kalman filtering

*SJ. Han and J. C. Doyle are with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA. Corresponding email: soojean@caltech.edu

method to networked systems of large scale. Employing the localized implementation of SLS, the LDKF method is able to design filters for large-scale networks of systems, a task which may not even be tractable using the traditional Kalman filtering method. Furthermore, LDKF is able to design such filters with performance comparable to that of the centralized Kalman filter (i.e., the accuracy of the estimate is the same between the two approaches when tested on networks where both methods are computationally tractable). SLS has additionally been developed into a localized, distributed robust control scheme for large-scale networks which additionally adapts to two separate types of uncertainties. First, each system estimates parametric uncertainties for its own local region independently of other systems in the network, allowing for a distributed implementation that can be mostly parallelized. Second, each system accounts for uncertainties in the topological structure of the network; this subsequently leads to an iterative scheme in which the system is able to handle multiple topological modifications over time.

In (the second half of) this manuscript, we focus on surveying the work on these two major extensions of the SLS framework and incorporate new, recent results for each respective topic. The first half of the paper will be dedicated to surveying relevant results from systems level synthesis, distributed control, graph theory, and consensus algorithms; past work related to each of the two extensions will also be discussed.

1.1 Previous and Related Work

System Level Synthesis: A parameterization of the set of internally-stabilizing controllers served as te basis for many controller synthesis methods. To achieve this in the case of linear time-invariant systems, the seminal work of [32] introduced the Youla parameterization approach, and [26] introduced the related stable factorization approach. Parameterizations such as these allow us to formulate the synthesis procedure as convex optimization problems [3] or linear programming problems [8]. This further allows us to incorporate miscellaneous controller design specifications by imposing them as additional constraints on the optimization problem (as long as they are convex). This ability to synthesize controllers through a principled optimization problem rather than a loop-at-a-time tuning process ultimately made way for foundational work on robust control and optimal control [12].

As present-day systems grow larger in scale with a rising amount of computational power, the degree of complexity in system dynamics and their interactions which the environment are also increasing. As such, the mentioned optimization problem for controller synthesis, which were typically implemented as a centralized procedure, may no longer be tractable to solve. This motivates the system-level parameterization, which essentially performs synthesis for the entire closed-loop response instead of solely the controller. The subsequent systems level approach was proposed and demonstrated in [28, 29, 13, 19]. The primary advantage of the systems level approach is it applicability to large-scale networks. This arises from the system level parameterization, which allows for decomposition of the synthesis optimization problem into multiple smaller subproblems, localized around each subsystem of the large-scale network. The recent survey paper [1] summarizes the systems level technique as well as a robust systems level design strategy for systems which may not satisfy some stringent constraints that the usual systems level approach imposes. Here, we will focus on the extensions which have been made

Distributed filtering and sensor fusion: Localized distributed Kalman filtering (LDKF) was a Kalman-like filtering method proposed by [30] as a way to extend the traditional Kalman filtering method to systems of large scale. Many distributed filtering methods have been proposed in the past, including the work of [21], the Kalman consensus filter [22], and the diffusion-based filtering approach of [4]. A distributed Bayesian filtering approach using maximum (or log) likelihood consensus was proposed in [2] as well. However, such approaches typically require each subsystem to store the entire plant network instead of just the local subset relevant to it, which leads to a major computational bottleneck. Instead, LDKF leverages the localized implementation of SLS, and is thus able to synthesis filters for large-scale networks which may not even be tractable for many of the aformentioned approaches. Furthermore, it was shown that LDKF is able to design such filters with performance comparable to that of the centralized Kalman filter (i.e., the accuracy of the estimate is the same between the two approaches when tested on networks where both methods are computationally tractable).

Adaptive control and dynamic network structures: Traditional adaptive control approaches [18, 25] and subsequent adaptive control algorithms which combine elements from data-driven control and machine learning [9, 10, 11] have been studied in a wide variety of settings. However, scalability to larger systems is not one of those settings. To this end, a systems-level approach of designing robust controllers which simultaneously adapts to uncertainties in parameters is proposed in [17]. Here, we develop a systems-level approach to handle uncertainties in the topology of the network, and propose a control law design scheme that is robust to changes in the network topology while locally and adaptively learning the locations at which the true topology differs from the assumed nominal. In the specific case of the power grid, we are motivated by natural failures such as power lines downed by severe weather conditions, and the subsequent need to upgrade the control scheme in an efficient way that does not involve redesigning from scratch. The interests of this study and the applications of the proposed scheme are not limited to just the grid. Collaborative tasks using distributed robotic systems that communicate with each other wirelessly (such as formation flying or information gathering) clearly necessitates a robust method of control to handle the highly dynamic communication network as the agents move about independently. In this direction, previous consensus-based approaches have been considered in cooperative robot control [6, 7] and oscillator synchronization [24], to name a few examples. Consensus-based methods of parameter estimation and sensor fusion have also been proposed [31, 23].

There exists a rich literature of work on the treatment of distributed networked control systems with dynamic structures; see [16], Table 2 for a comprehensive survey. [15] considers consensus for fault-detection in sensor networks whose topological structure switches according to a Markov chain. Event-triggered and sample-based consensus approaches for collections of systems arranged in a dynamic network have been studied in [14, 5, 20] and criterion for convergence are also provided. Such approaches have a lot of practical applicability, e.g., multiagent systems, which are not physically interconnected, perform their tasks independently of each other unless they meet and aggregate their observations. However, a primary difference between this branch of literature and the work presented here is that the true topology and/or uncertain parameters do not necessarily need to be fully determined prior to synthesizing a controller for it due to the ability of the systems level approach to designing robust localized controllers [19]. Hence, the performance of the consensus algorithm is less important than the performance of the stabilizing controller.

2 Notation

We use N_s to denote the number of systems in network, N_x (a multiple of N_s) to denote the total number of states, and $N_u \leq N_x$ as the number of control inputs. Denote the network of systems by an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with adjacency matrix $G \in \mathbb{R}^{N_s \times N_s}$ with vertex set $\mathcal{V} = \{1, 2, \cdots N_s\}$. Note that an undirected graph implies that G is a symmetric matrix. For two systems $i, j \in \mathcal{V}$, we have that j belongs in the set $\mathcal{N}(i)$, called the *neighbor set* of i, iff $(i, j) \in \mathcal{E}$. We say that the graph $\mathcal{G}$ is *connected* if there exists a path (i.e. a sequence of edges in $\mathcal{E}$) between every pair of nodes $i, j \in \mathcal{V}$.

3 A Brief Review of the Systems Level Approach

3.1 The Centralized Formulation

We will consider specific plants of the following linear, discrete-time form:

$$\mathbf{x}[t+1] = A\mathbf{x}[t] + B_1\mathbf{w}[t] + B_2\mathbf{u}[t] \tag{1a}$$
$$\bar{\mathbf{z}}[t] = C_1\mathbf{x}[t] + D_{11}\mathbf{w}[t] + D_{12}\mathbf{u}[t] \tag{1b}$$
$$\mathbf{y}[t] = C_2\mathbf{x}[t] + D_{21}\mathbf{w}[t] + D_{22}\mathbf{u}[t] \tag{1c}$$

where $\mathbf{x} \in \mathbb{R}^{N_x}$ denotes the state, $\mathbf{u} \in \mathbb{R}^{N_u}$ the control action, $\mathbf{w} \in \mathbb{R}^{N_w}$ the external disturbance, $\mathbf{y} \in \mathbb{R}^{N_y}$ the measurement, and $\bar{\mathbf{z}}$ the regulated output.

We desire to find a controller K which stabilizes the open-loop system when arranged in the feedback interconnection shown in Figure 1 such that the closed-loop system behavior minimizes some cost or error functional.

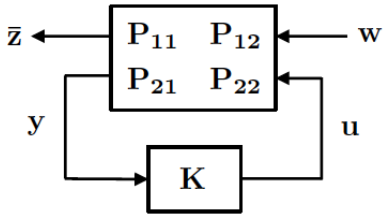

Figure 1: basic plant model

We can obtain a direct relationship between the disturbance $\mathbf{w}$ and the regulated output $\bar{\mathbf{z}}$. Apply the z-transform to the dynamics (1):

$$\mathbf{x}[t+1] = A\mathbf{x}[t] + B_1 w[t] + B_2 u[t] \Longrightarrow z\mathbf{x} = A\mathbf{x} + B_1 w + B_2 u \Longrightarrow \mathbf{x} = (zI - A)^{-1}(B_1 w + B_2 u)$$

$$\begin{cases} \bar{\mathbf{z}}[t] = C_1\mathbf{x}[t] + D_{11}\mathbf{w}[t] + D_{12}\mathbf{u}[t] \\ y[t] = C_2\mathbf{x}[t] + D_{21}\mathbf{w}[t] + D_{22}\mathbf{u}[t] \end{cases} \Longrightarrow \begin{cases} \bar{\mathbf{z}} = (C_1(zI - A)^{-1}B_1 + D_{11})\mathbf{w} + (C_1(zI - A)^{-1}B_2 + D_{12})\mathbf{u} \\ \mathbf{y} = (C_2(zI - A)^{-1}B_1 + D_{21})w + (C_2(zI - A)^{-1}B_2 + D_{22})\mathbf{u} \end{cases}$$

Let $P_{ij} = C_i(zI - A)^{-1}B_j + D_{ij}$ so that we can write:

$$\begin{bmatrix}\overline{\mathbf{z}}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ \mathbf{u}\end{bmatrix} = \begin{bmatrix}P_{11} & P_{12}\\ P_{21} & P_{22}\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ K\mathbf{y}\end{bmatrix}$$
$$\implies (I - P_{22}K)\mathbf{y} = P_{21}\mathbf{w}$$
$$\implies \overline{\mathbf{z}} = P_{11}\mathbf{w} + P_{12}K\mathbf{y} = (P_{11} + P_{12}K(I - P_{22})^{-1}P_{21})\mathbf{w}$$

The goal is to determine a K that will stabilize plant P and the cost/error we seek to minimize is a suitably chosen norm of the closed-loop transfer function from $\mathbf{w}$ to $\overline{\mathbf{z}}$. The **centralized optimal control formulation** is then as follows:

$$\min_K \left\| P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \right\| \tag{2}$$
$$\text{s.t. } K \text{ internally stabilizes } P$$

Two key assumptions are made about the system so that the above optimization problem is solvable:

1. The interconnection in Figure 1 is well-posed, meaning that all closed-loop transfer functions from each input to each output are proper. This further implies that the matrix $I - D_{22}D_k$ must be invertible.
2. Both the plant and controller realizations are stabilizable and detectable. This means that:
   - (A, B_2) and (A_k, B_k) are stabilizable: even though we cannot drive every state to their desired values, the components that are uncontrollable can at least be guaranteed to be stable and not problematic – a slightly weaker notion of controllability.
   - (A, C_2) and (A_k, C_k) are detectable: even though we cannot observe the value of every state, the ones that are unobservable can at least be guaranteed to be stable, hence not problematic – a slightly weaker notion of observability

3.2 System Level Parameterization

There are two main issues with the centralized formulation. First, the optimization problem (2) is nonconvex in terms of K, which makes it difficult to solve. Second, it may be intractable to solve in the case of large-scale systems (i.e., N_x is very large). This motivates the *system-level parameterization*, which essentially computes the entire closed-loop response instead of just the controller. In order to achieve this, K is allowed to have internal dynamics as well, instead of being simply constant:

$$\xi[t+1] = A_k\xi[t] + B_k\mathbf{y}[t] \tag{3a}$$
$$\mathbf{u}[t] = C_k\xi[t] + D_k\mathbf{y}[t] \tag{3b}$$

where ξ represents the internal state of the controller. Since the control law is given by $\mathbf{u} = K\mathbf{y}$, we can express $K = C_k(zI - A_k)^{-1}B_k + D_k$ by again taking the z-transform and determining the transfer function between $\mathbf{u}$ and $\mathbf{y}$.

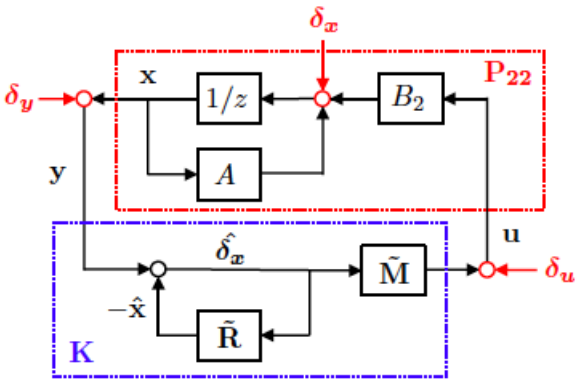

Fig. 3. The proposed state feedback controller structure, with $\tilde{\mathbf{R}} = I - z\mathbf{R}$ and $\tilde{\mathbf{M}} = z\mathbf{M}$.

Figure 2: Basic plant model for state feedback controller synthesis ([29]). Placeholder figure.

We will first look at the state feedback case, for the system (1) with $C_2 = I$ and $D_{21} = D_{22} = 0$:

$$\mathbf{x}[t+1] = A\mathbf{x}[t] + \mathbf{w}[t] + B\mathbf{u}[t] \tag{4a}$$
$$\mathbf{z}[t] = C_1\mathbf{x}[t] + D_{11}\mathbf{w}[t] + D_{12}\mathbf{u}[t] \tag{4b}$$
$$\mathbf{y}[t] = \mathbf{x}[t] \tag{4c}$$

With feedback control law written as $\mathbf{u} = K\mathbf{x}$ and expressing the z-transform, we get:

$$(zI - A)\mathbf{x} = BK\mathbf{x} + \mathbf{w} \Longrightarrow \mathbf{x} = (zI - A - BK)^{-1}\mathbf{w}$$
$$u = K\mathbf{x} = K(zI - A - BK)^{-1}\mathbf{w}$$

We then define the system response to be the transfer function matrices $\{\Phi_x, \Phi_u\}$ such that $\mathbf{x} = \Phi_x\mathbf{w}, \mathbf{u} = \Phi_u\mathbf{w}$, where $\Phi_x = (zI - A - BK)^{-1}$ and $\Phi_u = K(zI - A - BK)^{-1}$. This allows us to express the controller as $K = \Phi_u\Phi_x^{-1}$. However, computing the inverse Φ_x^{-1} will be difficult for large-scale systems, so instead of directly evaluating K, we arrange the blocks $\{\Phi_x, \Phi_u\}$ in the feedback interconnection shown in Figure 2. This arrangement further gives us the nice property that any sparsity structure imposed on the matrices $\{\Phi_x, \Phi_u\}$ will also be reflected in the structure of the controller K, a main advantage that will allow us to move from centralized controllers to distributed ones for systems of large scale.

We have the following algebraic characterization of the set of state-feedback system responses $\{\Phi_x, \Phi_u\}$ that are achievable by an internally stabilizing controller K.

Theorem 1 (State-Feedback System Level Parameterization)**.** The following results hold:

a. The affine subspace defined by

$$\begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \Phi_x \\ \Phi_u \end{bmatrix} = I \tag{5a}$$
$$\Phi_x, \Phi_u \in \frac{1}{z}\mathcal{RH}_\infty \tag{5b}$$

parametrizes all system responses from state disturbance $\mathbf{w}$ to (x, u) that give rise to achievable and internally stabilizing controllers.

b. Such controllers K are determined by $K = \Phi_u \Phi_x^{-1}$ and the system responses $\Phi_x = (zI - A - BK)^{-1}, \Phi_u = K(zI - A - BK)^{-1}$ are achieved.

Hence, any $\{\Phi_x, \Phi_u\}$ that satisfies (5) will yield an internally stabilizing controller $K = \Phi_u \Phi_x^{-1}$ for (4) that also gives us the desired system response. The types of desired system response include disturbance rejection (i.e, maintaining consensus among a network of subsystems despite a sudden entrance of $\mathbf{w}$ into one of the subsystems) or trajectory following/reference tracking.

To implement the closed-loop system, recall that signals in discrete-time can be represented by a linear combination of impulse signals. We solve a cost optimization problem similar to (2) and minimize over the *spectral component matrices* $\{\Phi_x[k], \Phi_u[k]\}$ corresponding to the system responses:

$$\Phi_x(z) = \sum_{k=1}^{T} \Phi_x[k] \frac{1}{z^i}, \ \Phi_u(z) = \sum_{k=1}^{T} \Phi_u[k] \frac{1}{z^i}$$

From the block diagram Figure 2, we can derive the following formulas for our signals of interest. First, the control law:

$$\mathbf{u}(z) = \tilde{\Phi}_u(z)\hat{\mathbf{w}}(z) = z\Phi_u(z)\hat{\mathbf{w}}(z) \Longrightarrow \mathbf{u}[t] = (\tilde{\Phi}_u * \mathbf{w})[t] = \sum_{k=0}^{T} \Phi_u[k]\hat{\mathbf{w}}[t-i]$$

because converting from z-domain to the time domain turns multiplication into convolution.

Next, the state vector can be written:

$$\begin{aligned}
&\hat{\mathbf{w}}(z) = \mathbf{x}(z) + \tilde{\Phi}_{\mathbf{x}}\hat{\mathbf{w}}(z) = \mathbf{x}(z) + (I - \Phi_x(z))\hat{\mathbf{w}}(z) \\
\Longrightarrow &\mathbf{x}(z) = \Phi_x(z)\hat{\mathbf{w}}(z) \Longrightarrow \mathbf{x}[t] = (\Phi_x * \mathbf{w})[t] = \sum_{k=0}^{T} \Phi_x[k]\hat{\mathbf{w}}[t-i] \\
\Longrightarrow &\mathbf{x}[t+1] = (\Phi_x * \mathbf{w})[t+1] = \sum_{k=0}^{T} \Phi_x[k]\hat{\mathbf{w}}[t+1-i]
\end{aligned}$$

Overall, the state-feedback controller is implemented as follows:

$$\hat{\mathbf{x}}[t] = \sum_{k=2}^{T} \Phi_x[k]\hat{\mathbf{w}}[t+1-k] \tag{6a}$$

$$\hat{\mathbf{w}}[t] = \mathbf{x}[t] - \hat{\mathbf{x}}[t] \tag{6b}$$

$$\mathbf{u}[t] = \sum_{k=1}^{T} \Phi_u[k]\hat{\mathbf{w}}[t+1-k] \tag{6c}$$

It was shown in [19] that even when this relationship is approximately satisfied, the implementation as in (6) produces a stable closed-loop response.

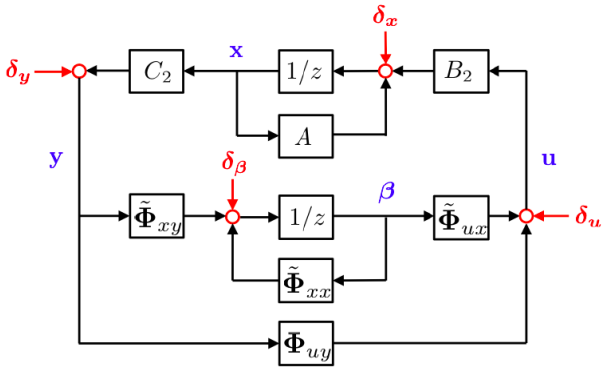

Figure 3: Output feedback controller structure with $\tilde{\Phi}_{xx} := z(I - z\Phi_{xx})$, $\tilde{\Phi}_{ux} := z\Phi_{ux}$, and $\tilde{\Phi}_{xy} := -z\Phi_{xy}$ (adapted from [29]). Placeholder figure.

Now we will consider the output-feedback case. We will restrict our attention to when $D_{22} = 0$ in (1), so that the output dynamic is not affected by the control input u. The overall plant dynamics become:

$$\begin{aligned} \mathbf{x}[t+1] &= A\mathbf{x}[t] + Bu[t] + \mathbf{w}[t] \\ y[t] &= C\mathbf{x}[t] + \mathbf{v}[t] \end{aligned}$$

Substituting the output-feedback control law $\mathbf{u} = K\mathbf{y}$ into the z-transform of the dynamics yields $(zI - A - BKC)\mathbf{x} = \mathbf{w} + BK\mathbf{v}$.

Similarly to the state-feedback case, we shall define the system response to be the matrices Φ_{xx}, Φ_{xy}, Φ_{ux}, Φ_{uy} such that:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \Phi_{xx} & \Phi_{xy} \\ \Phi_{ux} & \Phi_{uy} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix}$$

They are achieved by the controller K as follows:

$$\begin{aligned} \Phi_{xx} &= (zI - A - BKC)^{-1} \\ \Phi_{xy} &= KC\Phi_{xx} \\ \Phi_{ux} &= \Phi_{xx}BK \\ \Phi_{uy} &= K + KC\Phi_{xx}BK \end{aligned}$$

and the full closed-loop architecture is given by Figure 3.

Like the state-feedback case, we have a main result for the algebraic characterization of all achievable system responses $\{\Phi_{xx}, \Phi_{xy}, \Phi_{ux}, \Phi_{uy}\}$.

Theorem 2 (Output-Feedback System Level Parameterization)**.** The following results hold:

a. The affine subspace defined by

$$\begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \Phi_{xx} & \Phi_{xy} \\ \Phi_{ux} & \Phi_{uy} \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix}$$

$$\begin{bmatrix} \Phi_{xx} & \Phi_{xy} \\ \Phi_{ux} & \Phi_{uy} \end{bmatrix} \begin{bmatrix} zI - A \\ -C \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}$$

$$\Phi_{xx}, \Phi_{ux}, \Phi_{xy} \in \frac{1}{z}\mathcal{RH}_\infty, \Phi_{uy} \in \mathcal{RH}_\infty$$

parametrizes all system responses from state disturbance that give rise to achievable and internally stabilizing controllers.

b. Such controllers K are determined by $K = \Phi_{uy} - \Phi_{ux}\Phi_{xx}^{-1}\Phi_{xy}$ and the system responses $\Phi_{xx} = (zI - A - BKC)^{-1}, \Phi_{ux} = KC\Phi_{xx}, \Phi_{xy} = \Phi_{xx}BK, \Phi_{uy} = K + KC\Phi_{xx}BK$ are achieved.

3.3 Spatial/Temporal Locality and Delays

Let the network be denoted by $\mathcal{G}(\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ the set of vertices (subsystems) and $\mathcal{E}$ the set of unweighted, directed edges. The distance metric $d(v_i \to v_j)$ on $\mathcal{G}$ is the shortest path from v_i to v_j where $v_i, v_j \in \mathcal{V}$. The direction of the arrow must be noted because the edges are directed.

Locality is measured in two ways: spatially and temporally.

1. **Spatial locality** ensures that the disturbance is only felt within a neighborhood of the entrance. To model this as a constraint, we introduce a few definitions.

   Definition 1. Denote $\Phi_{x,ij}$ to be the transfer matrix describing the perturbation from disturbance component w_j (which directly affects system j) to the state estimate x_i of system i. Then the map Φ_x is *d-localizable* iff for every pair of systems $i, j \in \mathcal{V}$, $\Phi_{x,ij} = 0$ for all i such that $\text{dist}(i, j) > d$. Analogously, the definition for d-localizability of the map Φ_u to the control law u_i follows the same way. Accordingly, we will associate a local *d-hop set* with each system i as the set of systems j where the (i, j)th entry of G^d is nonzero.

   Informally, a system is d-localizable if the ith system in the network can construct a controller using a plant model which is limited to the neighborhood of systems within distance d of i and implemented using information signals that are gathered only from this same neighborhood.

   We denote the subspace $\mathcal{L}_d$ as the d-localized constraint set. It constrains Φ_x to be d-localized and Φ_u to be $(d+1)$-localized. This forces any disturbance to be rejected in finite space(i.e., so that it does not propagate throughout the entire network). This is implemented by imposing sparsity patterns on each spectral factor of Φ_x and Φ_u:

$$\text{supp}\left(\Phi_x[t]\right) = \bigcup_{s=1}^{t} \frac{1}{z^s}\text{supp}\left(A^{\min(d, \lfloor t_c(s-1) \rfloor)}\right)$$
$$\text{supp}\left(\Phi_u[t]\right) = \bigcup_{s=1}^{t} \frac{1}{z^s}\text{supp}(B^T)\text{supp}\left(A^{\min(d+1, \lfloor t_c(s-1) \rfloor)}\right)$$

   where t_c is the communication speed of the controller network relative to the plant network, and $\text{supp}(\cdot)$ denotes the support operator. Clearly, if $t_c < 1$, it would be impossible to localize the system because the controller is not reacting quickly enough to prevent the disturbance effects from propagating throughout the plant.

2. **Temporal locality** ensures that the system response Φ_x, Φ_u has a finite impulse response of horizon length T. This forces the disturbance to be rejected in finite time. We enforce the

constraint as $\{\Phi_x, \Phi_u\} \in \mathcal{F}_T$, where

$$\mathcal{F}_T := \left\{ G \in \mathcal{RH}_\infty \middle| G = \sum_{i=0}^{T} \frac{1}{z^i} G[i] \right\}$$

The z-transform affine constraint can then be expressed in the time-domain as follows:

$$\begin{bmatrix} I & -A & & & & \\ & I & -A & & & \\ & & & \ddots & \ddots & \\ & & & & & -A \\ & & & & & I \end{bmatrix} \begin{bmatrix} \Phi_x[T+1] \\ \Phi_x[T] \\ \vdots \\ \Phi_x[2] \\ \Phi_x[1] \end{bmatrix} = \begin{bmatrix} B\Phi_u[T] \\ B\Phi_u[T-1] \\ \vdots \\ B\Phi_u[1] \\ I \end{bmatrix}$$

where the fact that the matrix sizes and vector lengths are limited by T is a result of the finite time-horizon constraint.

Thus, in each of the subproblems described in the previous subsection, we can express the constraint as membership in the set $\mathcal{S} = \mathcal{L}_d \cap \mathcal{F}_T$. Systems that satisfy both spatial and time constraints simultaneously with d-hops and horizon T are said to be (d, T)-*localizable*.

3.4 System Level Synthesis

Now, instead of solving (2), the desired closed-loop behavior can be achieved by enforcing appropriate convex constraints on $\{\Phi_x, \Phi_u\}$ and solving the optimal control problem of the form:

$$\min_{\{\Phi_x, \Phi_u\}} f(\Phi_x, \Phi_u, Q, R) \tag{8a}$$

$$\text{s.t. } \{\Phi_x, \Phi_u\} \in \mathcal{S} \tag{8b}$$

where $Q \in \mathbb{R}^{N_x \times N_x}, R \in \mathbb{R}^{N_u \times N_u}$ are cost matrices which assign weight to Φ_x, Φ_u respectively, f is a convex function of both Φ_x and Φ_u (typically a weighted norm or LQR cost), and $\mathcal{S}$ is a convex set which encodes the constraints. Of particular interest, especially in distributed, decentralized implementations, is when $\mathcal{S}$ includes the spatial and temporal constraints described in 3.3.

4 Localized, Distributed Kalman Filters

When the standard LQR problem is formulated in terms of (8), it is known as the **localized linear quadratic regulator** (LLQR), and is expressed as follows:

$$\min_{\{\Phi_x[k],\Phi_u[k]\}_{k=1}^T} \sum_{k=1}^{T} \operatorname{tr}\left(\Phi_x[k]^T Q \Phi_x[k] + \Phi_u[k]^T R \Phi_u[k]\right)$$

$$\text{s.t.} \begin{bmatrix} I & -A & & & \\ & I & -A & & \\ & & \ddots & \ddots & \\ & & & & -A \\ & & & & I \end{bmatrix} \begin{bmatrix} \Phi_x[T+1] \\ \Phi_x[T] \\ \vdots \\ \Phi_x[2] \\ \Phi_x[1] \end{bmatrix} = \begin{bmatrix} B\Phi_u[T] \\ B\Phi_u[T-1] \\ \vdots \\ B\Phi_u[1] \\ I \end{bmatrix}$$

$$\operatorname{supp}\left(\Phi_x[k]\right) = \bigcup_{j=1}^{k} \frac{1}{z^j} \operatorname{supp}\left(A^{\min(d,\lfloor t_c(j-1)\rfloor)}\right)$$

$$\operatorname{supp}\left(\Phi_u[k]\right) = \bigcup_{j=2}^{k} \frac{1}{z^j} \operatorname{supp}(B^T) \operatorname{supp}\left(A^{\min(d+1,\lfloor t_c(j-1)\rfloor)}\right)$$

(d, T)-localizability allows us to decompose the constraints into smaller subconstraints. Furthermore, the objective function may also be decomposed, as the following algebraic manipulation shows:

$$\begin{aligned}
\sum_{k=1}^{T} \operatorname{tr}\left(\Phi_x[k]^T Q \Phi_x[k] + \Phi_u[k]^T R \Phi_u[k]\right) &= \sum_{i=1}^{N_x} \sum_{k=1}^{T} \operatorname{tr}\left(\Phi_x[k]^T Q \Phi_x[k] e_i e_i^T\right) + \operatorname{tr}\left(\Phi_u[k]^T R \Phi_u[k] e_i e_i^T\right) \\
&= \sum_{i=1}^{N_x} \sum_{k=1}^{T} \operatorname{tr}\left(e_i^T \Phi_x[k]^T Q \Phi_x[k] e_i\right) + \operatorname{tr}\left(e_i^T \Phi_u[k]^T R \Phi_u[k] e_i\right) \\
&= \sum_{i=1}^{N_x} \sum_{k=1}^{T} e_i^T \Phi_x[k]^T Q \Phi_x[k] e_i + e_i^T \Phi_u[k]^T R \Phi_u[k] e_i \\
&= \sum_{i=1}^{N_x} \sum_{k=1}^{T} (\Phi_x[k]^T)_i Q (\Phi_x[k])_i + (\Phi_u[k]^T)_i R (\Phi_u[k])_i
\end{aligned}$$

Here, we will formulate the LLQR for Kalman filtering. The system of interest is of the form

$$\mathbf{x}[k+1] = A\mathbf{x}[k] + \mathbf{w}[k] \tag{9a}$$

$$\mathbf{y}[k] = C\mathbf{x}[k] + \mathbf{v}[k] \tag{9b}$$

where $\mathbf{w}[k] \in \mathbb{R}^{N_x}, \mathbf{v}[k] \in \mathbb{R}^{N_y}$ are now Gaussian vectors with mean 0 and respective covariance matrices $Q = q * I_{N_x}$ and $R = r * I_{N_y}$ with $q, r > 0$.

Recall the Kalman filter state estimate update equation:

$$\hat{\mathbf{x}}[k+1|k] = A[k](\hat{\mathbf{x}}[k|k-1] + L[k](\mathbf{y}[k] - C[k]\hat{\mathbf{x}}[k|k-1]))$$

where $\mathbf{x}[k+1|k]$ denotes the state estimate at time $k+1$ given information obtained until time k.

For SLS to apply, we can derive an equivalent version of the affine constraints in Theorem 1. Taking the z-transform of the Kalman filter equation yields:

$$\begin{aligned}\hat{\mathbf{x}}[k+1|k] = A(\hat{\mathbf{x}}[k|k-1] + L[k]\tilde{\mathbf{y}}[k]) &\implies (zI - A)\hat{\mathbf{x}} = L(y - C\hat{\mathbf{x}}) \\ &\implies (zI - A + LC)\hat{\mathbf{x}} = L\mathbf{y} = L(C\mathbf{x} + \mathbf{w})\end{aligned}$$

Now take the z-transform of the system dynamics and add a term of $LC\mathbf{x}$ to both sides.

$$(zI - A + LC)\mathbf{x} = LC\mathbf{x} + \mathbf{w}$$

Subtracting the two z-transformed equations yields:

$$(zI - A + LC)\tilde{\mathbf{x}} = \mathbf{w} - L\mathbf{v}$$

where $\tilde{\mathbf{x}} := \mathbf{x} - \hat{\mathbf{x}}$ is the estimation error. Rewritten:

$$\tilde{\mathbf{x}} = \underbrace{(zI - A + LC)^{-1}}_{=:\Phi_w}\mathbf{w} + \underbrace{(-(zI - A + LC)^{-1}L)}_{=:\Phi_v}\mathbf{v}$$

and we can now treat Φ_w, Φ_v like our system response maps Φ_x, Φ_u. More specifically, $\Phi_w \in \mathbb{R}^{N_x \times N_x}$ represents the closed-loop transfer matrix from $\mathbf{w}$ to $\tilde{\mathbf{x}}$ and $\Phi_v \in \mathbb{R}^{N_y \times N_x}$ represents the closed-loop map from $\mathbf{v}$ to $\tilde{\mathbf{x}}$. By Theorem 2, the achievable set of observers can be parameterized by the affine subspace

$$\begin{bmatrix}\Phi_w & \Phi_v\end{bmatrix}\begin{bmatrix}zI - A \\ -C\end{bmatrix} = I$$
$$\Phi_w, \Phi_v \in \frac{1}{z}\mathcal{RH}_\infty$$

and we can implement the gain matrix $L = -\Phi_w^{-1}\Phi_v$.

To relate better to the characterization in Theorem 1, we can take the transpose across:

$$\begin{bmatrix}zI - A^T & -C^T\end{bmatrix}\begin{bmatrix}\Phi_w \\ \Phi_v\end{bmatrix} = I$$

After solving for internally stabilizing Φ_v and Φ_w, we can recover an estimate $\hat{\mathbf{x}}$ using the relationship $\hat{\mathbf{x}} = \mathbf{x} - \tilde{\mathbf{x}} = \mathbf{x} - \Phi_w\mathbf{w} - \Phi_v\mathbf{v}$. Alternatively, because one would not know what $\mathbf{x}, \mathbf{w}, \mathbf{v}$ are in real implementation, we write instead $\hat{\mathbf{x}} = \Phi_w B\mathbf{u} + \Phi_v\mathbf{y} = \Phi_v\mathbf{y}$, which is obtained directly from the z-transform of the Kalman filter equation and using $L = -\Phi_w^{-1}\Phi_v$. Note that because there is no control term in (9), we set $\mathbf{u} = 0$.

The overall LLQR problem that we must solve in order to obtain $\Phi_w[k], \Phi_v[k]$ for $k = 1, 2, \cdots, T$

such that any disturbances can be localized within d hops and time horizon T is:

$$\min_{\{\Phi_w[k],\Phi_v[k]\}_{k=1}^T} \sum_{k=1}^{T} \mathrm{tr}\left(\Phi_w[k] Q \Phi_w[k]^T + \Phi_v[k] R \Phi_v[k]^T\right) \tag{10a}$$

$$\text{s.t.} \begin{bmatrix} I & -A^T & & & \\ & I & -A^T & & \\ & & \ddots & \ddots & \\ & & & & -A^T \\ & & & & I \end{bmatrix} \begin{bmatrix} \Phi_w^T[T+1] \\ \Phi_w^T[T] \\ \vdots \\ \Phi_w^T[2] \\ \Phi_w^T[1] \end{bmatrix} = \begin{bmatrix} C^T \Phi_v^T[T] \\ C^T \Phi_v^T[T-1] \\ \vdots \\ C^T \Phi_v^T[1] \\ I \end{bmatrix} \tag{10b}$$

$$\mathrm{supp}\left(\Phi_w^T[k]\right) = \bigcup_{j=1}^{k} \frac{1}{z^j} \mathrm{supp}\left(A^{\min(d, \lfloor t_c(j-1) \rfloor)}\right) \tag{10c}$$

$$\mathrm{supp}\left(\Phi_v[k]\right) = \bigcup_{j=2}^{k} \frac{1}{z^j} \mathrm{supp}(C) \mathrm{supp}\left(A^{\min(d+1, \lfloor t_c(j-1) \rfloor)}\right) \tag{10d}$$

Consider a specific network of 4 subsystems, illustrated in Figure 4. It is clear that it has the following adjacency matrix:

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

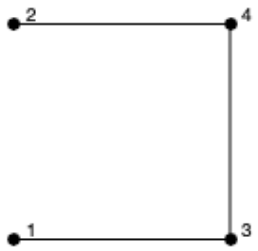

Figure 4: Example network for illustration of distributed Kalman filtering.

The ith subsystem obeys the following dynamics

$$\mathbf{x}_i[k+1] = A_{ii}\mathbf{x}_i[k] + \sum_{j \in \mathcal{N}_i} A_{ij}\mathbf{x}_j[k] + \mathbf{w}_i[k]$$
$$\mathbf{y}_i[k] = C_i \mathbf{x}_i[k] + \mathbf{v}_i[k]$$

where

$$A_{ii} = \begin{bmatrix} 1 & 0.2 \\ a_{ii}^{(1)} & a_{ii}^{(2)} \end{bmatrix}, \quad A_{ij} = \begin{bmatrix} 0 & 0 \\ a_{ij} & 0 \end{bmatrix} \quad C_i = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

This means that $N_x = 8$ and $N_y = 4$.

We will define

$$E_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbb{1}_k = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^k$$

The full A matrix can be written as follows

$$A = \left[\begin{array}{cc|cc|cc|cc} 1 & 0.2 & & & 0 & 0 & & \\ a_{11}^{(1)} & a_{11}^{(2)} & & & a_{13} & 0 & & \\ \hline & & 1 & 0.2 & & & 0 & 0 \\ & & a_{22}^{(1)} & a_{22}^{(2)} & & & a_{24} & 0 \\ \hline 0 & 0 & & & 1 & 0.2 & 0 & 0 \\ a_{31} & 0 & & & a_{33}^{(1)} & a_{33}^{(2)} & a_{34} & 0 \\ \hline & & 0 & 0 & 0 & 0 & 1 & 0.2 \\ & & a_{42} & 0 & a_{43} & 0 & a_{44}^{(1)} & a_{44}^{(2)} \end{array}\right]$$

where the blank spaces are all 2×2 matrices of zeros. Note that the support of A is given by the matrix

$$\text{supp}(A) = \text{supp}(G) \otimes E_2 = \left[\begin{array}{cc|cc|cc|cc} 1 & 1 & & & 1 & 1 & & \\ 1 & 1 & & & 1 & 1 & & \\ \hline & & 1 & 1 & & & 1 & 1 \\ & & 1 & 1 & & & 1 & 1 \\ \hline 1 & 1 & & & 1 & 1 & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ \hline & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

and for the $k \geq 0$ power of A, $\text{supp}(A^k) = \text{supp}(G^k) \otimes E_2$ where $\otimes$ denotes the Kronecker product.

Similarly, stack the C_i into a full C matrix

$$C = \left[\begin{array}{cc|cc|cc|cc} 1 & 0 & & & & & & \\ \hline & & 1 & 0 & & & & \\ \hline & & & & 1 & 0 & & \\ \hline & & & & & & 1 & 0 \end{array}\right]$$

Take the communication speed (of the controller network relative to the plant network) to be $t_c = 1$, let $d = 1$, and let $T = 3$. Our goal is to solve the LLQR problem with respect to these (d, T)-locality constraints to obtain $\{\Phi_w[i], \Phi_v[i]\}$ for all $i = 1, \cdots, T$ and form an estimate $\hat{\mathbf{x}}$ of the full state $\mathbf{x} := \begin{bmatrix} \mathbf{x}_1^T & \cdots & \mathbf{x}_{N_x}^T \end{bmatrix}^T$ of the system.

Step 1 is to take care of the spatial locality constraints (10c) and (10d) by designating the supports of $\Phi_w[i]$ and $\Phi_v[i]$ for each time $i = 1, \cdots, T$.

As in the affine achievability constraint (10b), $\Phi_w[i] = I_8$ is fixed, where I_8 is the 8×8 identity matrix. So we will begin with $\Phi_v[1]$, then alternate between Φ_w and Φ_v until we reach time T.(SJ: short version: replace all matrix of 1s with E_2.)

$$\operatorname{supp}(\Phi_v[1]) = \operatorname{supp}(C)\operatorname{supp}(A^0) = \operatorname{supp}(C) \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \end{bmatrix}$$

$$\operatorname{supp}(\Phi_w[2]) = \operatorname{supp}\left(A^{\min(1,\lfloor 1(2-1)\rfloor)}\right) = \operatorname{supp}(A)$$

$$\operatorname{supp}(\Phi_v[2]) = \operatorname{supp}(C)\operatorname{supp}(A) = \operatorname{supp}(C) \begin{bmatrix} 1 & 1 & & & 1 & 1 & & \\ 1 & 1 & & & 1 & 1 & & \\ & & 1 & 1 & & & 1 & 1 \\ & & 1 & 1 & & & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & & & 1 & 1 & & \\ & & 1 & 1 & & & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\operatorname{supp}(\Phi_w[3]) = \operatorname{supp}\left(A^{\min(1,\lfloor 1(3-1)\rfloor)}\right) = \operatorname{supp}(\Phi_w[2])$$

$$\operatorname{supp}(\Phi_v[3]) = \operatorname{supp}\left(A^{\min(1,\lfloor 1(3-1)\rfloor)}\right) = \operatorname{supp}(C)\operatorname{supp}(A^2)$$

$$\operatorname{supp}(A^2) = \begin{bmatrix} 1 & 1 & & & 1 & 1 & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \implies \operatorname{supp}(\Phi_v[3]) = \begin{bmatrix} 1 & 1 & & & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Note that for $k > T$, $\operatorname{supp}(\Phi_w[k]) = \operatorname{supp}(\Phi_w[T])$ and $\operatorname{supp}(\Phi_v[k]) = \operatorname{supp}(\Phi_v[T])$ because $\min(d+1, t_c(k-1)) = d+1$.

We want to partition the optimization problem (10) into further subproblems. So Step 2 will be to determine the d-localized regions of each subsystem. For that, we must look at the final support matrices $\operatorname{supp}(A^{d+1}) = \operatorname{supp}(A^2)$ and $\operatorname{supp}(C^2)$.

1. Subsystem 1: We want to ensure that inputting a disturbance into Subsystem 1 will be localized by the controller we design through the optimization problem (10). The d-outgoing region of subsystem 1 can be determined by looking at the first row of $\text{supp}(A^2), \text{supp}(C^2)$ and determining the locations of the nonzero entries. This will help us determine which of the four subsystems are affected by the disturbance and which of them we need to get information from in order to kill it.

   With that, we can see that the active subsystems are $\begin{bmatrix}1 & 3 & 4\end{bmatrix}$ and the indices corresponding to A are therefore $\begin{bmatrix}1 & 2 & 5 & 6 & 7 & 8\end{bmatrix}$. The submatrix of A and C formed by using these indices are:

$$A_{l,1} = \left[\begin{array}{cc|cc|cc} 1 & 0.2 & 0 & 0 & & \\ a_{11}^{(1)} & a_{11}^{(2)} & a_{13} & 0 & & \\ \hline 0 & 0 & 1 & 0.2 & 0 & 0 \\ a_{31} & 0 & a_{33}^{(1)} & a_{33}^{(2)} & a_{34} & 0 \\ \hline & & 0 & 0 & 1 & 0.2 \\ & & a_{43} & 0 & a_{44}^{(1)} & a_{44}^{(2)} \end{array}\right], \quad C = \left[\begin{array}{cc|cc|cc} 1 & 0 & & & & \\ \hline & & 1 & 0 & & \\ \hline & & & & 1 & 0 \end{array}\right]$$

   where the subscript of l indicates that it is local and 1 denotes the subsystem around which the local subproblem is built. We will say $A_{l,1}$ has dimension $N_x^{(1)}$-by-$N_x^{(1)}$, where $N_x^{(1)} = 6$, and $C_{l,1}$ has dimension $N_y^{(1)}$-by-$N_x^{(1)}$, where $N_y^{(1)} = 3$.

2. Subsystem 2: Similarly, we want to ensure that inputting a disturbance into Subsystem 2 will be localized. The d-outgoing region of subsystem 1 can be determined by looking at the third row of $\text{supp}(A^2)$ (to account for the fact that each subsystem has two dimensions), second row of $\text{supp}(C^2)$. The active subsystems are $\begin{bmatrix}2 & 3 & 4\end{bmatrix}$ and the indices corresponding to A are therefore $\begin{bmatrix}3 & 4 & 5 & 6 & 7 & 8\end{bmatrix}$.

   The submatrix of A and C formed by using these indices are:

$$A_{l,1} = \left[\begin{array}{cc|cc|cc} 1 & 0.2 & & & 0 & 0 \\ a_{22}^{(1)} & a_{22}^{(2)} & & & a_{24} & 0 \\ \hline & & 1 & 0.2 & 0 & 0 \\ & & a_{33}^{(1)} & a_{33}^{(2)} & a_{34} & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0.2 \\ a_{42} & 0 & a_{43} & 0 & a_{44}^{(1)} & a_{44}^{(2)} \end{array}\right], \quad C = \left[\begin{array}{cc|cc|cc} 1 & 0 & & & & \\ \hline & & 1 & 0 & & \\ \hline & & & & 1 & 0 \end{array}\right]$$

   We will say $A_{l,1}$ has dimension $N_x^{(2)}$-by-$N_x^{(2)}$, where $N_x^{(2)} = 6$, and $C_{l,1}$ has dimension $N_y^{(2)}$-by-$N_x^{(2)}$, where $N_y^{(2)} = 3$.

3. For Subsystems 3 and 4, the d-hop regions become the entire network because all subsystems are within $(d+1)$ hops away. Thus $N_x^{(3)} = N_x^{(4)} = N_x = 8$ and $N_y^{(3)} = N_y^{(4)} = N_y = 4$. For the remainder of the algorithm illustration, we will restrict our focus to subsystems 1 and 2, but the same steps follows for subsystems 3 and 4.

Step 3 will be to embed the time-horizon and spatial locality constraints into all our matrices and split the LLQR into four simple quadratic programs with linear constraints, where four comes from the number of subsystems.

We first stack together the support matrices derived from Step 1 to create the following matrix

$$S := \begin{bmatrix} \text{supp}(\Phi_v[1]) \\ \hline\hline \text{supp}(\Phi_w[2]) \\ \hline\hline \text{supp}(\Phi_v[2]) \\ \hline\hline \text{supp}(\Phi_w[3]) \\ \hline\hline \text{supp}(\Phi_v[3]) \end{bmatrix} = \left[\begin{array}{cc|cc|cc|cc} 1 & 1 & & & & & & \\ \hline & & 1 & 1 & & & & \\ \hline & & & & 1 & 1 & & \\ \hline & & & & & & 1 & 1 \\ \hline\hline 1 & 1 & & & 1 & 1 & & \\ 1 & 1 & & & 1 & 1 & & \\ \hline & & 1 & 1 & & & 1 & 1 \\ & & 1 & 1 & & & 1 & 1 \\ \hline 1 & 1 & & & 1 & 1 & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ \hline & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\hline 1 & 1 & & & 1 & 1 & & \\ \hline & & 1 & 1 & & & 1 & 1 \\ \hline 1 & 1 & & & 1 & 1 & 1 & 1 \\ \hline & & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\hline 1 & 1 & & & 1 & 1 & & \\ 1 & 1 & & & 1 & 1 & & \\ \hline & & 1 & 1 & & & 1 & 1 \\ & & 1 & 1 & & & 1 & 1 \\ \hline 1 & 1 & & & 1 & 1 & 1 & 1 \\ 1 & 1 & & & 1 & 1 & 1 & 1 \\ \hline & & 1 & 1 & 1 & 1 & 1 & 1 \\ & & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline\hline 1 & 1 & & & 1 & 1 & 1 & 1 \\ \hline & & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

where $\text{supp}(\Phi_w[1])$ is not part of the stack because we set $\Phi_w[1] = I_8$. The dimensions of S are $(TN_y + (T-1)N_x)$-by-N_x.

Therefore, in the original optimization problem (10), the total number of variables to optimize over is equal to the number of ones in the matrix S above. We will denote the variables as $\Phi_w[k](i,j)^{(l)} \in \mathbb{R}$ or $\Phi_v[k](i,j)^{(l)} \in \mathbb{R}$, where $k = 1, \cdots, T$ denotes the time as usual, (i,j) is such that i denotes the row position of the entry relative to the matrix it belongs in, whether it be $\Phi_w[k]$ or $\Phi_v[k]$, j denotes the subsystem number, and $l = 1$ or 2 denotes the whether it corresponds to the first or second state of subsystem j (recall each subsystem state is two dimensions). To make the notation more clear, the first two stacked matrices of the actual variable matrix

$$\begin{bmatrix} \Phi_v[1] \\ \hline\hline \Phi_w[2] \\ \hline\hline \Phi_v[2] \\ \hline\hline \Phi_w[3] \\ \hline\hline \Phi_v[3] \end{bmatrix}$$

are written as follows:

$$\left[\begin{array}{cc|cc|cc|cc}
\Phi_v[1](1,1)^{(1)} & \Phi_v[1](1,1)^{(2)} & & & & & & \\ \hline
 & & \Phi_v[1](2,2)^{(1)} & \Phi_v[1](2,2)^{(2)} & & & & \\ \hline
 & & & & \Phi_v[1](3,3)^{(1)} & \Phi_v[1](3,3)^{(2)} & & \\ \hline
 & & & & & & \Phi_v[1](4,4)^{(1)} & \Phi_v[1](4,4)^{(2)} \\ \hline\hline
\Phi_w[2](1,1)^{(1)} & \Phi_w[2](1,1)^{(2)} & & & \Phi_w[2](1,3)^{(1)} & \Phi_w[2](1,3)^{(2)} & & \\
\Phi_w[2](2,1)^{(1)} & \Phi_w[2](2,1)^{(2)} & & & \Phi_w[2](2,3)^{(1)} & \Phi_w[2](2,3)^{(2)} & & \\ \hline
 & & \Phi_w[2](3,2)^{(1)} & \Phi_w[2](3,2)^{(2)} & & & \Phi_w[2](3,4)^{(1)} & \Phi_w[2](3,4)^{(2)} \\
 & & \Phi_w[2](4,2)^{(1)} & \Phi_w[2](4,2)^{(2)} & & & \Phi_w[2](4,4)^{(1)} & \Phi_w[2](4,4)^{(2)} \\ \hline
\Phi_w[2](5,1)^{(1)} & \Phi_w[2](5,1)^{(2)} & & & \Phi_w[2](5,3)^{(1)} & \Phi_w[2](5,3)^{(2)} & \Phi_w[2](5,4)^{(1)} & \Phi_w[2](5,4)^{(2)} \\
\Phi_w[2](6,1)^{(1)} & \Phi_w[2](6,1)^{(2)} & & & \Phi_w[2](6,3)^{(1)} & \Phi_w[2](6,3)^{(2)} & \Phi_w[2](6,4)^{(1)} & \Phi_w[2](6,4)^{(2)} \\ \hline
 & & \Phi_w[2](7,2)^{(1)} & \Phi_w[2](7,2)^{(2)} & \Phi_w[2](7,3)^{(1)} & \Phi_w[2](7,3)^{(2)} & \Phi_w[2](7,4)^{(1)} & \Phi_w[2](7,4)^{(2)} \\
 & & \Phi_w[2](8,2)^{(1)} & \Phi_w[2](8,2)^{(2)} & \Phi_w[2](8,3)^{(1)} & \Phi_w[2](8,3)^{(2)} & \Phi_w[2](8,4)^{(1)} & \Phi_w[2](8,4)^{(2)} \\ \hline\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right]$$

Step 2 gives us a way to optimize over all these variables by splitting the matrix S columnwise. We will use the terminology Subproblem j to denote the subproblem corresponding to subsystem j, for $j = 1, \cdots, 4$. Then Subproblem 1 optimizes over the variables in columns 1 and 2 of S, Subproblem 2 optimizes over columns 3 and 4 of S, Subproblem 3 over columns 5 and 6, and Subproblem 4 over columns 7 and 8.

First, we will rewrite (10b) to better match the format of the stacked matrix S as follows:

$$\underbrace{\begin{bmatrix} C^T & -I_8 & 0 & 0 & 0 \\ 0 & A^T & C^T & -I_8 & 0 \\ 0 & 0 & 0 & A^T & C^T \end{bmatrix}}_{=:M} \begin{bmatrix} \Phi_v^T[1] \\ \Phi_w^T[2] \\ \Phi_v^T[2] \\ \Phi_w^T[3] \\ \Phi_v^T[3] \end{bmatrix} = \begin{bmatrix} A^T \\ 0 \\ 0 \end{bmatrix} \tag{11}$$

where the dimensions of M are TN_x-by-$(TN_y + (T-1)N_x)$. The second and third rows are simply an algebraic rearrangement of the achievability characterization. The first row equation has an

additional algebraic manipulation due to the fact that $\Phi_w[1] = I_8$:

$$\Phi_w^T[2] = A^T\Phi_w^T[1] + C^T\Phi_v^T[1] = A^T + C^T\Phi_v^T[1] \Longrightarrow C^T\Phi_v^T[1] - \Phi_w^T[2] = A^T$$

1. Subproblem 1: Take a subset of the constraint (11) corresponding to the variables that it is responsible for. Namely, the first two columns of variables $\left\{\Phi_w[k](i,1)^{(l)}, \Phi_v[k](i,1)^{(l)}\right\}$. We will denote the matrix that stacked these variables by Φ_1.

   To reduce M to the submatrix we desire, we use the first column of S and take the columns of M for every place where there is a 1.

$$\left[\begin{array}{cccc|cccccccc|cccccccccc}
1&0&0&0&-1&&&&&&&&&&&&&&&&&&\\
0&0&0&0&&-1&&&&&&&&&&&&&&&&&\\
0&1&0&0&&&-1&&&&&&&&&&&&&&&&\\
0&0&0&0&&&&-1&&&&&&&&&&&&&&&\\
0&0&1&0&&&&&-1&&&&&&&&&&&&&&\\
0&0&0&0&&&&&&-1&&&&&&&&&&&&&\\
0&0&0&1&&&&&&&-1&&&&&&&&&&&&\\
0&0&0&0&&&&&&&&-1&&&&&&&&&&&\\
\hline
&&&&1&a_{11}^{(1)}&&&0&a_{31}&&&1&0&0&0&-1&&&&&&&&\\
&&&&0.2&a_{11}^{(2)}&&&0&0&&&0&0&0&0&&-1&&&&&&&\\
&&&&&&1&a_{22}^{(1)}&&&0&a_{42}&0&1&0&0&&&-1&&&&&&\\
&&&&&&0.2&a_{22}^{(2)}&&&0&0&0&0&0&0&&&&-1&&&&&\\
&&&&0&a_{13}&&&1&a_{33}^{(1)}&0&a_{43}&0&0&1&0&&&&&-1&&&&\\
&&&&0&0&&&0.2&a_{33}^{(2)}&0&0&0&0&0&0&&&&&&-1&&&\\
&&&&&&0&a_{24}&0&a_{34}&1&a_{44}^{(1)}&0&0&0&1&&&&&&&-1&&\\
&&&&&&0&0&0&0&0.2&a_{44}^{(2)}&0&0&0&0&&&&&&&&-1\\
\hline
&&&&&&&&&&&&&&&&1&a_{11}^{(1)}&&&0&a_{31}&&\\
&&&&&&&&&&&&&&&&0.2&a_{11}^{(2)}&&&0&0&&\\
&&&&&&&&&&&&&&&&&&1&a_{22}^{(1)}&&&0&a_{42}\\
&&&&&&&&&&&&&&&&&&0.2&a_{22}^{(2)}&&&0&0\\
&&&&&&&&&&&&&&&&0&a_{13}&&&1&a_{33}^{(1)}&0&a_{43}\\
&&&&&&&&&&&&&&&&0&0&&&0.2&a_{33}^{(2)}&0&0\\
&&&&&&&&&&&&&&&&&&0&a_{24}&0&a_{34}&1&a_{44}^{(1)}\\
&&&&&&&&&&&&&&&&&&0&0&0&0&0.2&a_{44}^{(2)}\\
\hline
&\vdots&&&&&&&&&&&&\vdots&&&&&&&&&&\ddots\\
\hline\hline
1&&&&1&1&&&1&1&&&1&&1&&1&1&&&1&1&&\cdots\\
\uparrow&&&&\uparrow&\uparrow&&&\uparrow&\uparrow&&&\uparrow&&\uparrow&&\uparrow&\uparrow&&&\uparrow&\uparrow&&\cdots
\end{array}\right]$$

   The resulting submatrix, obtained by collecting all these chosen columns together, will be a stacked matrix of the submatrices of $A_{l,1}$ and $C_{l,1}$. We will denote this matrix by M_1, and it has dimension $TN_x^{(1)}$-by-n_1, where $n_1 \leq N_x^{(1)}$ is the number of Φ_w, Φ_v variables that subsystem 1 is responsible for.

   We do the same with the right side of (11). Replace A with $A_{l,1}$ and take the first two columns of the matrix:

$$\begin{bmatrix} 1 & a_{11}^{(1)} \\ 0.2 & a_{11}^{(2)} \\ 0 & a_{13} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \end{bmatrix}$$

We will denote this reduced matrix by b_1. It has dimension $TN_x^{(1)}$-by-2.

Finally, we must do the same with the objective function. First, we will arrange copies of the cost matrices R and Q so that they match the pattern of the stacked Φ:

$$\begin{bmatrix} \Phi_v[1] \\ \hline \Phi_w[2] \\ \hline \Phi_v[2] \\ \hline \Phi_w[3] \\ \hline \Phi_v[3] \end{bmatrix} \qquad J := \begin{bmatrix} R & & & & \\ \hline & Q & & & \\ \hline & & R & & \\ \hline & & & Q & \\ \hline & & & & R \end{bmatrix}$$

where recall $R = r * I_{N_y} \in \mathbb{R}^{N_y}$ and $Q = q * I_{N_x} \in \mathbb{R}^{N_x}$ are the covariance matrices of the noise that enters into the system. Repeat the same reduction process: take the first column of S and take the rows and columns of J wherever there is a 1:

$$\left[\begin{array}{cc||cccccc|cccccc|cccccc}
\to & 1 & r & & & & & & & & & & & & & & & & & \\
 & & & r & & & & & & & & & & & & & & & & \\
 & & & & r & & & & & & & & & & & & & & & \\
 & & & & & r & & & & & & & & & & & & & & \\
\hline
\to & 1 & & & & & q & & & & & & & & & & & & & \\
\to & 1 & & & & & & q & & & & & & & & & & & & \\
 & & & & & & & & q & & & & & & & & & & & \\
 & & & & & & & & & q & & & & & & & & & & \\
\to & 1 & & & & & & & & & q & & & & & & & & & \\
\to & 1 & & & & & & & & & & q & & & & & & & & \\
 & & & & & & & & & & & & q & & & & & & & \\
 & & & & & & & & & & & & & q & & & & & & \\
\hline
\to & 1 & & & & & & & & & & & & & r & & & & & \\
 & & & & & & & & & & & & & & & r & & & & \\
 & & & & & & & & & & & & & & & & r & & & \\
\to & 1 & & & & & & & & & & & & & & & & r & & \\
\hline
\to & 1 & & & & & & & & & & & & & & & & & q & \\
\to & 1 & & & & & & & & & & & & & & & & & & q \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
\to & 1 & & & & & & & & & & & & & & & & & & \\
\to & 1 & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
 & & & & & & & & & & & & & & & & & & & \\
\hline
 & & & & & & & & \vdots & & & & & & & & & & \ddots & \\
\hline\hline
 & 1 & & & 1 & 1 & & & 1 & 1 & & 1 & 1 & & & 1 & 1 & \cdots & & \\
 & \uparrow & & & \uparrow & \uparrow & & & \uparrow & \uparrow & & \uparrow & \uparrow & & & \uparrow & \uparrow & \cdots & &
\end{array}\right]$$

We will denote this reduced cost J matrix as J_1. It has dimension n_1-by-n_1.

Now Subproblem 1 is now a quadratic-cost, linear-constrained optimization problem:

$$\begin{aligned} &\min_{\Phi_1} \operatorname{tr}(\Phi_1^T J_1 \Phi_1) \\ \text{s.t. } &M_1 \Phi_1 = b_1 \end{aligned}$$

which can be analytically solved using Lagrange multipliers

$$\begin{aligned} \mathcal{L}(\Phi_1, \lambda) &= \operatorname{tr}(\Phi_1^T J_1 \Phi_1) + \lambda^T (M_1 \Phi_1 - b_1) \\ \implies \left.\frac{\partial \mathcal{L}}{\partial \Phi_1}\right|_{\substack{\Phi_1=\Phi_1^* \\ \lambda=\lambda^*}} &= (J_1 + J_1^T)\Phi_1^* + \lambda^* M_1 := 0 \\ \left.\frac{\partial \mathcal{L}}{\partial \lambda}\right|_{\substack{\Phi_1=\Phi_1^* \\ \lambda=\lambda^*}} &= M\Phi_1^* - b_1 := 0 \end{aligned}$$

yielding a linear system of equations that can be solved directly by taking inverse (or pseudoinverse):

$$\underbrace{\begin{bmatrix} 2J_1 & M_1^T \\ M_1 & 0 \end{bmatrix}}_{=:M_1'} \begin{bmatrix} \Phi_1^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 0 \\ b_1 \end{bmatrix} \implies \begin{bmatrix} \Phi_1^* \\ \lambda^* \end{bmatrix} = M_1'^{-1} \begin{bmatrix} 0 \\ b_1 \end{bmatrix}$$

Note that M_1' is square because J_1 is square.

An alternative way to solve the optimization problem is to formulate it as follows:

$$\begin{aligned} &\min_{\Phi_1} \left\| \sqrt{J_1} \Phi_1 \right\|_2 \\ \text{s.t. } &M_1 \Phi_1 = b_1 \end{aligned}$$

and performing a change of variables $\Psi_1 := \sqrt{J_1}\Phi_1$:

$$\begin{aligned} &\min_{\Psi_1} \|\Psi_1\|_2 \\ \text{s.t. } &M_1 J_1^{-1} \Psi_1 = b_1 \end{aligned}$$

which is simply solving a linear system of equations, and has solution $\Psi_1 = (M_1 J_1^{-1})^+ b_1$, and so $\Phi_1 = J_1^{-1}(M_1 J_1^{-1})^+ b_1$. In implementation, either method yields the same solution up to a tiny fluctuation in values that are extremely close to zero (of magnitude $10^{-16}, 10^{-17}$).

2. Subproblem 2: The construction follows in exactly the same way, with the only difference being that submatrices $A_{l,2}$ and $C_{l,2}$ are used and that the variables Φ_2 that it is responsible for are determined by the third and fourth columns of S.

Once we have solved for all $\{\Phi_w[i], \Phi_v[i]\}, i = 1, \cdots, T$, we can use the relationship $\hat{\mathbf{x}}[k] = \Phi_v[k]\mathbf{y}[k]$ to obtain our state estimate.

Simulation results with $N_s = N_y = 625, N_x = 625 * 2 = 1250$, and $d = 1, t_c = 2$, and $T = 5$ are presented in Figure 5 and Figure 6.

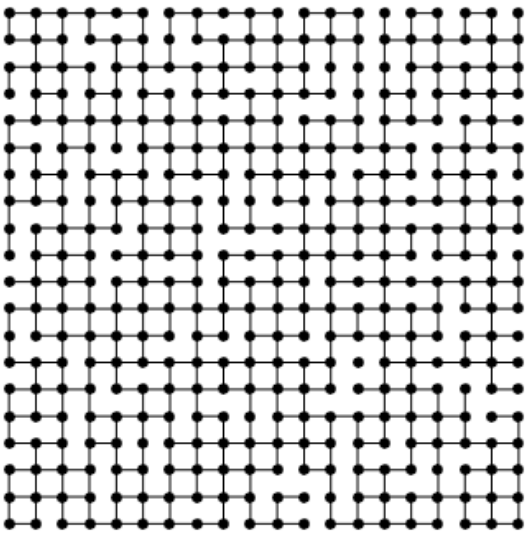

Figure 5: 20×20 mesh network of subsystems.

Compared against the centralized Kalman filter (which is implemented by stacking the matrices and estimating a $N_x \times 1$ state vector), the error of estimates for the first eight states (i.e, error of state estimate for the first four subsystems) are plotted below. There is a huge spike in the first few seconds in the distributed SLS implementation, but in steady-state, the performance against the Kalman filter is nearly identical. The real benefit of distributed SLS filtering over a centralized approach is the computation time; indeed, it takes the Kalman filter 19.3891 seconds to compute the estimate state trajectory, whereas the distributed SLS filter finishes in 3.2585 seconds. Thus, we are trading off a slight degradation in performance (though nearly similar) for a large boost in computation time.

5 Localized Robust and Adaptive Controllers

5.1 Centralized Implementation

We will restrict our attention to the case where only the topological structure of the plant network varies according to link modifications while the controller network remains static. That is, the adjacency matrix G (from which the dynamics matrix A is built) is varied while the B matrix remains constant. The parameter values of each subsystem are assumed to be known.

We begin with the knowledge of a nominal topological structure A^* of the network. Moreover, we are aware that at least one link has been disconnected, and although we do not know which one(s), we are given a finite collection of K candidate link failure matrices D, one of which gives us the true topology $A = A^* + D$. The assumption is reasonable because in real-world scenarios we are oftentimes able identify the local region in which a potential link failure has occurred (i.e., power grid). We assume that none of the candidate matrices causes the graph to become disconnected.

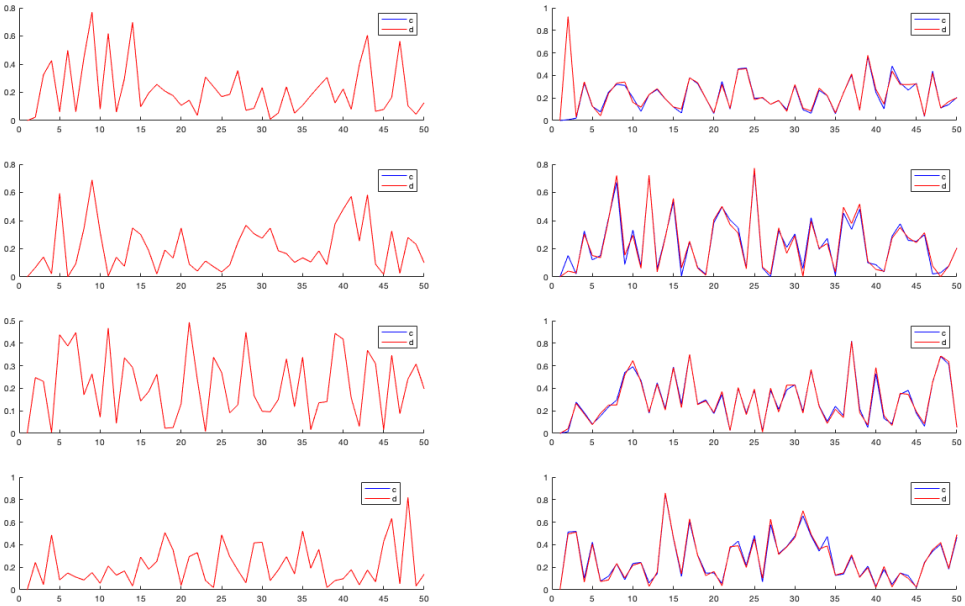

Figure 6: Error estimate for the first eight state components of the system.

With this premise, the system dynamics can be expressed as:

$$\mathbf{x}[t+1] = (A^* + D)\mathbf{x}[t] + B\mathbf{u}[t] + \mathbf{w}[t]$$

where A^* denotes the known nominal system and link failures D enter in the form of perturbations to A^*.

To characterize the set of D, we introduce basis matrices $\mathcal{A}_l$ to encode all possible single-link disconnections so that linear combinations of them can be used to model a general number of failures corresponding to each candidate failure matrix.

$$D \in \left\{ \sum_{l=1}^{M} \xi_l \mathcal{A}_l \ : \ \xi_l \in \{-1, 0, 1\} \right\} =: \mathcal{P}_0$$

where coefficient $\xi_l = 1$ is for when a link is added, $\xi_l = -1$ is when a link is deleted, $\xi_l = 0$ is when a link is unchanged. Overall, we have a discrete combinatorial set of points in topology adaptation. Furthermore, to make the problem tractable, we impose $K << 2^M$, and take "link failures" to mean disconnections of existing links (i.e., $\xi_l \in \{0, -1\}$ only). The algorithm can easily be extended to handle new link additions by simply allowing ξ_l to take value 1 and creating more basis matrices for D corresponding to where the links have been added.

We refer to $\mathcal{P}_0$ as the initial *consistent set*. At each timestep, the consistent set is updated using new observations of $(\mathbf{x}[t+1], \mathbf{x}[t], \mathbf{u}[t])$: we iterate over all the matrices in the previous consistent set and discard those that do not satisfy

$$\left\| \mathbf{x}[t+1] - \left(A^* + \sum_{l=1}^{M} \xi_l \mathcal{A}_l \right) \mathbf{x}[t] - B\mathbf{u}[t] \right\|_\infty \leq \eta$$

where we assume bounded disturbance $\|\mathbf{w}\|_\infty \leq \eta$. Hence, $\mathcal{P}_{t+1} = \mathcal{P}_t \cap \mathcal{C}_{t+1}$. Clearly, we have $|\mathcal{P}_{t+1}| \leq |\mathcal{P}_t|$.

To design the controller, start with the implementation (6). Because we are updating our estimates of the values at each time step as we adapt, the system responses $\Phi_x^{(t)}, \Phi_u^{(t)}$ are denoted with superscript (t) to make explicit that they are time-varying. Define

$$\begin{aligned}\Delta_k(A, B, \Phi_x, \Phi_u) &= \Phi_x[k+1] - A\Phi_x[k] - B\Phi_u[k] \quad \forall k < T \\ \Delta_T(A, B, \Phi_x, \Phi_u) &= -A\Phi_x[T] - B\Phi_u[T]\end{aligned}$$

Then from (6b):

$$\begin{aligned}\hat{\mathbf{w}}[t] &= \mathbf{x}[t] - \sum_{k=2}^{T} \Phi_x^{(t)}[k]\hat{\mathbf{w}}[t+1-k] \\ &= (A\mathbf{x}[t-1] + Bu[t-1] + \mathbf{w}[t-1]) - \sum_{k=2}^{T} \Phi_x^{(t)}[k]\hat{\mathbf{w}}[t+1-k] \\ &= A\left(\sum_{k=1}^{T} \Phi_x^{(t-1)}[k]\hat{\mathbf{w}}[t-k]\right) + B\left(\sum_{k=1}^{T} \Phi_u^{(t-1)}[k]\hat{\mathbf{w}}[t-k]\right) + \mathbf{w}[t-1] - \sum_{k=2}^{T} \Phi_x^{(t)}[k]\hat{\mathbf{w}}[t+1-k] \\ &= -\sum_{k=1}^{T} \Delta_k(A, B, \Phi_x^{(t-1)}, \Phi_u^{(t-1)})\hat{\mathbf{w}}[t-k] + \sum_{k=1}^{T-1} \left(\Phi_x^{(t-1)} - \Phi_x^{(t)}\right)[k+1]\hat{\mathbf{w}}[t-k] + \mathbf{w}[t-1]\end{aligned}$$

where the second equality comes from substituting in the expression for $\mathbf{u}[t]$ and using the fact that $\Phi_x^{(t)}[1] = I$ for all t. By the triangle inequality and the submultiplicativity property of norms:

$$\|\hat{\mathbf{w}}[t]\| \leq \sum_{k=1}^{T} \left\|\Delta_k(A, B, \Phi_x^{(t-1)}, \Phi_u^{(t-1)})\right\| \|\hat{\mathbf{w}}[t-k]\| + \left\|\sum_{k=1}^{T} \left(\Phi_x^{(t-1)} - \Phi_x^{(t)}\right)[k+1]\hat{\mathbf{w}}[t-k]\right\| + \eta \tag{12}$$

As done in [17], one way to satisfy (12) is by constraining the individual terms of the sum

$$\sum_{k=1}^{T} \left\|\Delta_k(A', B, \Phi_x^{(t-1)}, \Phi_u^{(t-1)})\right\| \leq \lambda_t \tag{13a}$$

$$\left\|\sum_{k=1}^{T} \left(\Phi_x^{(t-1)} - \Phi_x^{(t)}\right)[k+1]\hat{\mathbf{w}}[t-k]\right\| \leq \gamma \tag{13b}$$

for every possible matrix $A' = A^* + D$, where $D \in \mathcal{P}_t$. λ_t is referred to as the *robustness margin* and for each timestep t it determines whether the controller is stabilizable with the tth polytope of uncertainties. We initially solve (8) with $f(\Phi_x, \Phi_u, Q, R, \lambda_t) = \lambda_t$, and if $\lambda_t \leq \lambda^*$ for a chosen value λ^*, we resolve (8) with

$$f(\Phi_x, \Phi_u, Q, R, \lambda_t) = \sum_{k=1}^{T} \|Q\Phi_x[k] + R\Phi_u[k]\|_1$$

Two steps are taken because optimizing for a performance objective is only reasonable if robust stability is feasible with uncertainty $\mathcal{P}_t$.

γ is the *adaptation margin* and ensures that the system response Φ_x doesn't fluctuate wildly with varying $\mathbf{w}$. One can observe that if $\mathbf{w}$ is very large, $\Phi_x^{(t)}$ is forced to be close to the previous iteration's $\Phi_x^{(t-1)}$ in order to compensate and obey the constraint. However, if γ is chosen too small, it may learn the incorrect system response and subsequently remain close to this response for all future time, resulting in an unstable controller. For this reason, the addition the constraint (13b) is typically made optional. In the case of topological uncertainties, the sparsity patterns of all the candidate topologies D may be different. So it may not be reasonable to constrain the system responses $\Phi_x^{(t)}$ to evolve in a way that keeps close to $\Phi_x^{(t-1)}$ of the previous iteration. We will only append (13a) to $\mathcal{S}$ in the optimization problem (8).

This yields the full optimization problem for centralized robust control which adapts to topological changes:

$$\begin{aligned}
&\min_{\left\{\Phi_x^{(t)}[k],\Phi_u^{(t)}[k]\right\}_{k=1}^{T},\lambda_t} \begin{cases} \lambda_t & \text{if } \lambda_t \le \lambda^* \\ \sum_{k=1}^{T} \left\| Q\Phi_x^{(t)}[k] + R\Phi_u^{(t)}[k] \right\|_1 & \text{else} \end{cases} \\
&\text{s.t. } \{\Phi_x^{(t)}, \Phi_u^{(t)}\} \in \mathcal{L}_d \cap \mathcal{F}_T \\
&\quad \sum_{k=1}^{T} \left\| \Delta_k(A', B, \Phi_x^{(t-1)}, \Phi_u^{(t-1)}) \right\| \le \lambda_t
\end{aligned}$$

In theory, we will need to impose additional assumptions on the initial consistent set $\mathcal{P}_0$ in order to justify the elimination of (13b). However, we show through simulation that the algorithm is still able to maintain good performance even with the constraint removed.

We illustrate performance using power grid dynamics. Following the model used in [27], power networks are composed of several synchronized oscillators coupled together to operate under the following nonlinear second-order ODE swing dynamics

$$c_i\ddot{\theta}_i + b_i\dot{\theta}_i = -\sum_{j\in\mathcal{N}_i} a_{ij}(\theta_i - \theta_j) + w_i + u_i$$

where for the ith system (typically called a bus) of the network, c_i is its inertia, b_i is a damping factor, w_i is the external disturbance, and u_i is the control action. The state of each system is described by the relative phase angle θ_i between the bus's rotor axis and resultant magnetic field, as well as its derivative, the frequency $\dot{\theta}_i$. Use $\mathbf{x}^{(i)} = \begin{bmatrix}\theta_i & \dot{\theta}_i\end{bmatrix}^T$ and rewrite the dynamics in matrix form as:

$$\begin{bmatrix}\dot{\theta}_i \\ \ddot{\theta}_i\end{bmatrix} = \begin{bmatrix}0 & 1 \\ -\frac{a_i}{c_i} & -\frac{b_i}{c_i}\end{bmatrix}\begin{bmatrix}\theta_i \\ \dot{\theta}_i\end{bmatrix} + \sum_{j\in\mathcal{N}(i)} \begin{bmatrix}0 & 0 \\ \frac{a_{ij}}{c_i} & 0\end{bmatrix}\begin{bmatrix}\theta_j \\ \dot{\theta}_j\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} w_i + \begin{bmatrix}0 \\ 1\end{bmatrix} u_i, \quad a_i := \sum_{j\in\mathcal{N}(i)} a_{ij} \tag{14}$$

Discretize the system with sampling time Δt using the mean-value theorem

$$\dot{\mathbf{x}} \approx \frac{\mathbf{x}[t+\Delta t] - \mathbf{x}[t]}{\Delta t} = A\mathbf{x}[t] + Bu[t] + Dw[t]$$

to obtain the following approximated model:

$$\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}[t+1] = \begin{bmatrix} 1 & \Delta t \\ -\frac{a_i}{c_i}\Delta t & 1-\frac{b_i}{c_i}\Delta t \end{bmatrix}\begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}[t] + \sum_{j\in\mathcal{N}(i)} \begin{bmatrix} 0 & 0 \\ \frac{a_{ij}}{c_i}\Delta t & 0 \end{bmatrix}\begin{bmatrix} x_1^{(j)} \\ x_2^{(j)} \end{bmatrix}[t] + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} w_i[t] + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} u_i[t] \tag{15}$$

The initial nominal topological configuration is a square (4-subsystem) ring and the true topology is a rearranged chain network with the following adjacency matrix:

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

In Figure 7, we simulate this system for 20 time steps and two different values of η, both cases driven by the same noise process $\mathbf{w}$ such that $\|\mathbf{w}[t]\|_\infty \leq 0.3$. Initial conditions are made small: for each system, $\mathbf{x}_1^{(i)}[0] \in \text{Unif}[0,2]$ and $\mathbf{x}_2^{(i)}[0] \in \text{Unif}[3,5]$. As expected, it takes longer for $\eta = 1$ to deduce to the true link failure D than it takes $\eta = 0.5$. Furthermore, the control law for $\eta = 0.5$ does not stabilize the states of the systems to zero as closely as the control law for $\eta = 1$ does, which is illustrative of the fact that the system adapts better with perturbations that yield nonzero state values even at the expense of driving the system unstable.

5.2 Localized Implementation

Now, we make a transition from the centralized algorithm to a localized version that employs a decomposition of the full problem into multiple independent pieces, to achieve computational scalability to systems of larger size. Define local set $\mathcal{L}(i)$ for each system $i = 1, \cdots, N_s$ to be the d-hop set of i. Further, let d_c be the communication delay matrix between systems of the network, with the (i,j)th entry defined as

$$d_c(i,j) = \begin{cases} |j-i| & \text{if } j \in \mathcal{L}(i) \\ \infty & \text{else} \end{cases}$$

Each system i begins with a local set of candidate matrices

$$\mathcal{P}_0^{(i)} = \left\{ \sum_{l=1}^{M_i} \xi_l \mathcal{A}_l^{(i)} \ : \ \xi_l \in \{-1, 0, 1\} \right\}$$

where the M_i basis matrices $\mathcal{A}_l^{(i)}$ have dimensions equal to that is the submatrix A_i, which only includes the rows of A corresponding to the systems in $\mathcal{L}(i)$. Hence, each system only keeps track of link modifications within its own local subset.

Each consistent set is locally updated according to the rule $\mathcal{P}_t^{(i)} = \mathcal{P}_{t-1}^{(i)} \cap \mathcal{C}_t^{(i)}$, where $\mathcal{C}_t^{(i)}$ is the consistent set constructed from local state observations similarly to the centralized algorithm: we

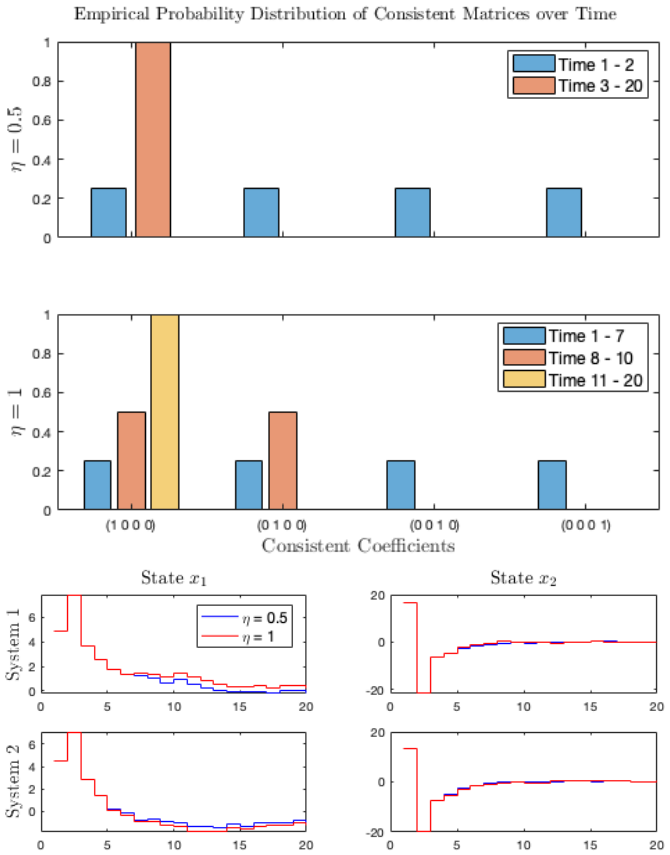

Figure 7: In the top figure, the evolution over time of the empirical PMF, which is simply set to be uniformly distributed over all candidate topologies consistent with the current accumulation of observations from the dynamics. On the bottom figure, the state trajectories for systems 1 and 2 are shown.

eliminate matrices from $\mathcal{P}_{t-1}^{(i)}$ which do not satisfy

$$\left\| \mathbf{x}_i[t+1] - \left(A_i^* + \sum_{l=1}^{M_i} \xi_l \mathcal{A}_l^{(i)} \right) \mathbf{x}[t] - B_i \mathbf{u}[t] \right\|_\infty \leq \eta$$

To design local controllers, we solve a local optimization problem of the form (8) for the ith columns of the system response matrices $\Phi_{x,i}^{(t)}, \Phi_{u,i}^{(t)}$. The constraints follow analogously to (13). Define the submatrix

$$\Delta_k^j \left(A, B, \Phi_{x,ji}^{(t)}, \Phi_{u,ji}^{(t)} \right) := \Phi_{x,ji}^{(t)}[k+1] - \sum_{l \in \mathcal{N}(j)} A_{jl} \Phi_{x,li}^{(t)}[k] - B_j \Phi_{u,ji}^{(t)}[k]$$

where $i, j = 1, \cdots, N_s$ regardless of whether they are neighbors or not, $k = 1, \cdots, T$, and t iterates over the simulation time. This allows us to define our robustness margin constraints

$$\left\| \sum_{j \in \mathcal{L}(i)} \Delta_k^j \left(A, B, \Phi_{x,ji}^{(t)}, \Phi_{u,ji}^{(t)} \right) \right\| \leq c_i \rho^{k-1} \; \forall k \leq T-1$$

$$c_i \sum_{k=1}^{T} \rho^{k-1} \leq \lambda_t^{(i)} + \epsilon$$

and in place of a constant, we introduce $\rho > 0$ and $c_i > 0$ to ensure faster exponential convergence to zero, which is motivated by the possibility of local disturbances propagating throughout the network in a cascading manner if it is not killed quickly enough within the local region. The slack variable ϵ helps us reduce the two-step process (checking for robust stabilizability prior to determining a legitimate control law) of solving (8) into a single step optimization problem. The objective function to minimize becomes

$$f(\Phi_x, \Phi_u, Q, R) = \sum_{k=1}^{T} \left\| Q\Phi_x[k] + R\Phi_u[k] \right\|_1 + r_c \epsilon$$

where $r_c > 0$ is a *relaxation constant* hyperparameter and we impose the additional constraint $\epsilon \geq 0$ within $\mathcal{S}$ in (8). The reasoning is that if the solution causes ϵ to be very large, this implies that the system is unable to determine a control law that is robustly stabilizable to all uncertainties in the current $\mathcal{P}_t$. We choose a very large relaxation constant so that even the slightest deviation of ϵ from 0 causes the cost objective to become large. When this happens, the optimization problem focuses on decreasing the uncertainty in the polytope. On the other hand, if the cost is small, it indicates that there exists a solution Φ_x, Φ_u that is robustly stabilizable with respect to $\mathcal{P}_t$.

Additionally, the method used in [17] ensured that these constraints were satisfied for all extreme points of $\mathcal{P}_t$, just as in the centralized version. However the number of extreme points scaled immensely , leading to much burden in the computation. To bypass this, we employ an alternative strategy using smart random selection of a point in the polytope. Note that the ith polytope of parameters can be described by a set of equations $M_i \boldsymbol{\xi}^{(i)} \leq q_i$ for parameter vector of dimension p_i. We sample a Gaussian random vector $\mathbf{c}_i \in \mathbb{R}^{p_i}$, normalized to be uniform, and solve the following minimization problem to obtain our selection $\boldsymbol{\xi}_i$ in the direction of the sampled random vector

$$\min_{\boldsymbol{\xi}^{(i)}} \mathbf{c}_i^T \boldsymbol{\xi}^{(i)}$$

$$\text{s.t. } M_i \boldsymbol{\xi}^{(i)} \leq q_i$$

Note that because we are minimizing over an affine subspace, this inevitably leads to one of the extreme points to be chosen. Once the selection is made, we construct a local A_i submatrix and solve the local optimal control problem (8) with constraints and objective function set as above.

5.3 Iterative Multi-Stage Implementation

Suppose the system dynamics is a time-varying hybrid version of (4):

$$\begin{aligned} \mathbf{x}[t+1] &= A(\gamma(t))\mathbf{x}[t] + B\mathbf{u}[t] + \mathbf{w}[t] \\ \mathbf{x}_i[t+1] &= A_{ii}(\gamma(t))\mathbf{x}_i[t] + \sum_{j \in \mathcal{N}_i(\gamma(t))} A_{ij}\mathbf{x}_j[t] + B_i\mathbf{u}[t] + \mathbf{w}_i[t] \ \forall i = 1, \cdots, N_s \end{aligned} \tag{16}$$

where $\gamma(t) \in \mathbb{N}$ is a discrete-valued signal which switches with time and B is kept constant across all possible states of A. According to [23, 31], consensus among distributed systems is achievable with time-varying topologies, under conditions such as joint-connectedness (the union of the graphs in the entire collection of topologies is connected) among topologies that are visited infinitely many times. As such, we will restrict our attention in this paper to hybrid dynamics where the signal γ switches according to an ergodic Markov chain with a finite number K of states. Furthermore, each of K topogolies are assumed to be connected and undirected, which trivially implies that they are balanced graphs.

Each system i maintains an estimate $\hat{\theta}_i$ of some central value θ, which updates its value over time according to the dynamics

$$\hat{\theta}_i[t+1] = W_{ii}[t]\hat{\theta}_i[t] + \sum_{i \in \mathcal{N}_i(\gamma(t))} W_{ij}[t]\hat{\theta}_j[t] \tag{17}$$

where the W_{ij} are normalized weights. For simplicity, we choose them to be uniform, although convergence to the true value can also be achieved with other values, as mentioned in [31].

Our objective is to robustly control a system with dynamics described in (16) subject to uncertainties in the Markov chain model under which the topology for A switches. We assume that the disturbance $\|\mathbf{w}[t]\|_\infty \leq \eta$ for a constant $\eta > 0$ and for all t. At the start, the true system topology $A(\gamma(0))$ (with adjacency matrix $G(\gamma(0))$) is known.

Suppose link modifications occur for a sequence of times $\{T_1, T_2, \cdots\}$. This sequence of times is unknown to each state and must be determined using local information, as we will describe in the process below. Each time the occurrence of event occurs, we are aware that at least one link has been either added or deleted, but we do not know which one(s). Thus, each system keeps a nominal topology estimate $A^{(i)}(\gamma^*(t))$ and updates it whenever it detects that a switch has been made. As mentioned in our brief review of consensus algorithms, B is fixed constant and known because we would know exactly where in the network we placed our actuators. Additionally, we assume that the transition probabilities of the Markov chain at configuration $\gamma(t)$ are also unknown. Each system hence maintains an estimate $\hat{P}^{(i)}$ of the transition probability matrix P, which it updates both locally and through averaging with its other neighbors according to (17).

An initial distributed control law $\mathbf{u}[t]$ and internal state value $\hat{\mathbf{w}}[t]$ are easily computed using the standard SLS method developed in [27]. We now would like to identify the set of modified links

when a switch occurs. The methodology that follows is the same across all subsystems $i = 1, \cdots, N_s$ and is only performed on the local subset $\mathcal{L}_i(t)$ and corresponding local A submatrix formed by extracting the local rows of the total A matrix. With that understanding, the relevant subscript i is removed for notational simplicity.

For each of the K topologies of the Markov chain, we are given $M_k, k = 1, \cdots, K$ basis matrices $\mathcal{A}_\ell^{(k)} \in \mathbb{R}^{N_x \times N_x}$ corresponding to a single link modification; they essentially serve as guesses for where we believe a link has changed. The collective modification may have been a linear combination of these bases. Suppose we have a fixed number $N_j^{(k)}$ of candidate coefficients at each link modification T_j. Then there exists a vector of coefficients $\boldsymbol{\xi} \in \mathcal{P}_0 := \{\boldsymbol{\xi} \in \mathbb{R}^M : \xi_\ell \in \{-1, 0, 1\}, \ell = 1, \cdots, M\}$ such that $A = A^* + \sum_{\ell=1}^M \xi_\ell \mathcal{A}_\ell$, where $A^* := A^{(i)}(\gamma^*(t))$ for shorthand notation, and $\xi_\ell = -1$ if link ℓ has been deleted from the nominal topology, 0 if there is no change, and 1 if new link ℓ has been added.

At each timestep t, an observation $\mathbf{x}[t]$ is made from the system (16). We identify which coefficients remain consistent with the system dynamics $(\mathbf{x}[t], \mathbf{x}[t-1], \mathbf{u}[t-1])$ by updating

$$\mathcal{P}_{t+1} := \left\{ \boldsymbol{\xi} \in \mathcal{P}_t \;\middle|\; \|\mathbf{x}[t+1] - (A^* + D)\mathbf{x}[t] + B\mathbf{u}[t]\| \leq \eta,\ D := \sum_{\ell=1}^M \xi_\ell \mathcal{A}_\ell \right\}$$

We refer to the set $\mathcal{P}_t$ to be the *consistent set* of coefficients at time t. This consistent set begins with cardinality $N_j^{(k)}$ and shrinks with increasing time until it becomes a singleton set containing the coefficient $\boldsymbol{\xi}$ corresponding to the true topology $A(\gamma(t))$ where $\gamma(t) = k$. Because identification for each system i was only done using information local to i, additional consensus may be performed to further narrow down the set of consistent coefficients in order to exactly estimate the state of the Markov chain. Rather than an average, we would take a maximum likelihood estimate because the state of the Markov chain is a discrete quantity.

As this identification and consensus process is being done, it is important to maintain system stability. We use the communication delay matrix d_c defined in 5.2 and time horizon T described. Because we will be considering adaptive scenarios where the control law is expected to vary with time, with topology, and as we learn more about our system, we will denote the system responses $\Phi_x^{(t)}, \Phi_u^{(t)}$ with superscript (t). Define the submatrix

$$\Delta_k^j \left(A, B, \Phi_{x,ji}^{(t)}, \Phi_{u,ji}^{(t)} \right) := \Phi_{x,ji}^{(t)}[k+1] - \sum_{\ell \in \mathcal{N}(j)} A_{j\ell} \Phi_{x,\ell i}^{(t)}[k] - B_j \Phi_{u,ji}^{(t)}[k]$$

where $i, j = 1, \cdots, N_s$, $k = 1, \cdots, T$, and t is the simulation timestep. To construct a topologically-robust controller $\{\Phi_{x,i}^{(t)}, \Phi_{u,i}^{(t)}\}$ for each system i, we proposed solving the following distributed optimization problem in our previous work [**?**].

$$\min f(\Phi_{x,i}^{(t)}, \Phi_{u,i}^{(t)}, Q_i, R_i) = \sum_{k=1}^T \left\| Q\Phi_{x,i}^{(t)}[k] + R\Phi_{u,i}^{(t)}[k] \right\|_1 + r_c \epsilon \tag{18a}$$

$$\text{s.t.} \left\| \sum_{j \in \mathcal{L}(i)} \Delta_k^j \left(A', B, \Phi_{x,ji}^{(t)}, \Phi_{u,ji}^{(t)} \right) \right\| \leq c_i \rho^{k-1}, \forall k \leq T-1, A' = A^* + \sum \xi_\ell \mathcal{A}_\ell, \boldsymbol{\xi} \in \mathcal{P}_t \tag{18b}$$

$$c_i \sum_{k=1}^T \rho^{k-1} \leq \lambda_t^{(i)} + \epsilon, \quad \epsilon > 0 \tag{18c}$$

where $\rho > 0$ and $c_i > 0$, for all i. But in the specific case where the consistent set $\mathcal{P}_t$ is a singleton set, the usual SLS method [27] can be applied with the corresponding A instead of solving (18).

We will modify the case study of the power grid with a hexagon topology considered in the previous subsections. The noise vector $\mathbf{w}$ is generated such that $\|\mathbf{w}\| \leq 0.3$ and we begin with initial conditions $\mathbf{x}_i[0] = (3,0)^T, \mathbf{x}_i[0] = (0,0)^T \ \forall i > 1$. Our hexagon network switches topologically according to a Markov chain with four states, as shown in Figure 8. The true values of the transition probabilities are $p_1 = 1 - p_2 = 0.4$ and $p_3 = 1 - p_4 = 0.8$, but they are unknown to each system in the network. The initial topology configuration is State 1, the fully-connected hexagon network. As a simplification, we assume that the entire set of states are known to each system, so that there are at most four consistent coefficients in $\mathcal{P}_t$ per system over all t. For the control law, we take $\rho = 0.7$, time horizon $T = 5$, and information from 1-hop regions about each system and assume communication delay of 1 timestep.

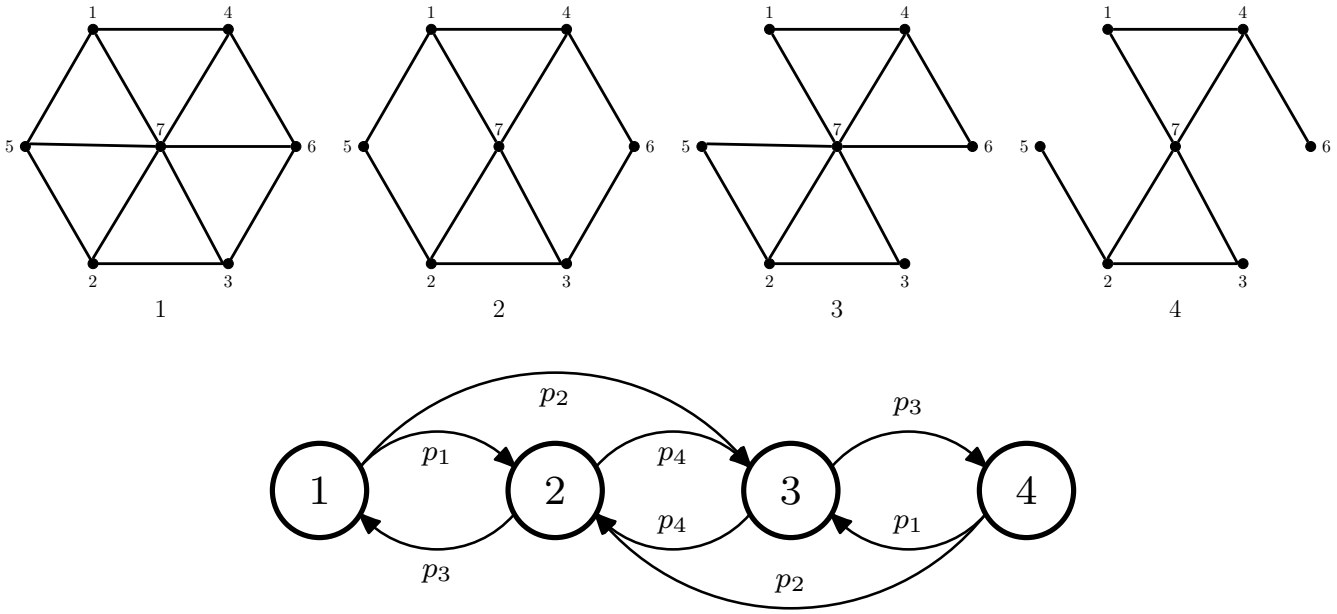

Figure 8: [Top] Hexagon topologies arranged in a Markov chain with four states. [Bottom] The state transition diagram for the four states.

We simulate our algorithm for $T_s = 120$ timesteps and visualize the results in Figure 9. In the first two figures, the black stems indicate the time and state of switching, where a height of $\frac{1}{2}s$ indicates a switch to state s. In the first figure, the number of consistent coefficients versus time is shown for systems $4, 5$, and 6. Each time a switch occurs and the single consistent coefficient corresponding to the previous configuration is no longer consistent, each system resets with the entire set of four coefficients corresponding to all the states of the Markov chain. In the second figure, each system estimates the actual topological configuration of the current time. Note that although each system manages to ultimately estimate the correct state, there is a slight time delay in when it achieves this. The final three rows of Figure 9 show the control law u and two state values x_1, x_2 for each of systems $1, 3$, and 5. We see that it stabilizes the system and is only a little rough during the switching phases when the system is uncertain of the current topology, as expected. The state values for system 5 are more unstable than those of systems 1 and 3 because its local links vary the most across all four topological configurations.

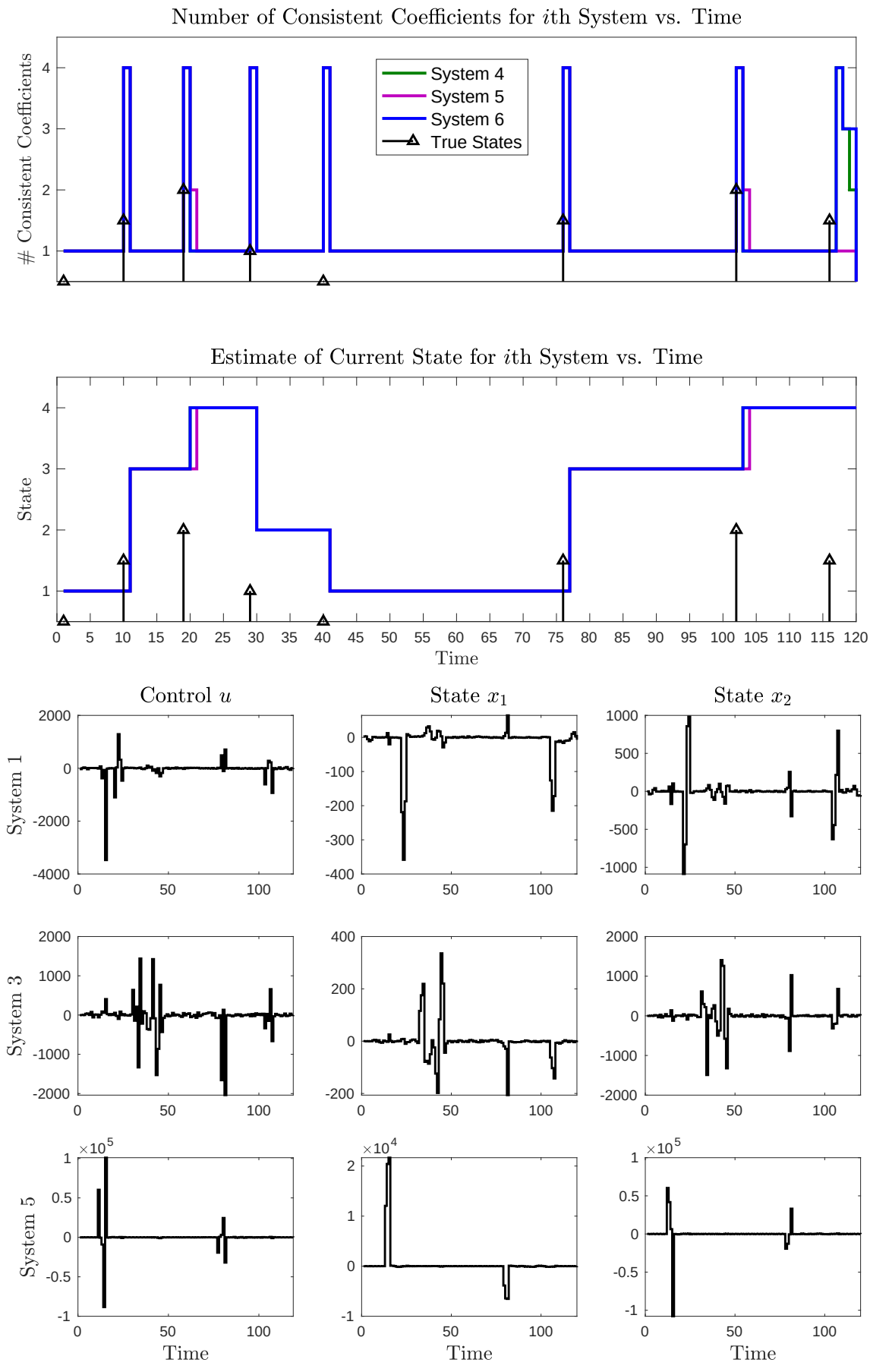
Number of Consistent Coefficients for ith System vs. Time
Consistent Coefficients
System 4
System 5
System 6
True States
Estimate of Current State for ith System vs. Time
State
Time
Control u
State x_1
State x_2
System 1
System 3
System 5
Time

Figure 9

We run the algorithm for a longer simulation time $T_s = 300$ and estimate a transition probability matrix $\hat{P}^{(i)}$ for each $i = 1, \cdots, N_s$. Once it determines which state it is currently in, each system i empirically computes the proportion of transitions that go between different states. It then averages its own estimate with the estimates of its other neighbors and reaches a consensus with them. In this particular simulation, because the network is small and rather densely connected through system 7, all systems converge to the same transition matrix:

$$\hat{P}^{(i)} = \begin{bmatrix} 0 & 0.3333 & 0.6667 & 0 \\ 0.75 & 0 & 0.25 & 0 \\ 0 & 0.1429 & 0 & 0.8571 \\ 0 & 0.6 & 0.4 & 0 \end{bmatrix}$$

By the Law of Large numbers, it would follow that as we take $T_s \to \infty$, the transition probabilities converge to the exact values of $p_k, k = 1, \cdots, 4$.

References

[1] Anderson, J., Doyle, J. C., Low, S., and Matni, N. (2019). System level synthesis. *ArXiv preprint, arXiv:1904.01634*.

[2] Bandyopadhyay, S. and Chung, S.-J. (2018). Distributed bayesian filtering using logarithmic opinion pool for dynamic sensor networks. *Automatica*, 97:7–17.

[3] Boyd, S. and Barratt, C. *Linear controller design: limits of performance*. Prentice Hall Information and System Sciences Series. Prentice Hall.

[4] Cattivelli, F. S. and Sayed, A. H. (2010). Diffusion strategies for distributed Kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, 55(9):2069–2084.

[5] Cheng, L., Wang, Y., Hou, Z.-G., Tan, M., and Cao, Z. (2013). Sampled-data based average consensus of second-order integral multi-agent systems: Switching topologies and communication noises. *Automatica*, 49(5):1458 – 1464.

[6] Chung, S.-J. and Slotine, J. E. (2009). Cooperative robot control and concurrent synchronization of Lagrangian systems. *IEEE Transactions on Robotics*, 25(3):686–700.

[7] Chung, S.-J. and Slotine, J.-E. (2010). On synchronization of coupled Hopf-Kuramoto oscillators with phase delays. In *49th IEEE Conference on Decision and Control (CDC)*, pages 3181–3187.

[8] Dahleh, M. and Díaz-Bobillo, I. (1995). *Control of uncertain systems: a linear programming approach*. Prentice Hall.

[9] Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. (2017). On the sample complexity of the linear quadratic regulator. *ArXiv preprint, arXiv:1710.01688*.

[10] Dean, S., Mania, H., Matni, N., Recht, B., and Tu, S. (2018a). Regret bounds for robust adaptive control of the linear quadratic regulator. *ArXiv preprint, arXiv:1805.09388*.

[11] Dean, S., Tu, S., Matni, N., and Recht, B. (2018b). Safely learning to control the constrained linear quadratic regulator. *ArXiv preprint, arXiv:1809.10121*.

[12] Doyle, J., Glover, K., Khargonekar, P., and Francis, B. (1989). State-space solutions to standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847.

[13] Doyle, J. C., Matni, N., Wang, Y., Anderson, J., and Low, S. (2017). System level synthesis: A tutorial. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2856–2867.

[14] Gao, Y. and Wang, L. (2011). Sampled-data based consensus of continuous-time multi-agent systems with time-varying topology. *IEEE Transactions on Automatic Control*, 56(5):1226–1231.

[15] Ge, X. and Han, Q.-L. (2014). Distributed fault detection over sensor networks with Markovian switching topologies. *International Journal of General Systems*, 43(3-4):305–318.

[16] Han, Q.-L., Yang, F., and Ge, J. (2017). Distributed networked control systems: A brief overview. *Information Sciences*, 380(20):117–131.

[17] Ho, D. and Doyle, J. C. (2019). Scalable robust adaptive control from the system level perspective.

[18] Ioannou, P. A. and Sun, J. (1995). *Robust Adaptive Control*. Prentice-Hall, Inc., USA.

[19] Matni, N., Wang, Y., and Anderson, J. (2017). Scalable system level synthesis for virtually localizable systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 3473–3480.

[20] Nedic, A., Olshevsky, A., Ozdaglar, A., and Tsitsiklis, J. N. (2009). On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517.

[21] Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. In *2007 46th IEEE Conference on Decision and Control*, pages 5492–5498.

[22] Olfati-Saber, R. (2009). Kalman-consensus filter : Optimality, stability, and performance. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 7036–7042.

[23] Olfati-Saber, R. and Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533.

[24] Slotine, J.-J., Wang, W., and Rifai, K. (2004). Contraction analysis of synchronization in networks of nonlinearly coupled oscillators. In *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems*, pages 5–8.

[25] Tao, G. (2014). Multivariable adaptive control: A survey. *Automatica*, 50(11):2737 – 2764.

[26] Vidyasagar, M. (2011). *Control System Synthesis: A Factorization Approach*. Morgan & Claypool Publishers, 1st edition.

[27] Wang, Y.-S., Matni, N., and Doyle, J. (2018). Separable and localized system level synthesis for large-scale systems. *IEEE Transactions on Automatic Control*, 63(12):4234–4249.

[28] Wang, Y.-S., Matni, N., and Doyle, J. C. (2014). Localized LQR optimal control.

[29] Wang, Y.-S., Matni, N., and Doyle, J. C. (2016). A system level approach to controller synthesis.

[30] Wang, Y.-S., You, S., and Matni, N. (2015). Localized distributed Kalman filters for large-scale systems. volume 48.

[31] Xiao, L., Boyd, S., and Lall, S. (2005). A scheme for robust distributed sensor fusion based on average consensus. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70.

[32] Youla, D., Jabr, H., and Bongiorno, J. (1976). Modern Wiener-Hopf design of optimal controllers–Part II: The multivariable case. *IEEE Transactions on Automatic Control*, 21(3):319–338.