Data-Driven Control of Linear-Threshold Network Dynamics

Xuan Wang Jorge Cortés

Abstract— **This paper studies the data-driven stabilization of linear-threshold network models. Our goal is to design a linear state feedback controller to stabilize the system to the origin purely based on data samples, instead of a parametric model. To achieve this, we first establish a data-based representation for the linear-threshold network in an open-loop form. This is done by introducing a map that represents the state-input pair as a transformation of data matrices. We then employ a linear feedback controller in the formulation of the map and obtain a closed-loop data-based representation for the system. To facilitate the design of the feedback gain matrix, we rewrite the dynamics as a switched linear system relying on the special structure of linear thresholding. The design problem then becomes equivalent to solving a system of linear matrix inequalities (LMIs). We analyze the associated computational complexity and, to reduce it, we introduce a different set of LMIs that act as a sufficient condition to obtain the linear control design. Simulations demonstrate the effectiveness of the proposed approach.**

I. Introduction

Linear-threshold network models are widely used in computational neuroscience [1]–[3], social sciences [4], [5], and artificial neural network for deep learning [6], [7]. Driven by the increasing interest in these applications, an emerging research area aims to regulate the dynamical behavior of linear-threshold networks by means of control. Towards this end, traditional system approaches are heavily model-based [8] and rely on precise model parameters to design appropriate control schemes. However, since the application scenarios of linear-threshold models are usually associated with large-scale networks, obtaining their model parameters is challenging. Motivated by this, this paper focuses on developing an approach to control linear-threshold networks that, instead of using the system parametric model, is solely based on data samples.

Literature review: Linear-threshold network models have diverse application backgrounds. In computational neuroscience, they have a long story as descriptors of the mesoscale dynamics of brain networks [1]. Each node of the network represents a population of neurons; the state of the node represents the average firing rate of the population, which is regulated by a linear-threshold activation function to characterize the saturation of firing rates; and the edges are defined by the different neuron populations whose firing rates are interactive. In social sciences, linear-threshold networks have been used to build influence propagation models [4], [5], where the nodes and their associated states represent individuals and their opinions, respectively; a linear-threshold is introduced to each node to gauge the condition when individuals' opinions change; and the edges of the network characterize the relation closeness between individuals. In deep learning, linear-threshold models are also known as modified rectified linear units (RELU with max-limits) [7]. RELUs are generally used within the hidden nodes in the deep neural networks [9], which allows good robustness and versatility for function approximation [10]. Compared with the sigmoid and tanh activation functions, RELU networks have low computational complexity [11] because its gradient is either a constant or zero. By the same reason, ReLU networks do not suffer vanishing gradient problems [12].

Given a linear threshold network model, one can resort to the available characterizations of its stability properties, see e.g., [3] to develop model-based control schemes to stabilize it. However, such approaches require an accurate identification of the model parameters, i.e., the edge weights that characterize the strength of nodal interactions [13]. For large-scale neural or social network systems, such parameters are usually difficult to determine, either from first principles (due to limited knowledge about the system) or through system identification techniques (due to the non-linearity in the model, cf. [14]). This motivates our research here on data-driven control techniques for linear-threshold networks, which are based on synthesizing controllers directly from the system input-output data instead of using parametric models. Along this direction, the results in behavioral theory [15] provide effective tools for the data-driven control of linear systems [16]. The key idea is to use sampled system trajectories to construct a data-based representation that characterizes the dynamic behavior. Based on this, [17] studies the stabilization and optimal control of the system, by associating the gains of linear feedback controllers with the solutions of a linear matrix inequality (LMI) and a semidefinite program (SDP), respectively. This data-driven approach has also been employed in model predictive control [18], [19], including the use of data available across a network [20]. Other works [17], [18], [21] have explored the generalization of data-driven approaches to nonlinear systems using the low-rank approximation technique.

Statement of Contributions: We study the data-driven stabilization of linear-threshold network models via linear feedback controllers. Assuming input-output data of the system is available, we first describe the open-loop system dynamics via a purely data-based representation. This is done by introducing a map that represents the state-input pair as a transformation of data matrices. We build on this

This work was supported by ONR N00014-18-1-2828.

X. Wang is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, xwang64@gmu.edu. J. Cortés is with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, cortes@ucsd.edu

formulation to obtain a data-based representation in closed-loop form for a linear feedback controller. We show how the computation of this representation can be conveniently set up in a way that is amenable to our ultimate objective, which is the design itself of the feedback gain matrix. To do this, our idea is to view the system as a switched system and find a common controller that stabilizes all modes. This allows us to associate the stabilizing gain with the solution to a set of LMIs. Finally, we validate the effectiveness of the proposed algorithm by simulations. We introduce measurement noise to the data samples and show that the proposed approach outperforms the model-based approach in terms of convergence. For reasons of space, the proofs are omitted and will appear elsewhere.

Notation: Let $\mathbb{R}$ denote the set of real numbers. Let $\mathbf{1}_r$ denote the vector in $\mathbb{R}^r$ with all entries equal to 1. Let I_r denote the $r \times r$ identity matrix. We let $\operatorname{col}\{A_1, A_2, \cdots, A_r\} = \begin{bmatrix} A_1^\top & A_2^\top & \cdots & A_r^\top \end{bmatrix}^\top$ be a vertical stack of matrices $A_1, \cdots, A_r$ possessing the same number of columns. Let $\operatorname{diag}\{A_1, A_2, \cdots, A_r\}$ be a diagonal stack of matrices $A_1, \cdots, A_r$. Let $\boldsymbol{x}[i] \in \mathbb{R}$ be the ith entry of vector $\boldsymbol{x}$; correspondingly, let $M[i,j] \in \mathbb{R}$ be the entry of matrix M on its ith row and jth column. We denote by $M^\top$ the transpose of M. For $s > 0$ and $x \in \mathbb{R}$, the threshold function $[x]_0^s$ is defined as

$$[x]_0^s = \begin{cases} s & \text{for} \quad x > s \\ x & \text{for} \quad 0 \leqslant x \leqslant s \\ 0 & \text{for} \quad x < 0 \end{cases}$$

For a vector $\boldsymbol{x} \in \mathbb{R}^r$, $[\boldsymbol{x}]_0^s$ denotes the component-wise application of this definition.

II. Linear Threshold Model

We are interested in a linear-threshold network governed by the following dynamics.

$$\boldsymbol{x}(t+1) = \alpha \boldsymbol{x}(t) + [W\boldsymbol{x}(t) + B\boldsymbol{u}(t)]_0^s, \quad t \in \mathbb{N}. \tag{1}$$

Here $\boldsymbol{x} \in \mathbb{R}^n_{\geqslant 0}$ is the network state, $W \in \mathbb{R}^{n \times n}$ is the network connectivity matrix, characterizing the interactions between different nodes, $\boldsymbol{u}(t) \in \mathbb{R}^m$ and $B \in \mathbb{R}^{n \times m}$ are the external inputs and the associated input matrix, respectively. For each node, its state evolves according to an intrinsic decay rate $\alpha \in (0,1)$, and a linear-threshold activation function denoted by $[\cdot]_0^s$. The input of the linear-threshold function is co-generated by the node's neighbors and external inputs. Network dynamics like (1) arise in the modeling of the dynamical behavior of the firing rates of neuronal populations [1]; the opinion propagation of individuals in social networks [5]; and the use of artificial neural networks to approximate nonlinear dynamics for learning and control tasks [7], [10].

We assume the parameters α and s are known, but the matrices W and B are unknown. Instead, system inputs and states can be sampled from experiments. Let T_d be the total number of available data points, and let $\boldsymbol{x}_d^+(k)$, $\boldsymbol{x}_d(k)$ $\boldsymbol{u}_d(k)$, $k \in \{1, \ldots, T_d\}$ denote the data samples (corresponding to $\boldsymbol{x}(t+1)$, $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$, respectively). We employ the index k as an indicator that distinguishes one data sample from another. It is possible that all the sampling instances of the data are chosen consecutively from a system trajectory, where all the data samples are head-tail connected, i.e., $\boldsymbol{x}_d^+(k)$ of the former data can be used as the $\boldsymbol{x}_d(k)$ of the latter one. In general, we allow data samples to be collected at independent time instances, and even from various trajectories of the same system.

Problem 1: In system (1), suppose α and s are known. Given data samples $\boldsymbol{x}_d(k)$, $\boldsymbol{x}_d^+(k)$ and $\boldsymbol{u}_d(k)$, $k \in \{1, \ldots, T_d\}$, design a linear feedback controller

$$\boldsymbol{u}(t) = K\boldsymbol{x}(t), \tag{2}$$

which asymptotically stabilizes the system to the origin.

The focus of Problem 1 on the data-driven stabilization to the origin is motivated by the following considerations. In neuroscience, for instance, driving the firing rate of a neuronal population to zero means inhibiting its activity, which is something the brain does continuously, while at the same time driving the activity of other neuronal populations to specific patterns of activity. In social science, this happens when the authorities want to shape public opinion by depressing the disagreement from a certain group of people. In control applications where a linear-threshold artificial neural network approximates certain nonlinear dynamics, the stabilization of the latter can be achieved by stabilizing the neural network model. To stabilize the network dynamics (1), the controller we employ has a simple linear feedback form, which can stabilize the system regardless the operating mode of the linear-threshold model. Here, we see solving Problem 1 as a first step towards developing advanced techniques for the network dynamics (1). These includes more sophisticated controller design, such as data-driven model predictive control [22]; and more complex control objectives, such as stabilizing the system to other equilibria (not necessarily the origin) and track various classes of dynamic trajectories.

We assume Problem 1 is solvable. This means that (i) the matrices W and B are such that a controller stabilizing the system of the form (2) exists, see [3] for classes of systems for which this holds, and (ii) the data samples available are sufficiently rich to allow us to determine K without knowing the matrices W and B. In the following, we develop a direct data-driven approach to solve Problem 1. An alternative indirect route would first identify the system and then resort to a model-based controller design (e.g., making the closed-loop system matrix $\widehat{W} = W + BK$ satisfy the sufficient stability condition described in [3]). We pursue the direct route over the indirect one because the performance of the latter is subject to errors from both the model identification phase and the controller design phase.

III. Data-Based Representations of Linear Threshold Models

In this section, we provide data-based reformulations of the system (1) which describe the system update by means of the available data samples, instead of the unknown matrices W and B.

A. Open-loop data-based representation

Let us start by defining $\boldsymbol{z}(t) = \boldsymbol{x}(t+1) - \alpha \boldsymbol{x}(t)$ and $\boldsymbol{p}(t) = \operatorname{col}\{\boldsymbol{x}(t), \boldsymbol{u}(t)\}$. Then, the update (1) can be rewritten as

$$\boldsymbol{z}(t) = [H\boldsymbol{p}(t)]_0^s \tag{3}$$

where

$$H = \begin{bmatrix} W & B \end{bmatrix} = \begin{bmatrix} — & h_1^\top & — \\ — & h_2^\top & — \\ & \vdots & \\ — & h_n^\top & — \end{bmatrix} \in \mathbb{R}^{n \times (n+m)}.$$

Let $\boldsymbol{h} = \operatorname{col}\{h_1, h_2, \ldots, h_n\} \in \mathbb{R}^{n(n+m)}$ be a vectorized system parameter encoding the matrices W and B. Then,

$$H\boldsymbol{p}(t) = \begin{bmatrix} h_1^\top \boldsymbol{p}(t) \\ h_2^\top \boldsymbol{p}(t) \\ \vdots \\ h_n^\top \boldsymbol{p}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}(t)^\top h_1 \\ \boldsymbol{p}(t)^\top h_2 \\ \vdots \\ \boldsymbol{p}(t)^\top h_n \end{bmatrix} = \left(I_n \otimes \boldsymbol{p}(t)^\top\right) \boldsymbol{h}.$$

Thus, equation (3) reads

$$\boldsymbol{z}(t) = \left[\left(I_n \otimes \boldsymbol{p}(t)^\top\right) \boldsymbol{h}\right]_0^s. \tag{4}$$

In order to obtain a data-based representation for the system, a key step is to represent $\boldsymbol{h}$ with the data samples $\{\boldsymbol{x}_d^+(k), \boldsymbol{x}_d(k), \boldsymbol{u}_d(k)\}_{k=1}^{T_d}$. Towards this end, let

$$\mathcal{Z}_d = \begin{bmatrix} \boldsymbol{x}_d^+(1) - \alpha \boldsymbol{x}_d(1) \\ \boldsymbol{x}_d^+(2) - \alpha \boldsymbol{x}_d(2) \\ \vdots \\ \boldsymbol{x}_d^+(T_d) - \alpha \boldsymbol{x}_d(T_d) \end{bmatrix}, \; \mathcal{P}_d = \begin{bmatrix} \left(I_n \otimes \boldsymbol{p}_d^\top(1)\right) \\ \left(I_n \otimes \boldsymbol{p}_d^\top(2)\right) \\ \vdots \\ \left(I_n \otimes \boldsymbol{p}_d^\top(T_d)\right) \end{bmatrix},$$

with $\boldsymbol{p}_d(k) = \operatorname{col}\{\boldsymbol{x}_d(k), \boldsymbol{u}_d(t)\}$, $\mathcal{Z}_d \in \mathbb{R}^{nT_d}$ and $\mathcal{P}_d \in \mathbb{R}^{nT_d \times n(n+m)}$. According to (4), we have

$$\mathcal{Z}_d = [\mathcal{P}_d \boldsymbol{h}]_0^s. \tag{5}$$

Define $f(\mathcal{Z}_d) = \mathcal{P}_d\boldsymbol{h} - [\mathcal{P}_d\boldsymbol{h}]_0^s$, where $f(\cdot)$ represents the part of $\mathcal{P}_d\boldsymbol{h}$ truncated by the linear threshold $[\cdot]_0^s$. Clearly, $f(\mathcal{Z}_d)[i] \neq 0$ only if $\mathcal{Z}_d[i] = s$ or $\mathcal{Z}_d[i] = 0$. Thus, we define a diagonal matrix $E_d \in \mathbb{R}^{nT_d \times nT_d}$ such that for all $i \in \{1, \ldots, nT_d\}$,

$$E_d[i,i] = \begin{cases} 1 & \text{if } \mathcal{Z}_d[i] = s \text{ or } \mathcal{Z}_d[i] = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Then, there exists a vector $v \in \mathbb{R}^{nT_d}$ such that

$$f(\mathcal{Z}_d) = E_d v, \tag{7}$$

which, together with (5), yields $\mathcal{P}_d\boldsymbol{h} = \mathcal{Z}_d + E_d v$. Using the property that $(I - E_d)E_d = E_d - E_d = 0$, one has

$$(I - E_d)\mathcal{P}_d\boldsymbol{h} = (I - E_d)\mathcal{Z}_d. \tag{8}$$

This equation describes the vectorized system parameter $\boldsymbol{h}$ in the form of a data-based equality constraint. We make the following assumption on the data samples.

Assumption 1: (Data richness): Given data samples $\{\boldsymbol{x}_d^+(k), \boldsymbol{x}_d(k), \boldsymbol{u}_d(k)\}_{k=1}^{T_d}$, the matrix $(I - E_d)\mathcal{P}_d$, has full column rank.

Assumption 1 can be directly verified by computation. Note that Assumption 1 becomes easier to satisfy as the number of data samples grows. Based on this assumption, we can combine equations (4) and (8) to obtain a data-based representation of the system dynamics.

Lemma 3.1: (Open-loop data-based representation): Under Assumption 1, let $F : \mathbb{R}^{m+n} \to \mathbb{R}^{n \times nT_d}$ be such that

$$F(\boldsymbol{p}) \cdot (I - E_d)\mathcal{P}_d = I_n \otimes \boldsymbol{p}^\top, \tag{9}$$

for any state-input pair $\boldsymbol{p} = \operatorname{col}\{\boldsymbol{x}, \boldsymbol{u}\}$. Then the dynamics (1) has the following data-based representation,

$$\boldsymbol{x}(t+1) = \alpha \boldsymbol{x}(t) + \left[F(\boldsymbol{p}(t)) \cdot (I - E_d)\mathcal{Z}_d\right]_0^s. \tag{10}$$

For each $\boldsymbol{p}$ in (9), $F(\boldsymbol{p})$ transforms the data matrix $(I - E_d)\mathcal{P}_d$ into the augmented state-input pair $I_n \otimes \boldsymbol{p}^\top$ used by (4). Based on this transformation, we obtain (10). We refer to it as a data-based representation of system (1) because it does not involve the matrices W and B (or their vectorized version $\boldsymbol{h}$), and instead allows, based on the data, to determine the state at the next timestep based on the current state and the control input. We refer to it as open-loop because the state-input pair is arbitrary and does not take into account feedback coupling of the form (2). We address this point next.

B. Closed-loop data-based representation

Given a feedback controller of the form (2), we have the following result.

Lemma 3.2: (Closed-loop data-based representation): Consider the feedback controller $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$. Let Assumption 1 hold. Then the closed-loop form of system (1) with controller (2) has the data-based representation,

$$\boldsymbol{x}(t+1) = \alpha \boldsymbol{x}(t) + \left[G(\boldsymbol{x}(t)) \cdot (I - E_d)\mathcal{Z}_d\right]_0^s \tag{11}$$

where $G : \mathbb{R}^n \to \mathbb{R}^{n \times nT_d}$ satisfies

$$G(\boldsymbol{x}) \cdot (I - E_d)\mathcal{P}_d = I_n \otimes \left(\boldsymbol{x}^\top \begin{bmatrix} I_n & K^\top \end{bmatrix}\right). \tag{12}$$

Equation (11) provides a closed-loop data-based description of system (1), with the existence of G following directly from the existence of F. Note that constraint (12) defines the map G in an implicit way. In the following, we provide an explicit construction of this map.

From the right-hand side of (12), we observe a blocked diagonal matrix resulting from the Kronecker product. We make use of such special pattern as follows. Define

$$\overline{\mathcal{P}}_d = I_n \otimes \begin{bmatrix} \boldsymbol{p}_d^\top(1) \\ \boldsymbol{p}_d^\top(2) \\ \vdots \\ \boldsymbol{p}_d^\top(T_d) \end{bmatrix}$$

and let $T_F \in \mathbb{R}^{nT_d \times nT_d}$ be the elementary matrix that switches the rows of $\mathcal{P}_d$ to obtain $\overline{\mathcal{P}}_d$, i.e.,

$$\overline{\mathcal{P}}_d = T_F \mathcal{P}_d. \tag{13}$$

Since E_d is a diagonal matrix, $\overline{E}_d = T_F E_d T_F^{-1} \in \mathbb{R}^{nT_d \times nT_d}$ is also a diagonal matrix, which we write as

$$\overline{E}_d = \text{diag}\,\{\overline{\mathrm{E}}_1, \cdots, \overline{\mathrm{E}}_n\},$$

with $\overline{\mathrm{E}}_i \in \mathbb{R}^{T_d \times T_d}$. Now, let

$$\begin{bmatrix} \overline{z}_1 \\ \vdots \\ \overline{z}_n \end{bmatrix} = T_F \mathcal{Z}_d,$$

with $\overline{z}_i \in \mathbb{R}^{T_d}$, $i \in \{1, \ldots, n\}$, and define $Z = \text{diag}\,\{Z_1, \cdots, Z_n\}$ by

$$Z_i = \overline{z}_i^\top (I - \overline{\mathrm{E}}_i) \in \mathbb{R}^{1 \times T_d}. \tag{14a}$$

Further define $Q = \text{diag}\,\{Q_1, \cdots, Q_n\}$ by

$$Q_i = \left(I - \overline{\mathrm{E}}_i\right) \begin{bmatrix} \boldsymbol{p}_d^\top(1) \\ \boldsymbol{p}_d^\top(2) \\ \vdots \\ \boldsymbol{p}_d^\top(T_d) \end{bmatrix} \in \mathbb{R}^{T_d \times (m+n)}. \tag{14b}$$

By definition, we have $Q = (I - \overline{E}_d)\overline{\mathcal{P}}_d$.

Lemma 3.3: (Modified closed-loop data-based representation): Consider the feedback controller $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$. Given data matrices Z and Q in (14), let Assumption 1 hold. Then, the system (1) under the feedback controller can be represented by

$$\boldsymbol{x}(t+1) = \alpha \boldsymbol{x}(t) + \left[Z(M\boldsymbol{x}(t))\right]_0^s. \tag{15}$$

where $M \in \mathbb{R}^{nT_d \times n}$ is a matrix satisfying

$$Q^\top M = \mathbf{1}_n \otimes \begin{bmatrix} I_n \\ K \end{bmatrix}. \tag{16}$$

Comparing the statements in Lemmas 3.2 and 3.3, we see that finding $G : \mathbb{R}^n \to \mathbb{R}^{n \times nT_d}$ satisfying (12) can be accomplished by finding the matrix $M \in \mathbb{R}^{nT_d \times n}$ satisfying (16). Regarding the solution of this latter equation, we note that when K is given, M can be readily computed. However, since our ultimate goal is to design a stabilizing controller gain matrix K itself, the solution in M and K of equation (16) poses the challenge of jointly solving for n coupled systems of linear equations. The following result provides a reformulation of the equation showing that K can be expressed as a function of M.

Lemma 3.4: (Decoupling constraints for closed-loop data-based representation): Define matrices $L \in \mathbb{R}^{n \times n}$, $C_1 \in \mathbb{R}^{n \times n(n+m)}$ and $C_2 \in \mathbb{R}^{n \times n(n+m)}$ as

$$L[i,j] = \begin{cases} n & \text{if } i = j \\ -1 & \text{otherwise} \end{cases},$$

$C_1 = \begin{bmatrix} I_n & \mathbf{0}_{n \times n(n+m)-n} \end{bmatrix}$, $C_2 = \begin{bmatrix} \mathbf{0}_{m \times n} & I_m & \mathbf{0}_{m \times (n-1)(n+m)} \end{bmatrix}$, and $\mathcal{L} = L \otimes I_{n+m}$. Then, the constraint (16) can be equivalently written as

$$\mathcal{L} Q^\top M = \mathbf{0}, \tag{17a}$$

$$C_1 Q^\top M = I_n, \tag{17b}$$

$$C_2 Q^\top M = K. \tag{17c}$$

The advantage of the constraint formulation (17) over (16) is that, in the former, K can be expressed as a function of M. In fact, as we vary M among all possible solutions of (17a)-(17b), K in (17c) takes every possible value in $\mathbb{R}^{n \times n}$. This means that we can use (15) and (17a)-(17b) to design the closed-loop behavior of the system, with M as the only variable. Then, to implement the desired closed-loop system, one can compute K using (17c) and apply it to the controller (2) of the closed-loop system. In the above equations, $\mathcal{L}$, C_1, C_2 are constant matrices; and Z, Q are known matrices constructed from the data samples.

IV. Data-Driven Control of Linear Threshold Models

In this section, we introduce an approach to design a feedback gain matrix to stabilize system (1). Note that the system is nonlinear due to the presence of the threshold function. Our idea is to view the system as a switched system and design a common linear feedback controller that stabilizes all modes.

A. LMI-based design of feedback gain matrix

We start by rewriting the system (15) into the following equivalent form

$$\begin{aligned} \boldsymbol{x}(t+1) &= \alpha \boldsymbol{x}(t) + \left[ZM\boldsymbol{x}(t)\right]_0^s \\ &= \alpha \boldsymbol{x}(t) + \overline{R}(\boldsymbol{x}(t)) ZM \boldsymbol{x}(t), \end{aligned} \tag{18}$$

where $\overline{R}(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ is a diagonal matrix with:

$$\overline{R}(\boldsymbol{x})[i,i] \triangleq \begin{cases} \frac{([ZM\boldsymbol{x}]_0^s)[i]}{(ZM\boldsymbol{x})[i]} & \text{if } (ZM\boldsymbol{x})[i] > s \\ & \quad \text{or } (ZM\boldsymbol{x})[i] < 0 \\ 1 & \text{otherwise} \end{cases}$$

For any $\boldsymbol{x}$, one has $0 \leqslant \overline{R}(\boldsymbol{x})[i,i] \leqslant 1$, $i \in \{1, \ldots, n\}$. The representation (18) describes the dynamics as a switched system. Therefore, we find it convenient to define $R_i \in \mathbb{R}^{n \times n}$, $i \in \{1, \ldots, 2^n\}$ as the vertices of the unit hypercube $[0,1]^n$, such that $R_i[j] \in \{0, 1\}$, $j \in \{1, \ldots, n\}$.

Theorem 4.1: (Data-driven synthesis via LMIs): If there exist matrices $\overline{P} \in \mathbb{R}^{n \times n}$ and $S \in \mathbb{R}^{nT_d \times n}$ satisfying the following conditions

$$\begin{bmatrix} \overline{P} & \left(\alpha \overline{P} + R_i Z S\right)^\top \\ \alpha \overline{P} + R_i Z S & \overline{P} \end{bmatrix} \succ 0 \tag{19a}$$

$$\mathcal{L} Q^\top S = 0 \tag{19b}$$

$$C_1 Q^\top S = \overline{P} \tag{19c}$$

for all R_i, $i \in \{1, \ldots, 2^n\}$, then system (1) can be stabilized by the controller $\boldsymbol{x}(t) = K\boldsymbol{u}(t)$, where

$$K = C_2 Q^\top S \overline{P}^{-1}. \tag{20}$$

Note that Theorem 4.1 only provides a sufficient condition for stabilizing system (1). The conservativeness is caused by the relaxation from the vector $\overline{R}(\boldsymbol{x}(t))$ in (18) to the

hypercube defined by the R_i, $i \in \{1, \dots, 2^n\}$ in (19a), where coupling between $\overline{R}(\boldsymbol{x})$ and $\boldsymbol{x}$ is ignored.

Remark 4.2: (Computational complexity of solving LMIs): LMIs with linear constraints can be efficiently solved using existing algorithms, cf. [23]. The computational complexity of solving a single (19a) is polynomial in T_d, which is the number of data samples and determines the sizes of matrices Z and Q. ▫

V. Simulation

In this section, we present an example from computational neuroscience [24] to validate the effectiveness of the proposed results. We simulate a brain neural network with $n = 8$ nodes, with each state representing the firing rate of a population of neurons. The dimension of the input $\boldsymbol{u}$ is chosen as $m = 8$. Given the state/input dimensions of the system, we first create matrices $W \in \mathbb{R}^{8\times 8}$ and $B \in \mathbb{R}^{8\times 8}$. We make sure the entries of the matrices are consistent with *Dale's law* [25], i.e., each column of W is either non-negative or non-positive depending on the excitatory or inhibitory properties of the nodes. The values of these entries are randomly chosen from $[0 \quad 0.15]$ or $[-0.1 \quad 0]$, with uniform distributions. For $B \in \mathbb{R}^{8\times 8}$, all its entries are randomly chosen from $[-0.1 \quad 0.15]$ with uniform distributions. We set $\alpha = 0.9$ and $s = 2$. The unforced system has the origin as an unstable equilibrium point.

Based on W, B, α and s, we create data samples, for $k \in \{1, \dots, T_d\}$ and $T_d = 250$. In the simulation, we intentionally introduce measurement noise. For different k, system states $\widetilde{\boldsymbol{x}}_d(k)$ and inputs $\widetilde{\boldsymbol{u}}_d(k)$ are chosen independently, i.e., the entries of $\widetilde{\boldsymbol{x}}_d(k)$ are randomly chosen from $[0 \quad 10]$; the entries of $\widetilde{\boldsymbol{u}}_d(k)$ are randomly chosen from $[0 \quad 5]$, with uniform distributions. For each pair of $\widetilde{\boldsymbol{x}}_d(k)$ and $\widetilde{\boldsymbol{u}}_d(k)$, we compute $\widetilde{\boldsymbol{x}}_d^+(k)$ based on the following discrete-time system model

$$\widetilde{\boldsymbol{x}}_d^+(k) = \alpha(\widetilde{\boldsymbol{x}}_d(k)) + [W(\widetilde{\boldsymbol{x}}_d(k)) + B(\widetilde{\boldsymbol{u}}_d(k))]_0^s$$

in accordance with (1). To simulate the effect of measurement noise, we assume $\widetilde{\boldsymbol{u}}_d(k)$, $\widetilde{\boldsymbol{x}}_d(k)$ and $\widetilde{\boldsymbol{x}}_d^+(k)$ are the real inputs/states of a system in experiments, while $\boldsymbol{u}_d(k) = \widetilde{\boldsymbol{u}}_d(k) + \mu_u(k)$, $\boldsymbol{x}_d(k) = \widetilde{\boldsymbol{x}}_d(k) + \mu_x(k)$ and $\boldsymbol{x}_d^+(k) = \widetilde{\boldsymbol{x}}_d^+(k) + \mu_{x^+}(k)$ are the collected data samples. Here, all $\mu_{(\cdot)}(k)$'s are randomly generated with $\|\mu_{(\cdot)}(k)\|_\infty \leqslant 0.01$, for all $k \in \{1, \dots, T_d\}$. By validation, the data samples satisfy Assumption 1.

Given data samples $\{\boldsymbol{u}_d(k), \boldsymbol{x}_d(k), \boldsymbol{x}_d^+(k)\}_{k=1}^{250}$, we employ two approaches to design the state feedback controller $\boldsymbol{u}(t) = K\boldsymbol{x}(t)$. One is the data-driven approach proposed in Theorem 4.1; the other one is the indirect approach which first identifies the system model parameters, then utilizes a model-based method to design the controller.

In order to make comparisons with the indirect method, we consider the following semi-definite programming (SDP)

$$\text{minimize} \quad \|ZS\|_{\mathrm{F}} \tag{21a}$$

$$\begin{bmatrix} I_n & (\alpha I_n + nR_iZS)^\top \\ \alpha I_n + nR_iZS & I_n \end{bmatrix} \succ 0 \tag{21b}$$

$$\mathcal{L}Q^\top S = 0 \tag{21c}$$

$$C_1 Q^\top S = I_n \tag{21d}$$

for all R_i, $r \in \{1, \dots, 2^n\}$, where $\|ZS\|_{\mathrm{F}}$ is the Frobenius norm. The motivation for problem (21) is as follows. From system update (15), due to the non-negative property of the linear threshold, the fastest way that the system can converge to zero is when $[Z(M\boldsymbol{x}(t))]_0^s = 0$. This observation leads us to seek the minimization of $\|ZM\|_{\mathrm{F}} = \|ZS\overline{P}^{-1}\|_{\mathrm{F}}$. Clearly, this objective function is not convex. Therefore, we fix $\overline{P} = I_n$, which yields (21a). Accordingly, (19) leads to (21b)-(21d). We validate that problem (21) is solvable, and compute the controller gain K according to (20). Given a random initial state $\boldsymbol{x}(0)$, we show in Figure 1 the system trajectory of the closed-loop system. One can observe that all the system states are stabilized.

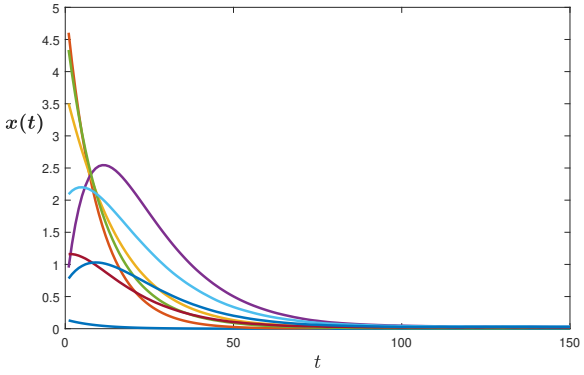

Fig. 1. Data-driven stabilization of an 8-node network. The synthesis of the feedback gain matrix is based on solving the LMIs specified in Theorem 4.1.

For the indirect model-based approach, we first identify the system parameters. This is done by solving equation (8), where $\boldsymbol{h}$ is the vectorized system parameter encoding matrices W and B. We denote the identified parameters as W_I and B_I, which are different from the true parameters due to measurement noise. To design the feedback controller, we employ the following SDP

$$\text{minimize} \quad \|W_I + B_I K\|_{\mathrm{F}} \tag{22a}$$

$$\begin{bmatrix} I_n & \star \\ \alpha I_n + n\widetilde{R}_k(W_I + B_I K) & I_n \end{bmatrix} \succ 0 \tag{22b}$$

for all $\widetilde{R}_k$, $k \in \{1, \dots, n\}$, where $\star$ denotes the symmetric block of the matrix. Here (22a)-(22b) are the same objective function and LMIs as (21a)-(21b), but arise from the following model-based representation

$$\boldsymbol{x}(t+1) = \alpha\boldsymbol{x}(t) + [(W + BK)\boldsymbol{x}(t)]_0^s .$$

Since in this case K is the direct variable, no extra linear constraints are required. We plot in Figure 2 the system trajectory of the closed-loop system starting from the same initial condition as in Figure 1, and observe that all the

system states are stabilized.

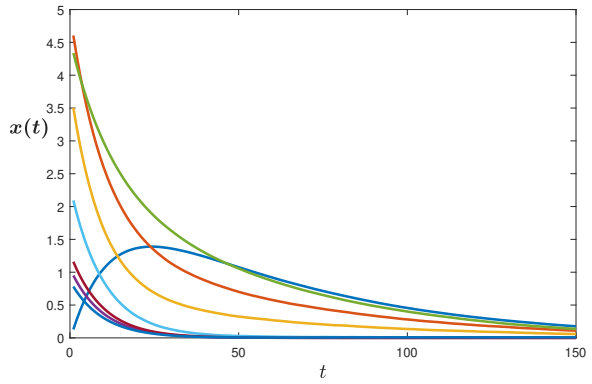

Fig. 2. Model-based stabilization of a 8-node network. The synthesis of the feedback gain matrix is based on first identifying the system parameters, then using this representation to formulate a similar set of LMIs to those in Theorem 4.1.

Comparing Figures 1 and 2, one can see that in the presence of measurement noise, the data-driven approach results in a better convergence speed than the indirect model-based approach. It is worth mentioning that, in the absence of any noise measurement, both approaches result in the same stabilizing feedback gain matrix.

VI. Conclusions and Future Work

We have studied the data-driven stabilization of linear-threshold network models. We have established both open-loop and closed-loop data-based representations of the dynamics that rely on the availability of suitable maps that transform the data matrices into a convenient form. Regarding the latter representation, such map can be easily obtained when the feedback gain matrix is specified a priori. We have also shown how the computation of this map can be formulated in a way that is amenable to our ultimate objective, which is the design itself of the feedback gain matrix. We have built on this to identify a set of LMIs whose solution provides a solution to the data-driven synthesis of a linear stabilizing controller. Finally, we have validated the effectiveness of the proposed approach in simulation. Future work will investigate controller designs beyond purely linear ones, the data-driven stabilization of equilibria other than the origin, the trajectory tracking of desirable signal patterns, and extend our results to scenarios where access to full-state information is not available.

References

[1] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, ser. Computational Neuroscience. Cambridge, MA: MIT Press, 2001.

[2] C. Curto, J. Geneson, and K. Morrison, "Fixed points of competitive threshold-linear networks," *arXiv preprint arXiv:1804.00794*, 2018.

[3] E. Nozari and J. Cortés, "Hierarchical selective recruitment in linear-threshold brain networks. Part I: Intra-layer dynamics and selective inhibition," *IEEE Transactions on Automatic Control*, vol. 66, no. 3, pp. 949–964, 2021.

[4] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *IEEE International Conference on Data Mining*, Sydney, Australia, Dec 2010, pp. 88–97.

[5] Y. D. Zhong, V. Srivastava, and N. E. Leonard, "On the linear threshold model for diffusion of innovations in multiplex social networks," in *IEEE Conf. on Decision and Control*, Melbourne, Australia, Dec. 2017, pp. 2593–2598.

[6] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1229–1262, 2014.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[8] Z. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[9] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of ICML*, Haifa, Israel, 2010, pp. 807–814.

[10] D. Foster, T. Sarkar, and A. Rakhlin, "Learning nonlinear dynamical systems from a single trajectory," in *Annual Conference on Learning for Dynamics and Control*. PMLR, 2020, pp. 851–861.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[12] M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *International Conference on Control and Robotics Engineering*, Bangkok, Thailand, 2017, pp. 201–206.

[13] D. Ehrens, D. Sritharan, and S. V. Sarma, "Closed-loop control of a fragile network: application to seizure-like dynamics of an epilepsy model," *Frontiers in Neuroscience*, vol. 9, p. 58, 2015.

[14] X. Wang and J. Cortés, "Data-driven reconstruction of firing rate dynamics in brain networks," in *IEEE Conf. on Decision and Control*, Austin, Texas, Dec. 2021, pp. 6456–6461.

[15] J. C. Willems, P. Rapisarda, I. Markovsky, and B. L. M. De Moor, "A note on persistency of excitation," *Systems & Control Letters*, vol. 54, no. 4, pp. 325–329, 2005.

[16] T. M. Maupong and P. Rapisarda, "Data-driven control: A behavioral approach," *Systems & Control Letters*, vol. 101, pp. 37–43, 2017.

[17] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality and robustness," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 909–924, 2019.

[18] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: in the shallows of the DeePC," in *European Control Conference*, Naples, Italy, June 2019, pp. 307–312.

[19] J. Berberich, J. Köhler, A. M. Müller, and F. Allgöwer, "Data-driven model predictive control with stability and robustness guarantees," *arXiv preprint arXiv:1906.04679*, 2019.

[20] A. Allibhoy and J. Cortés, "Data-based receding horizon control of linear network systems," *IEEE Control Systems Letters*, vol. 5, no. 4, pp. 1207–1212, 2021.

[21] C. De Persis and P. Tesi, "Designing experiments for data-driven control of nonlinear systems," *IFAC-PapersOnLine*, vol. 54, no. 9, pp. 285–290, 2021.

[22] D. Piga, M. Forgione, S. Formentin, and A. Bemporad, "Performance-oriented model learning for data-driven mpc design," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 577–582, 2019.

[23] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014, available at http://cvxr.com/cvx.

[24] E. Nozari and J. Cortés, "Hierarchical selective recruitment in linear-threshold brain networks. Part II: Inter-layer dynamics and top-down recruitment," *IEEE Transactions on Automatic Control*, vol. 66, no. 3, pp. 965–980, 2021.

[25] J. Eccles, "Chemical transmission and Dale's principle," in *Progress in Brain Research*. Elsevier, 1986, vol. 68, pp. 3–13.