Reconstructing network dynamics of coupled discrete chaotic units from data:
Supplemental Material

Irem Topal and Deniz Eroglu
Faculty of Engineering and Natural Sciences, Kadir Has University, 34083 Istanbul, Turkey

Contents

I. CONNECTIVITY MATRIX IN TERMS OF LAPLACIAN

Network dynamics with diffusive interaction is given by,

$$\boldsymbol{x}_i(t+1) = \boldsymbol{f}(\boldsymbol{x}_i(t)) + \sum_{j=1}^{n} w_{ij}[\boldsymbol{H}(\boldsymbol{x}_j) - \boldsymbol{H}(\boldsymbol{x}_i)] \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{f}\colon \mathbb{R}^m \to \mathbb{R}^m$ is chaotic dynamics of isolated nodes and $\boldsymbol{H}$ is a diffusive coupling function [1]. $\boldsymbol{W} = [w_{ij}] \in \mathbb{R}^{n\times n}$ is a weighted and directed adjacency matrix. Diffusive nature of the interaction allows represent the coupling in terms of the Laplacian matrix.

$$\begin{aligned}\sum_{j=1}^{n} w_{ij}[\boldsymbol{H}(\boldsymbol{x}_j) - \boldsymbol{H}(\boldsymbol{x}_i)] &= \sum_{j=1}^{n} w_{ij}\boldsymbol{H}(\boldsymbol{x}_j) - \boldsymbol{H}(\boldsymbol{x}_i)\sum_{j=1}^{n} w_{ij} \\ &= \sum_{j=1}^{n} w_{ij}\boldsymbol{H}(\boldsymbol{x}_j) - k_i\boldsymbol{H}(\boldsymbol{x}_i) \\ &= \sum_{j=1}^{n}(w_{ij} - \delta_{ij}k_i)\boldsymbol{H}(\boldsymbol{x}_j)\end{aligned} \tag{2}$$

where $k_i = \sum_j w_{ij}$ is the incoming degree of node i and δ_{ij} is the Kronecker delta. By introducing the Laplacian matrix, $\boldsymbol{L}$ with $L_{ij} = \delta_{ij}k_i - w_{ij}$ we obtain,

$$\boldsymbol{x}_i(t+1) = \boldsymbol{f}(\boldsymbol{x}_i(t)) - \sum_{j=1}^{n} L_{ij}\boldsymbol{H}(\boldsymbol{x}_j) \tag{3}$$

Then we can rewrite Eq. (3) in a compact form as,

$$\boldsymbol{X}(t+1) = \boldsymbol{F}(\boldsymbol{X}(t)) - (\boldsymbol{L}\otimes\boldsymbol{H})(\boldsymbol{X}(t)), \tag{4}$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]^T$, $\boldsymbol{F}(\boldsymbol{X}) = [\boldsymbol{f}(\boldsymbol{x}_1), \cdots, \boldsymbol{f}(\boldsymbol{x}_n)]^T$ and $\otimes$ is the Kronecker product [2].

II. NETWORK RECONSTRUCTION SCHEME ON A TOY MODEL

We present a step-by-step reconstruction algorithm using weakly coupled chaotic maps on a directed and weighted network of size $n = 20$ Fig. 1(a). The network is heterogeneous; the low-degree nodes are abundant, and the hub is the rarest, as illustrated in the network's in-degree distribution Fig. 1(b). The network has Rulkov maps interacting with each other through their u-components diffusively:

$$\begin{aligned}u_i(t+1) &= \frac{\beta}{1+u_i(t)^2} + v_i(t) - \sum_{j=1}^{n} L_{ij}u_j(t) \\ v_i(t+1) &= u_i(t) - \nu u_i(t) - \sigma\end{aligned} \tag{5}$$

where the constant parameters $\beta = 4.1$ and $\nu = \sigma = 0.001$ are fixed for chaotic bursting dynamics. We first simulate the system 15000 time steps and discard the first 14000 steps as a transient. We should note that the local dynamics (Rulkov map), the coupling function (u-coupling) and the connectivity matrix (L_{ij}) are all unknowns during the procedure; however, we present every piece of information in this toy model to show the method rigorously.

Fig. 1(c) shows the 2-dimensional data coming from the hub and one of the low degree nodes. We present the return maps of the same nodes in Fig. 1(d), the dispersion of the hub node due to the dominant coupling effect can be seen. On the other hand, we compute the pairwise Pearson correlation coefficients s_{ij} between the time series, see Fig. 1(e). Then we obtain the histogram $P(S)$, Fig. 1(f), by the row-sum of the correlation matrix in Fig. 1(e). Given the chaotic nature of the Rulkov maps, correlations between time series does not give any information about the network connectivity.

The reconstruction procedure is given by the following steps:

1. **Classification of the nodes.**

   The separability of low-degree nodes and hubs is crucial since the reconstruction procedure is not applicable if we cannot classify which nodes are low-degree or hubs at the beginning of the reconstruction recipe. The classification leads us to learn the isolated dynamics $\boldsymbol{f}$ and the coupling function $\boldsymbol{H}$ as follows: we, first, assume that the coupling is *weak*, meaning that if we classify the low-degree nodes, we can consider their dynamics as local dynamics $\boldsymbol{f}$ with some small fluctuations. If we classify which node is the hub and $\boldsymbol{f}$, then we can learn the coupling function $\boldsymbol{H}$ by discarding $\boldsymbol{f}(\boldsymbol{x}_h(t))$ from the hub's time series $\boldsymbol{x}_h(t+1)$ using the cumulative effect of interactions. Therefore, our first aim is to classify the nodes regarding their degrees.

   For the classification, we first learn a governing equation for each time series using sparse regression methods (Supp. Mat. Sec. VII) with a basis composed of candidate functions (Supp. Mat. Sec. VIII). After obtaining the n predicted models, we measure the difference between the models using normalized Euclidean distance between the coefficients of predicted models, d_{ij}. A distance matrix $D = [d_{ij}]$ summarizes the mismatches between the predicted models. Fig. 1(g). The row-sum of the distance matrix $D_i = \sum_j d_{ij}$ gives a histogram $P(D)$, and each bin of the histogram contains nodes with a similar degree (Fig. 1(h)).

   It is also important to note that weak coupling is also crucial to avoid possible synchronization in the network dynamics. If the system is synchronized, the time series will be identical; in other words, all the predicted models will be identical. This will make the classification impossible. Therefore, the weak coupling is quite crucial for this algorithm to work as desired.

2. **Learning the local dynamics.** To mimic brain network dynamics, we assume that the connectivity matrix $\boldsymbol{W} = [w_{ij}]$ represents a scale-free network whose degree distribution follows a power law; in other words, most of the nodes have small degrees $k \sim n^{\epsilon}$, and some nodes are hubs with degrees $k \sim n^{\frac{1}{2}+\epsilon}$ where ϵ is an arbitrarily small number. Therefore, the highest bin of the histogram (the most abundant frequencies) contains the low-degree nodes' models since there are many low-degree nodes (the orange bar in Fig. 1 (h)). As it is likely to have nodes having incoming links, $k_i > 0$, in the set of low-degree nodes, the models can contain some small fluctuations. Thus, we average the predicted coefficients of the low-degree nodes' models to infer the local dynamics as accurately as possible.

3. **Learning the coupling function.** Using the scale-free network topology assumption as in the previous section, we understand that the lowest bin of the histogram (the rarest frequencies) contains the most different models, which belong to the hubs (the blue bar in Fig. 1(h)). Then discarding the local dynamics of the hub ($\boldsymbol{f}(\boldsymbol{x}_h(t))$ from the hub measurement ($\boldsymbol{x}_h(t+1)$) gives the dominant coupling effect, which we aim to learn. Fig. 1(i) presents the remaining coupling effect. We fit a function to the data and learn an approximate coupling function plus an integration constant arising from the reduction theorem. To be more precise with the integration constant, for instance, if the diffusive coupling function is given as $h(x, y) = \phi(y) - \phi(x)$ then the effective coupling V for the expending maps is $V = \int h(x, y)d\mu(y) = -\phi(x) + C$ where C is the integration constant, which we call a *possible linear shift* for the numerically recovered effective coupling V. This is easy to estimate from the fitted coupling (Fig. 1 (i)) since we assume that $\phi(0) = 0$. Therefore, the appearing linear shift is the integration constant C, and its value can vary for different dynamical networks. Furthermore, the effective coupling is an approximation, which becomes exact when the network model is considered as a directed star graph with the link directions from low-degree nodes to the hub at the thermodynamical limits as $n \to \infty$. We have an approximated effective coupling for tree-like graphs, as scale-free graphs, under our assumptions related to node degrees (mentioned above). The size of the bounded fluctuation term κ_i in the reduction (Eq. (2) in the main text) can be considered as the measure of approximation.

   Here, the weak coupling and chaotic dynamics assumptions take an important role again for more general networks to preserve the statistical behavior of the nodes' dynamics. Although using chaotic oscillators helps to keep the state distribution of the states stable, this property can be destroyed by a large coupling. Therefore, if the network is denser, then the coupling constant should be scaled to a smaller coupling strength for our reduction theorem to work. For instance, if the hubs are rare in a scale-free network, the coupling strength must be scaled by $n^{1/2}$. If all the nodes are hubs, in other words, if the network is all-to-all connected, then the coupling strength must be scaled by n. In our work, we scaled the weights in $\boldsymbol{W}$ with $n^{1/2}$, as we considered scale-free networks (containing many low-degree nodes and rare hubs) since the work focuses on reconstructing brain networks. It is also important to remind that, for the node classification step, the synchronization of the nodes disallows the reconstruction since the machine learning-based methods cannot distinguish the data source between synchronized oscillators. Consequently, the chaotic oscillators also play an important role here since if we have periodic and identical oscillator networks, any positive coupling strength can synchronize the model, which is undesired for the reconstruction part. Therefore, the weak coupling strength and the chaotic dynamical regime are crucial assumptions for the theory and also for the numerical steps to learn the coupling function $\boldsymbol{H}$.

4. **Learning the connectivity matrix.** After learning the functions $\boldsymbol{f}$ and $\boldsymbol{H}$, we define $\boldsymbol{Y} = \boldsymbol{X}(t+1) - \boldsymbol{F}(\boldsymbol{X}(t))$, so Eq. (4) can be written as $\boldsymbol{Y} = \boldsymbol{G}\boldsymbol{X}$ where $\boldsymbol{G} = -(\boldsymbol{L} \otimes \boldsymbol{H})$. Therefore, learning the links becomes a regression problem by solving the linear equation $\boldsymbol{Y}^T = \boldsymbol{X}^T \boldsymbol{G}^T$. Here we employ a compressed sensing approach using ℓ_1-norm, called LASSO, to solve the equation since we are interested in reconstructing large network dynamics from short data. In other words, we aim to solve the problem when the length of the time series $< m \times n$ where m is the dimension of the local dynamics and n is the system size. The problems under this condition are called underdetermined regression problems. Finally, we identify the exact Laplacian matrix of the network by solving the linear equation (Fig. 1 (j)).

   Note that, in the previous steps of the procedure, the potential interaction functions of two nodes (or higher-order interactions if the hypernetworks are interested) are not involved in the basis library. Meaning that the size of the library is not grown exponentially. Therefore, the algorithm does not require a longer time series to reconstruct the network dynamics as the network size, n, is increasing. In this last step, we also only sparsely solve a linear equation where $\boldsymbol{G} \in \mathbb{R}^{mn \times mn}$, which is a square matrix. The discussion on the relation between the length of the time series and the sparsity can be found in Supp. Mat. Sec. XII.

III. ASSESSING THE PERFORMANCE

We use two standard indices to assess the reconstruction performance of our procedure: the fraction of the false negatives out of the positives (FNR) and the fraction of false positives out of the negatives (FPR). These metrics correspond to two types of errors in network reconstruction: missing a link where a link indeed exists (false negatives) or assigning a link between two nodes when they are not connected (false positives):

$$FNR = \frac{FN}{P} \text{ and } FPR = \frac{FP}{N}$$

where FN evaluates errors on the non-zero terms (P) of the correct matrix (underestimation) and FP evaluates errors of the zero terms (N) (overestimation). We count FN and FP links by $\sum_{i,j} \Theta(|L_{ij} - \hat{L}_{ij}| > \epsilon)$ to assess the accuracy of the link strengths where $\hat{L}_{ij}$ is the predicted Laplacian and ϵ is a tolerance value set as 10 times smaller than the smallest nonzero value in $\boldsymbol{L}$. We use a fixed tolerance value as $\epsilon = 0.0001$ for all analyses.

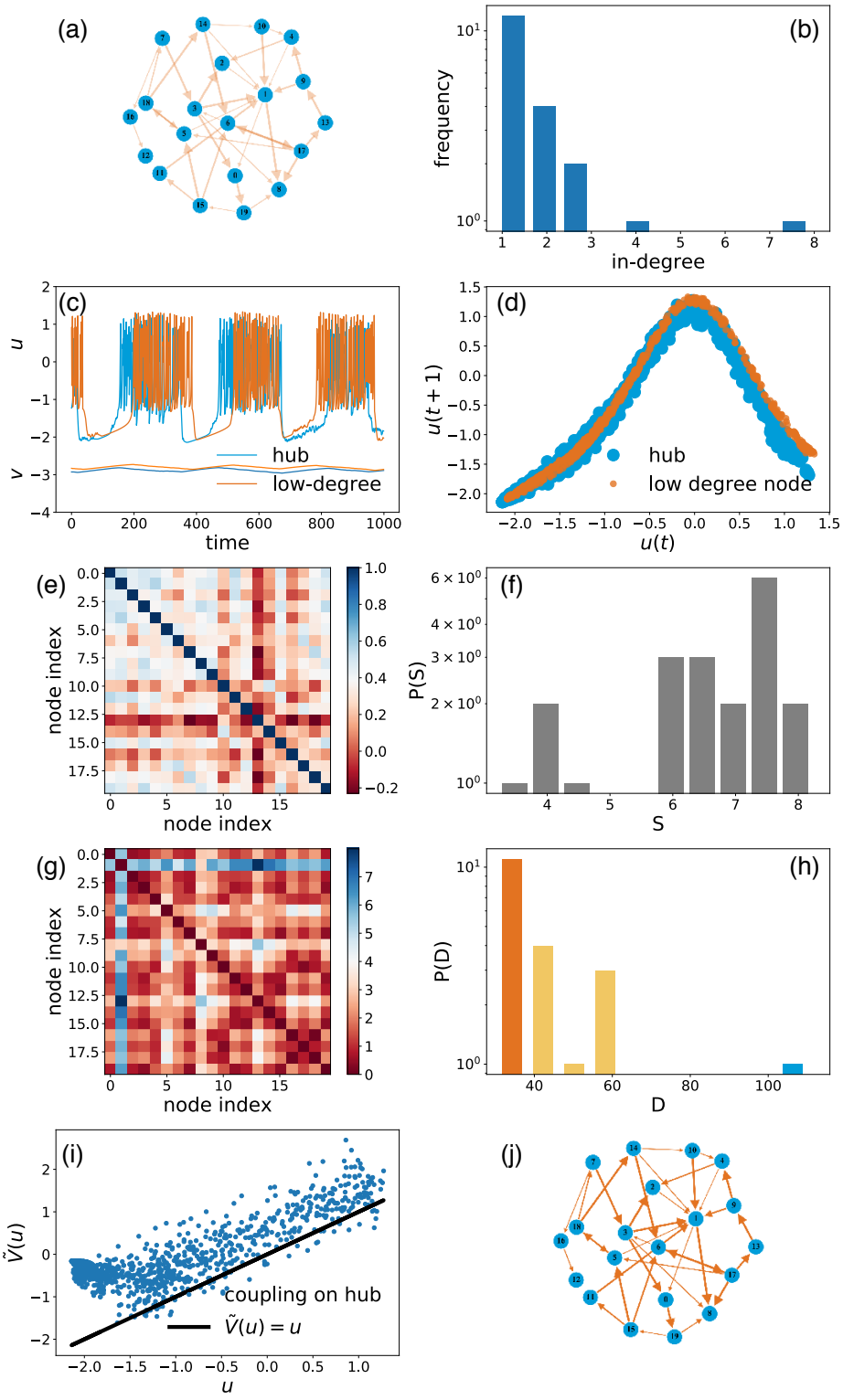

FIG. 1: (a) A directed and weighted network. As the network is unknown initially, its links were illustrated indistinctly. (b) Corresponding in-degree distribution of the network. (c) Time series of a low-degree node and the hub. (d) The return maps of these time series, the hub's data is dispersed according to the low-degree node due to the coupling effect accumulated on the hub. (e) Pairwise Pearson correlation matrix of the time series, which shows no similarity between, even connected, nodes dynamics due to the chaos. (f) The histogram obtained by the row-sum of the time series similarity matrix (subplot (e)) does not reflect any information about the network's original degree distribution; therefore, it is impossible to identify a low-degree node or the hub. (g) Pairwise distances between predicted models were obtained in the first step of our reconstruction procedure. Nodes with similar in-degrees show high similarity. (h) The histogram of the row-sum of the distance matrix (subplot (g)), which does also not contain the exact in-degree distribution of the network (subplot (b)), however, allows for the low-degree and hub node identification. The abundant models (the highest bin in (h)) represent the low-degree nodes, and these models represent the local dynamics, $\boldsymbol{f}$. The most distinct model (the lowest bin in (h)) belongs to the hub node. We learn the coupling function by filtering the local dynamics from the hub data. (i) The remaining coupling effect on the hub is plotted, and the reduction theorem recovers the coupling function. (j) After recovering the local dynamics and the coupling function, we solve the linear equation $y = Ax$ by sparse regression to obtain the network structure.

IV. MOUSE NEO-CORTEX NETWORK

We choose a neuronal connectivity consisting 1029 nodes representing a small volume of a young adult mouse neocortex [3]. The network is directed and contains multi-edges between some nodes. We considered the number of these multi edges between nodes as weights. Furthermore, the network contains some disconnected parts, we removed them and kept only the giant component. The resulting weighted and directed network has $n = 987$ nodes and $m = 1536$ links, which is illustrated in Fig. 2 (a) with its in-degree distribution in Fig. 2 (b). We downscale entries of the connectivity matrix by a small constant 0.1, and then normalize by in-degree of hub, k_h, so that the network is not synchronized.

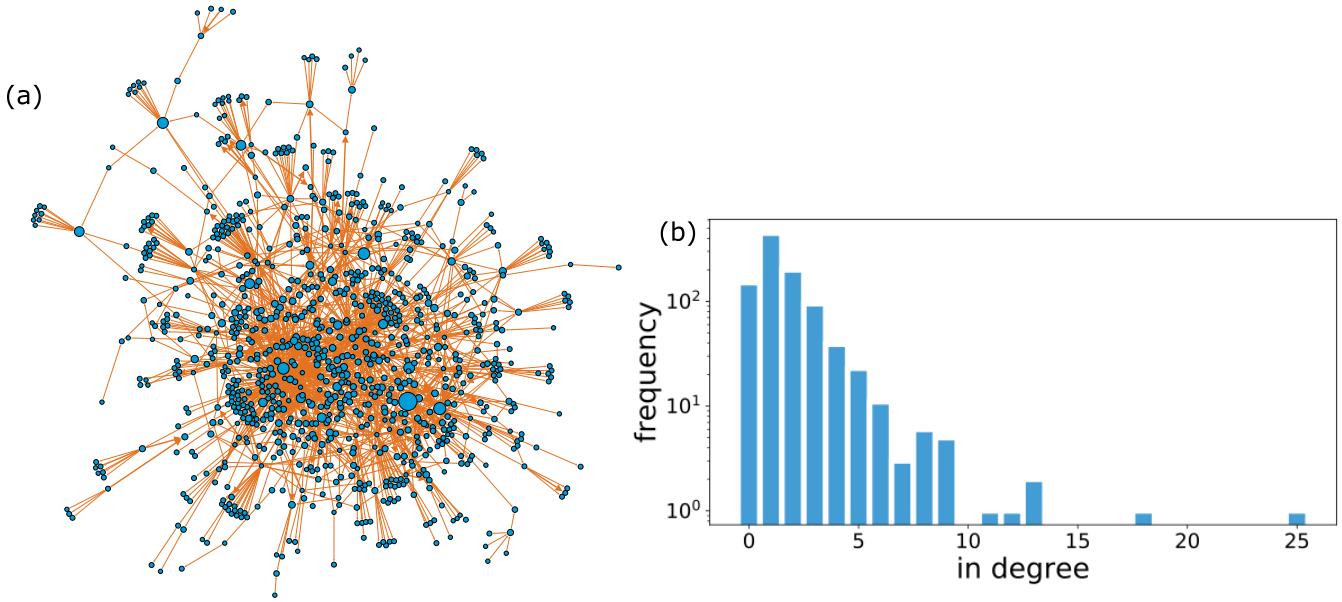

FIG. 2: (a) Pre-processed directed and weighted mouse neo-cortex network. The sizes of the nodes denote their in-coming degrees and the thickness of the arrows denote their weights. (b) In-degree distribution of the associated real network.

V. APPLICATIONS ON MOUSE NEO-CORTEX NETWORK

This section presents the results of the reconstruction procedure on various dynamical systems, namely Rulkov map, Hénon map and Tinkerbell map. For all the systems, we use mouse neo-cortex network which described in Section IV.

A. Rulkov map

We have already presented Rulkov map results in the main text. In our reconstruction scheme, we assume that a low-degree node can be taken as an isolated node to estimate local dynamics $\boldsymbol{f}$. Here, we compare the return maps of a low-degree node and a hub node's dynamics with an isolated Rulkov map under the effect of weak coupling (Fig 3).

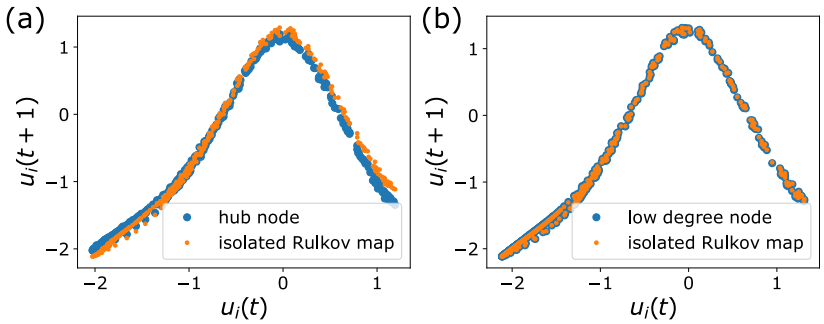

FIG. 3: a) The return map of the hub's data and isolated Rulkov map to show the weak coupling effect. b) The return map of one of the low-degree nodes shows that these nodes oscillate almost with the isolated local dynamics.

B. Hénon map

1. *u-component coupling*

We use Hénon map as $\boldsymbol{f}$ in Eq. (3), a discrete-time dynamical system exhibit chaotic behavior:

$$\begin{aligned} u_i(t+1) &= 1 - \alpha u_i(t)^2 + v_i(t) - \sum_{j=1}^{n} L_{ij} u_j(t) \\ v_i(t+1) &= \beta u_i(t) \end{aligned} \tag{6}$$

where $\alpha = 1.4$ and $\beta = 0.3$ are fixed during the simulations. Hénon maps are coupled through their u-components weakly. Initial positions are taken from the interval $[0, 0.1]$ randomly uniformly for each map. We iterate over 11000 steps and discard the first 10000 steps as a transient. Our procedure reveals the local dynamics as *Hénon map*, interaction function as *diffusive coupling* for $u-$component, and finally, the correct Laplacian matrix from time series data.

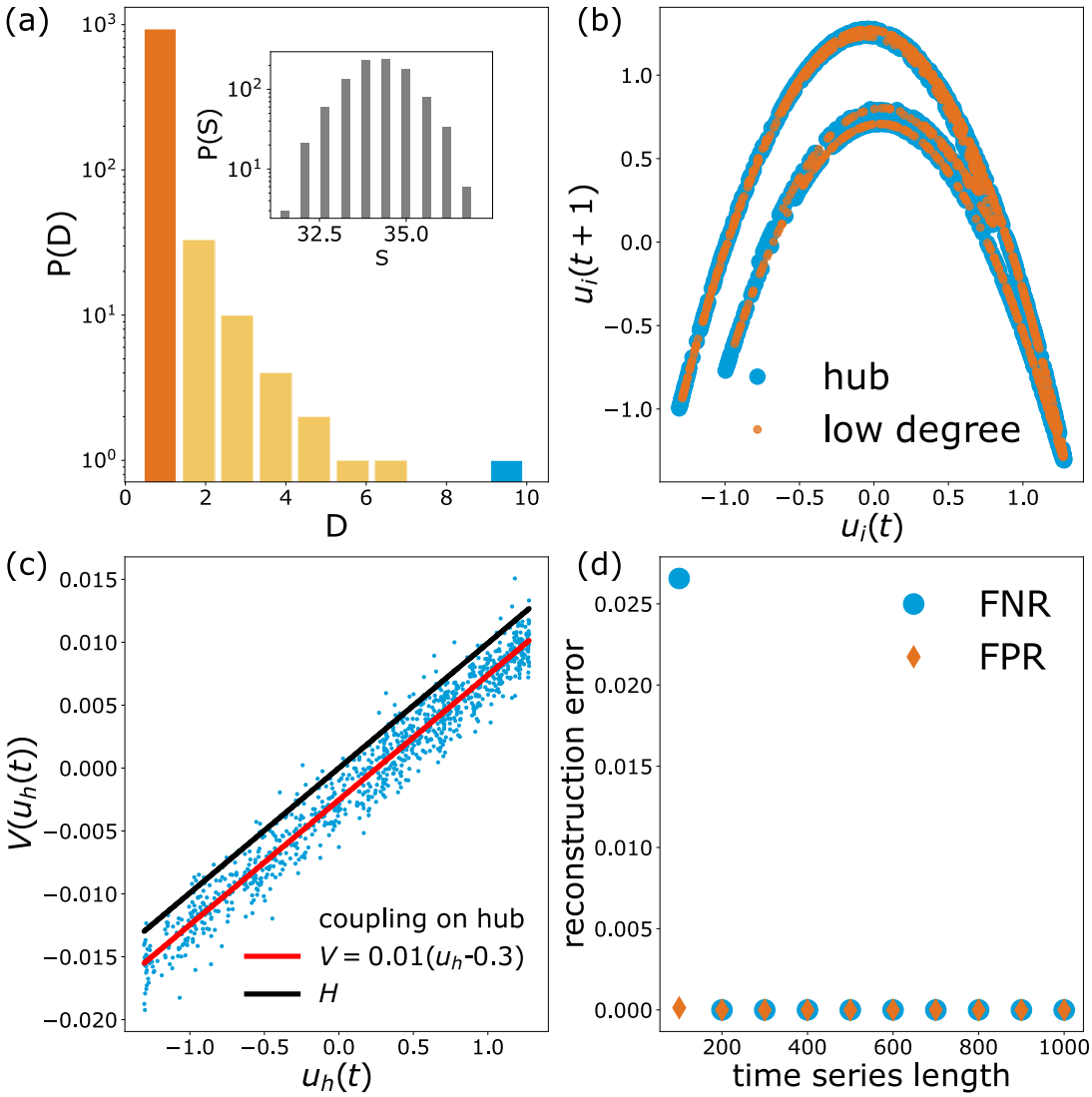

FIG. 4: Reconstruction procedure for weakly u-coupled Hénon maps on a real network. First, we learn a model for each node's time series data to classify them. (a) The histogram $P(D)$ based on the pairwise Euclidean distance matrix. While the highest bin of the histogram gives us $\boldsymbol{f}$, the lowest bin identifies the hub. The inset histogram emphasizes that the pairwise correlations of the time series observations do not provide any information about the network connectivity due to the chaotic dynamics. (b) The return maps of a low degree node and hub show the coupling effect slightly due to weak coupling. In (c), we plot the dominant coupling effect on the hub to see the model of $\boldsymbol{H}$. (d) shows FNR and FPR for different lengths of time series. 200 data points are enough for full reconstruction of the network dynamics with 987 nodes.

2. *u-component coupling with sinusoidal function*

We use the same $\boldsymbol{f}$ but different coupling function $\boldsymbol{H}$ in this example:

$$\begin{aligned} u_i(t+1) &= 1 - \alpha u_i(t)^2 + v_i(t) - \sum_{j=1}^{n} L_{ij} \sin(2\pi u_j(t)) \\ v_i(t+1) &= \beta u_i(t) \end{aligned} \tag{7}$$

Initial positions are taken from the interval $[0, 0.01]$ randomly uniformly for each map. We iterate over 12000 steps and discard the first 10000 steps as a transient. Fig. 5 shows the reconstruction results of sine-coupled Hénon maps. We simply added trigonometric functions to the candidate function library in the first step of our procedure and fully

reconstructed the network connections by learning the local dynamics as *Hénon map* and the coupling function as *sine function*.

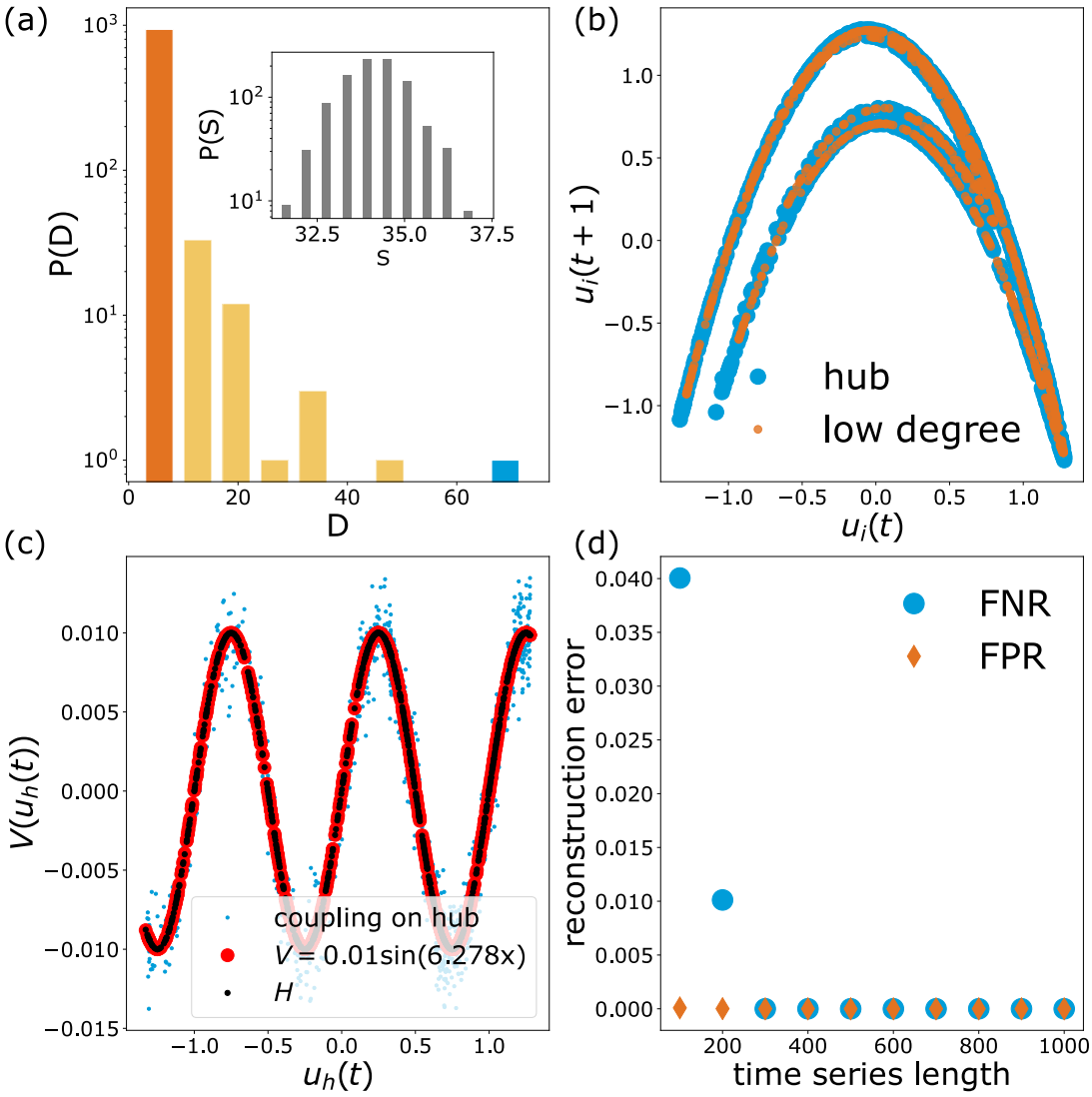

FIG. 5: Reconstruction procedure for weakly $\sin u$-coupled Hénon maps on a real network. First, we learn a model for each node's time series data to classify them. (a) The histogram $P(D)$ based on the pairwise Euclidean distance matrix. While the highest bin of the histogram gives us $\boldsymbol{f}$, the lowest bin identifies the hub. The inset histogram emphasizes that the pairwise correlations of the time series observations do not provide any information about the network connectivity due to the chaotic dynamics. (b) The return maps of a low degree node and hub show the coupling effect slightly due to weak coupling. In (c), we plot the the dominant coupling effect on the hub to see the model of $\boldsymbol{H}$. (d) shows FNR and FPR for different lengths of time series. 300 data points are enough for full reconstruction of the network dynamics with 987 nodes.

C. Tinkerbell Map

Tinkerbell map coupled through their u-component diffusively is given by:

$$\begin{aligned} u_i(t+1) &= u_i(t)^2 - v_i(t)^2 + au_i(t) + bv_i(t) - \sum_{j=1}^{n} L_{ij}u_j(t) \\ v_i(t+1) &= 2u_i(t)v_i(t) + cu_i(t) + dv_i(t) \end{aligned} \tag{8}$$

where $a = 0.9, b = -0.6013, c = 2.0$ and $d = 0.5$ are fixed during the simulations to ensure a fully chaotic regime. Data is generated from u-coupled Tinkerbell maps by iterating over 11000 steps and dropping the first 10000 time steps as a transient. We successfully reveal the local dynamics of the Tinkerbell map, interaction function and finally the Laplacian matrix from time series.

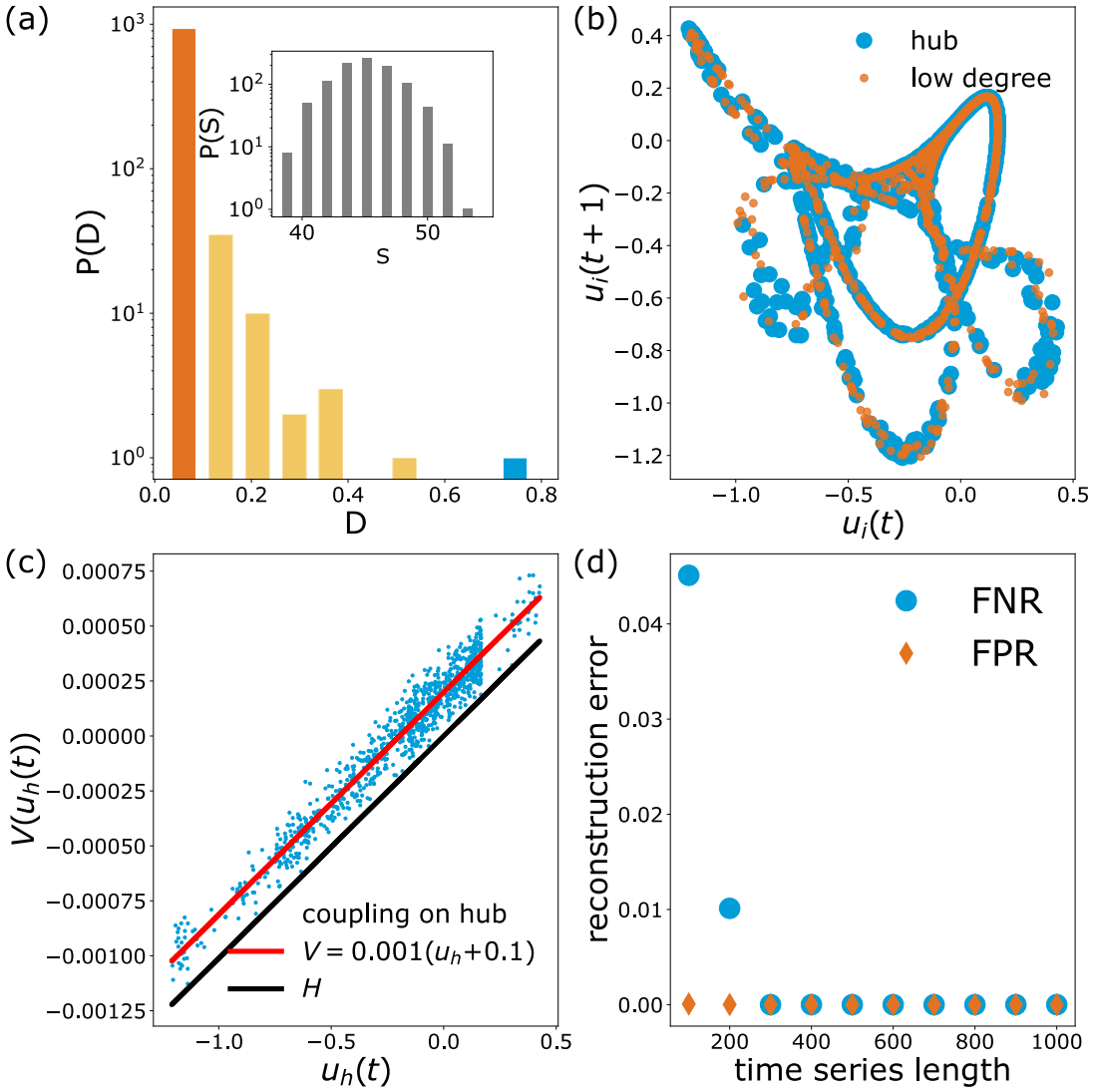

FIG. 6: Reconstruction procedure for weakly u-coupled Tinkerbell maps on a real network. First, we learn a model for each node's time series data to classify them. (a) The histogram $P(D)$ based on the pairwise Euclidean distance matrix. While the highest bin of the histogram gives us $\boldsymbol{f}$, the lowest bin identifies the hub. The inset histogram emphasizes that the pairwise correlations of the time series observations do not provide any information about the network connectivity due to the chaotic dynamics. (b) The return maps of a low degree node and hub show the coupling effect slightly due to weak coupling. In (c), we plot the dominant coupling effect on the hub to see the model of $\boldsymbol{H}$. (d) shows FNR and FPR for different lengths of time series. 300 data points are enough for full reconstruction of the network dynamics with 987 nodes.

As seen in figures 4(c), 5(c) and 6(c), the horizontal shift between the fitted curve on the data (red) and the predicted H curve (black) varies for each dynamical system. This observation is compatible with the reduction theorem, which is analytically proven only for expanding maps [4]. We show that even if the theorem is not rigorously proven for the Rulkov map, Hénon map or Tinkerbell map, the methodology accurately works. For the coupling, $H(x, y) = y - x$, $v(x) = \int H(x, y)dm(y)$ is found as $-x$ plus a constant. These constants are found as 1 for coupled Rulkov maps, -0.3 for coupled Hénon maps and 0.1 for coupled Tinkerbell maps. For sinusoidal coupling, $H(x, y) = \sin 2\pi y - \sin 2\pi x$, $v(x) = \int H(x, y)dm(y)$ is found as $-\sin 2\pi x$ plus a constant and it is 0 for sin-coupled Hénon maps.

VI. MACAQUE MONKEY VISUAL CORTEX NETWORK

We tried our procedure on a real monkey visual cortex network that is not a scale-free type [5, 6]. The network consists of 91 nodes and 581 undirected and unweighted edges, which is illustrated in Fig. 7(a) with its degree distribution in Fig. 7(b). It is impossible to classify nodes by degree in a network with such a degree distribution, failure to classify nodes results in an unable to learn local dynamics and identify the hub. So, we assume we know the local dynamics and which node is the hub. When we continue with this preliminary information, the reduction theorem allows us to learn the coupling function smoothly and see full reconstruction.

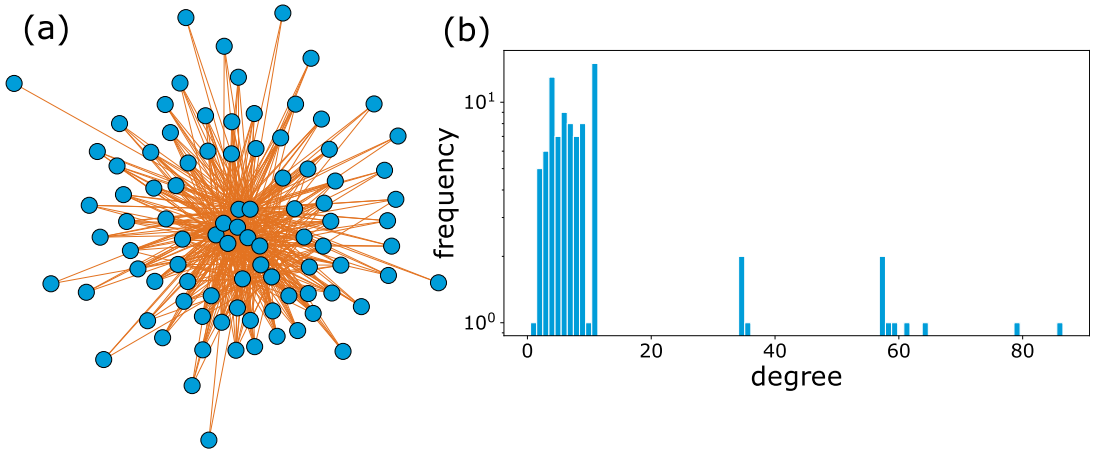

FIG. 7: (a) Undirected and unweighted macaque monkey visual-cortex network. (b) Degree distribution of the associated real network.

VII. SPARSE OPTIMIZATION

The model for the dynamical system is given by the function $\boldsymbol{g}$:

$$\boldsymbol{g}(\boldsymbol{x}) \approx \sum_{k=1}^{p} \psi_k(\boldsymbol{x})\xi^k \tag{9}$$

where $\boldsymbol{\Psi}(\boldsymbol{x}) = [\psi_1(\boldsymbol{x}), \psi_2(\boldsymbol{x}), ..., \psi_p(\boldsymbol{x})]$ is the library of basis functions. Only a few active terms characterize data that comes from natural systems on a well-chosen basis; most of the coefficients ξ^k are zero. To find interpretable models sparsity concept is beneficial. This idea was used in a compressed sensing framework for the first time as the sparsity requirement is satisfied [7]. If we have prior knowledge to decide the basis, sparse regression is instrumental in avoiding overfitting and noise [8]. In our reconstruction procedure, we use sparse regression in more than one step.

The least absolute shrinkage and selection operator (LASSO) [9] is a sparse regression method that uses ℓ_1-norm to promote sparsity. Sequentially-thresholded least-squares (STLS) is another method implemented in the Python package of SINDy [10]. In STLS, we start with an ordinary least squares solution which results in overfitting the time series at each point. We repeat least squares to the values obtained by thresholding the residues below a particular cut-off value called sparsity parameter at each sequence. Finally, an interpretable dynamical model is obtained.

VIII. LIBRARY OF BASIS FUNCTIONS

Although prior knowledge about the dynamical system is always helpful in constructing a basis library, it is impossible to access the information for all cases. Nevertheless, many functions can be accurately estimated using high-order polynomials. For instance, in this work, we mainly studied with the Rulkov maps defined as follows,

$$\begin{aligned} u(t+1) &= \frac{\beta}{1+u(t)^2} + v(t) \\ v(t+1) &= v(t) - \nu u(t) - \sigma \end{aligned} \tag{10}$$

Using a basis library containing only polynomials, it is possible to reconstruct an imperfect model for the Rulkov map, which generates quite an accurate time series to the original one (Fig. 8). For the Rulkov map, only the non-polynomial term is the rational one, $\frac{1}{1+u^2}$, which is absent in the library. However, this rational term can be expanded in a power series as follows,

$$\frac{1}{1+x^2} = \sum_{n=0}^{\infty}(-x^2)^n = 1 - x^2 + x^4 - x^6 + x^8 + \cdots \tag{11}$$

therefore, it is possible to model the Rulkov map with polynomials roughly. As the power expansion's convergent interval is $[-1, 1]$, the predicted model diverges when the orbit is out of the given interval. However, such convergence problems can be solved by normalizing the time series into the associated space. The interesting dynamics for our study is the bursting regime which can be modeled without the normalization Fig. 8 (b); therefore, we skip this step, but the model becomes very complicated with many higher-order polynomial terms.

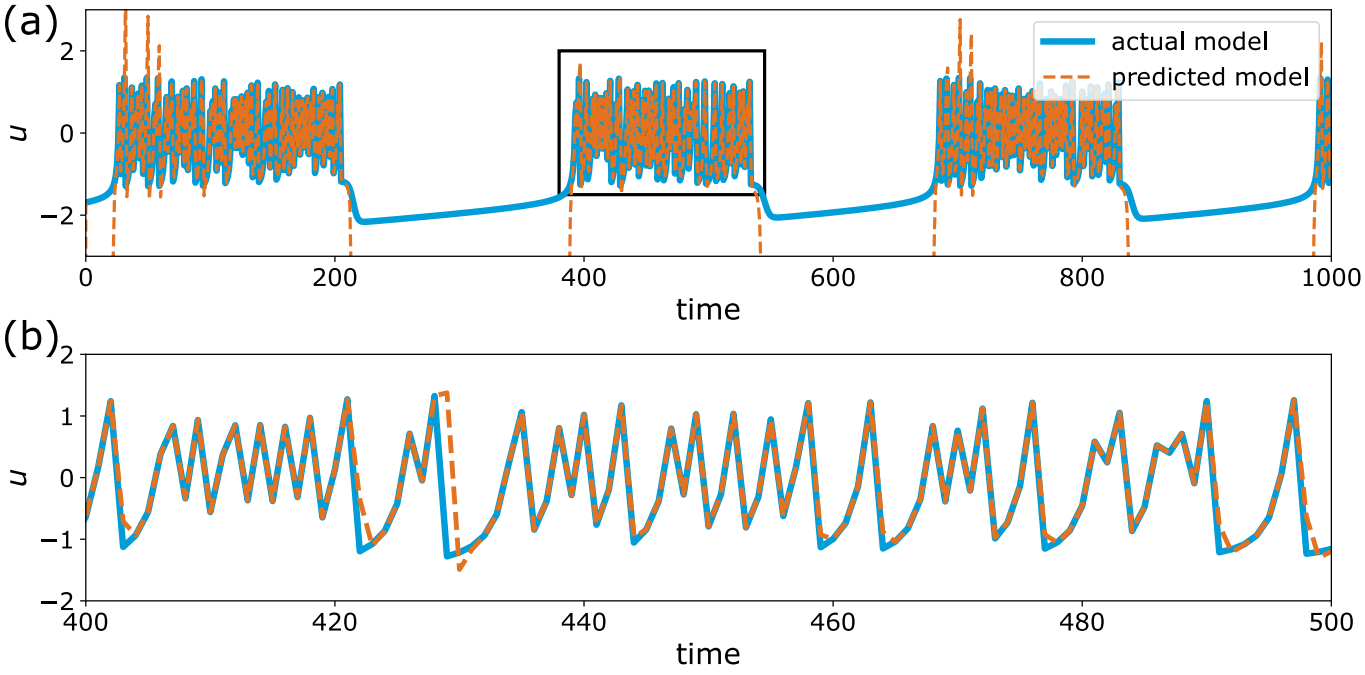

FIG. 8: The predicted time series of a reconstructed Rulkov model using a candidate functions library containing only polynomials. a) predicted u-variable of the model b) a bursting regime package zoomed from the black rectangle in a).

In order to avoid complicated models containing many polynomial terms and possible inaccurate predictions, we prepared a standard library including common functions, such as geometric series and trigonometric functions. Although unreasonably increasing the number of functions in the library is undesired, adding such nonpolynomial terms generally terminates a large number of polynomials. Thus, our computational cost mostly becomes cheaper using the following standard candidate library:

$$\boldsymbol{\Psi} = \left[\mathbf{1}, \mathscr{P}_d(\boldsymbol{x}), \left\{\sin(i\boldsymbol{x}), \cos(i\boldsymbol{x})\right\}_{i=1}^{r}, \left\{\frac{1}{\boldsymbol{x}^i}, \frac{1}{1 \pm \boldsymbol{x}^i}, \frac{1}{(1 \pm \boldsymbol{x})^i}\right\}_{i=1}^{q}\right]$$

where d is the degree of the polynomials, and for instance $\mathscr{P}_2(\boldsymbol{x})$ is:

$$\mathscr{P}_2(\boldsymbol{x}) = [u, v, u^2, uv, v^2].$$

Our main strategy is to start $d = r = q = 1$, and increase them slowly to estimate the model with the minimum number of candidate functions. The first step of our reconstruction algorithm aims to learn the local dynamics by classifying the inferred model equations due to their similarity. Here, we illustrate the reconstructed models on the mouse neocortex network using coupled Rulkov maps (Table I) through a set of selected nodes' with respect to their degrees k_i ($k_0 = 0$, $k_{46} = k_{136} = 4$, and $k_{218} = 26$). As node-0 has no incoming links, we reconstructed only the

Rulkov map dynamics. Node-46 has 4 incoming links, the inferred coefficients for it do not reflect the exact governing equation, but one can say that its governing equations are similar to node-136, which also has 4 incoming links. The most *distinct* model belongs to node-218, which is the hub of this particular network.

node id	1	u	v	$\cdots$	$\cos(u)$	$\frac{1}{1-u}$	$\frac{1}{1+u^2}$
0	-0.000	-0.000	1.000	$\vdots$	0.000	-0.000	4.100
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
46	0.000	-0.015	1.005	$\vdots$	0.005	0.000	4.093
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
136	-0.000	-0.015	1.006	$\vdots$	0.000	-0.000	4.101
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
218	-0.714	0.098	0.783	$\vdots$	0.016	0.001	4.075

TABLE I: Table for the inferred coefficients of the corresponding candidate functions for some of the selected nodes. We present only nonzero functions for any node.

The first step of the reconstruction algorithm is to classify nodes, which uses a distance between nodes' coefficients. We define the distance metric as, $d_{ij} = (\Sigma_{k=1}^{p} \frac{1}{V_k} |\xi_i^k - \xi_j^k|^2)^{1/2}$ where $|\cdot|$ is absolute value, V_k is the variance of the predicted coefficients of the k-th function in $\hat{\boldsymbol{\Psi}}$. As an example, we compute the pairwise distances of the learnt models given in Table I:

node id	0	46	136	218
0	0.000	2.830	2.105	18.351
46	2.830	0.000	1.925	16.808
136	2.105	1.925	0.000	16.979
218	18.351	16.808	16.979	0.000

TABLE II: Pairwise distances of the predicted models for some of the selected nodes.

As numerically shown in Table II, d_{ij} is small for the nodes with the same or similar degrees, such as the distance is 1.925 between 46 ($k_{46} = 4$) and 136 ($k_{136} = 4$), while the distance is large for different predicted models, such as the distance is 18.351 between 46 ($k_{46} = 4$) and 218 ($k_{218} = 26$).

IX. COMPARISON BETWEEN SPARSE REGRESSION METHODS

We compare two optimizers to perform sparse regression; LASSO and STLS.

LASSO is defined by the equation [11]:

$$\hat{\boldsymbol{w}}_{LASSO}(\lambda) = \underset{\boldsymbol{w}\in\mathbb{R}^m}{\operatorname{argmin}}(||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_1) \tag{12}$$

for a linear problem $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w}$ where $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ and λ is *penalty term* and uses ℓ_1-norm to penalize the weights.

In STLS, Ridge regression is used and defined by the equation [11]:

$$\hat{\boldsymbol{w}}_{Ridge}(\lambda) = \underset{\boldsymbol{w}\in\mathbb{R}^m}{\operatorname{argmin}}(||\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_2^2) \tag{13}$$

and λ is fixed as 0.05. Sparsity is obtained by masking out elements of $\hat{\boldsymbol{w}}$ that are below a given threshold. This threshold is called as *sparsity parameter* and it is the single hyper-parameter of STLS. In the final step of our reconstruction approach, we find a sparse solution for $\boldsymbol{w}$, it corresponds to $\boldsymbol{L}$ in our case. From the algorithmic point of view, fine-tuning of the hyper-parameters, penalty term for LASSO and sparsity parameter for STLS, is essential to improve the accuracy. These two parameters are not equivalent, however we can make a similar interpretation between them. The length of the time series determines the problem type since our network size is fixed in the experiments. The time series length lower than the network size ($T < nm = 987 \times 2 = 1974$) correspond to an under-determined case for the real network we use.

Fig. 9 presents the reconstruction performance for different time series lengths and two hyperparameters as FNR and FPR. We compare FNR values to evaluate the performance of LASSO and STLS in Fig. 9(a) and Fig. 9(c), since FPR values are always very small. The results show that LASSO overcomes STLS for shorter time series. Since the ℓ_1-norm penalized solution, is a generalization of compressive sensing approach, it provides a unique solution for underdetermined cases [12]. On the other hand, if there is long enough data, STLS becomes a fast alternative to the LASSO from an algorithmic point of view and works very well for our problem. We took one under-determined case ($T = 300$) to compare the performance of two different sparsity promoting methods in Fig. 10. If we have limited data, we see full reconstruction by LASSO but not by STLS.

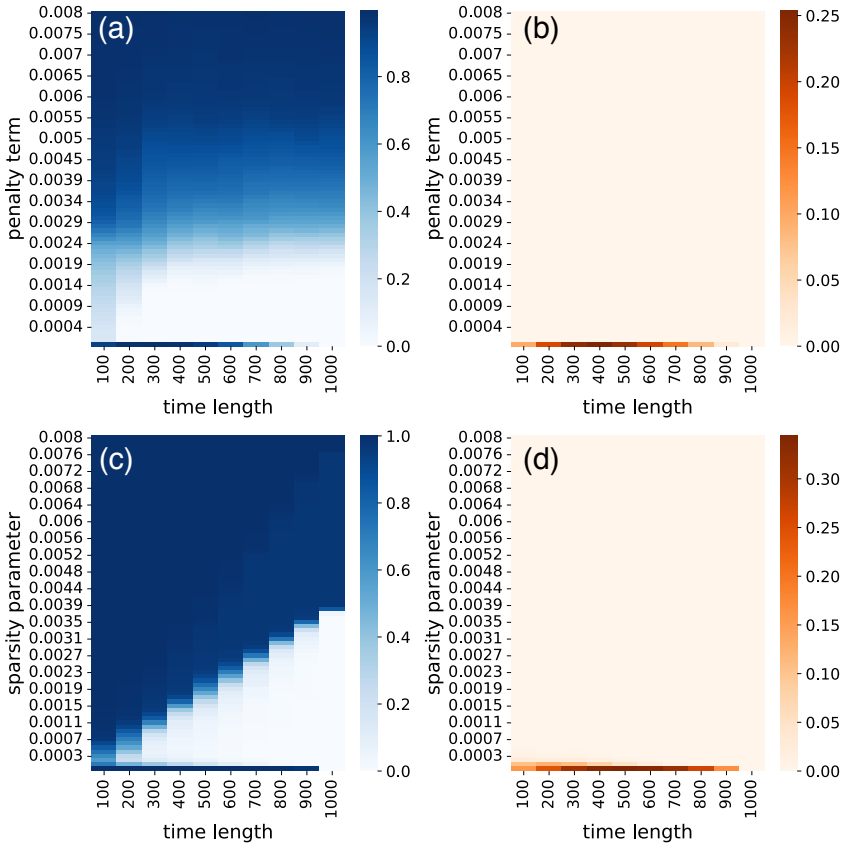

FIG. 9: (a) FNR and (b) FPR with respect to a list of penalty terms for LASSO. (c) FNR and (d) FPR with respect to a list of sparsity parameters for STLS. 10 different lengths of time series are used to compare the performance.

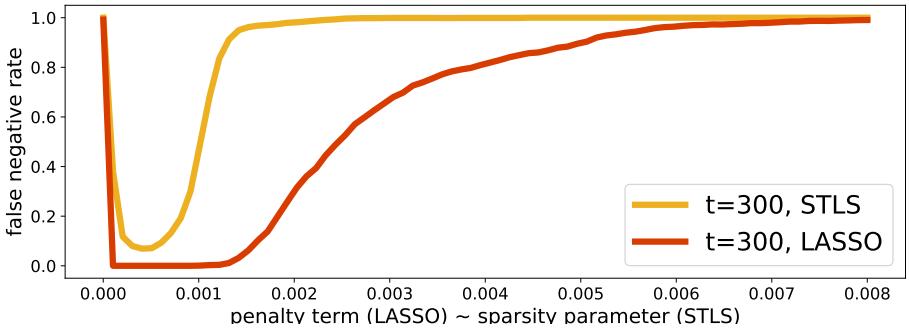

FIG. 10: FNR with respect to a list of penalty term for Lasso and sparsity parameter for STLS. ($T = 300$) case is used as a short time series example. There is no *full reconstruction* region for STLS.

X. EFFECT OF NETWORK SPARSITY ON RECONSTRUCTION PERFORMANCE

In addition to the lack of sufficient data, the sparsity condition of the Laplacian matrix should be satisfied to solve the linear regression problem within the compressed sensing framework. We present results in this direction based on synthetical networks. The algorithm we use to generate directed scale-free networks has three probabilistic parameters: α and γ are the probabilities of adding a new node connected to an existing node chosen randomly according to the in-degree and out-degree distribution, respectively. β is the probability of adding an edge between two existing nodes. We use two different parameter setting as $[\alpha = 0.41, \gamma = 0.05, \beta = 0.54]$ and $[\alpha = 0.2, \gamma = 0.5, \beta = 0.3]$ to obtain Laplacian matrices at different sparsity levels. The former (default) setting's power exponent is approximately 2 and generates denser networks, while the latter's power exponent is approximately 1 and generates sparser networks. As seen in Fig. 11, we see full reconstruction in sparser networks since the Laplacian matrix meets the sparsity condition for the compressive sensing approach [13].

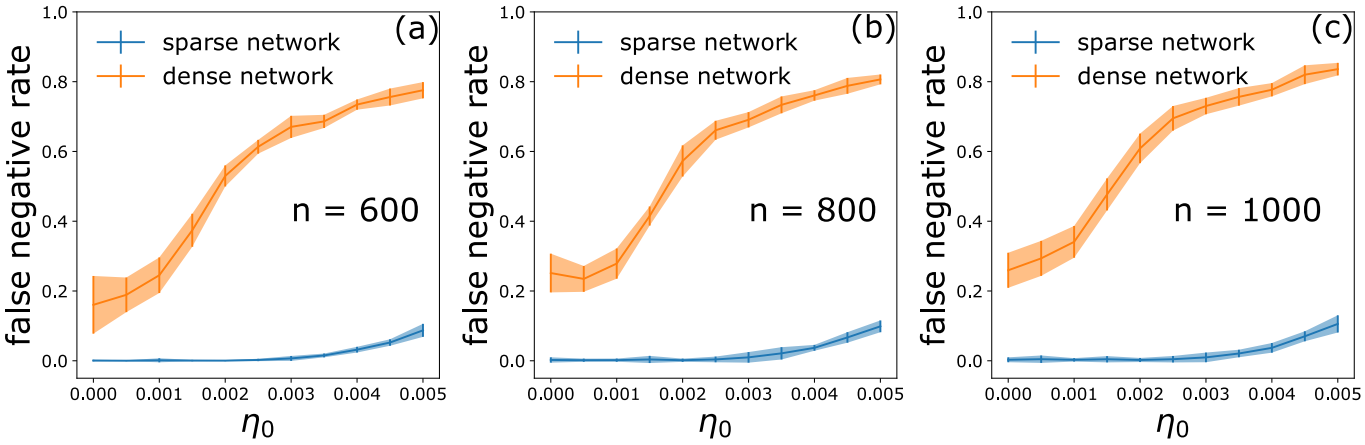

FIG. 11: FNR against noise for sparse and dense networks of sizes (a) $n = 600$, (b) $n = 800$ and (c) $n = 1000$. All points show an average error over 10 different realizations of scale free networks and shaded regions present standard deviations. Time series length is 500, which corresponds to underdetermined cases for all three network sizes.

XI. CROSS VALIDATION OF INFERRED LAPLACIAN MATRICES

In the main text, we use the ground-truth Laplacian matrix to show the performance of the reconstruction for various hyper-parameters and data lengths. However, it is unlikely to know the true matrix to evaluate the methodology's success in real-world problems. Therefore, it is important to assess the reconstruction outcome when the real matrix is absent using a cross-validation technique. To illustrate the cross-validation approach, we obtained *predicted time series* of length 500 using inferred Laplacian matrices for various hyper-parameters (penalty term) $\lambda = 0.008, 0.003$ and 0.0003. Out of those three parameters, the outcome for $\lambda = 0.0003$ gives the best-fit, Fig. 12(a).

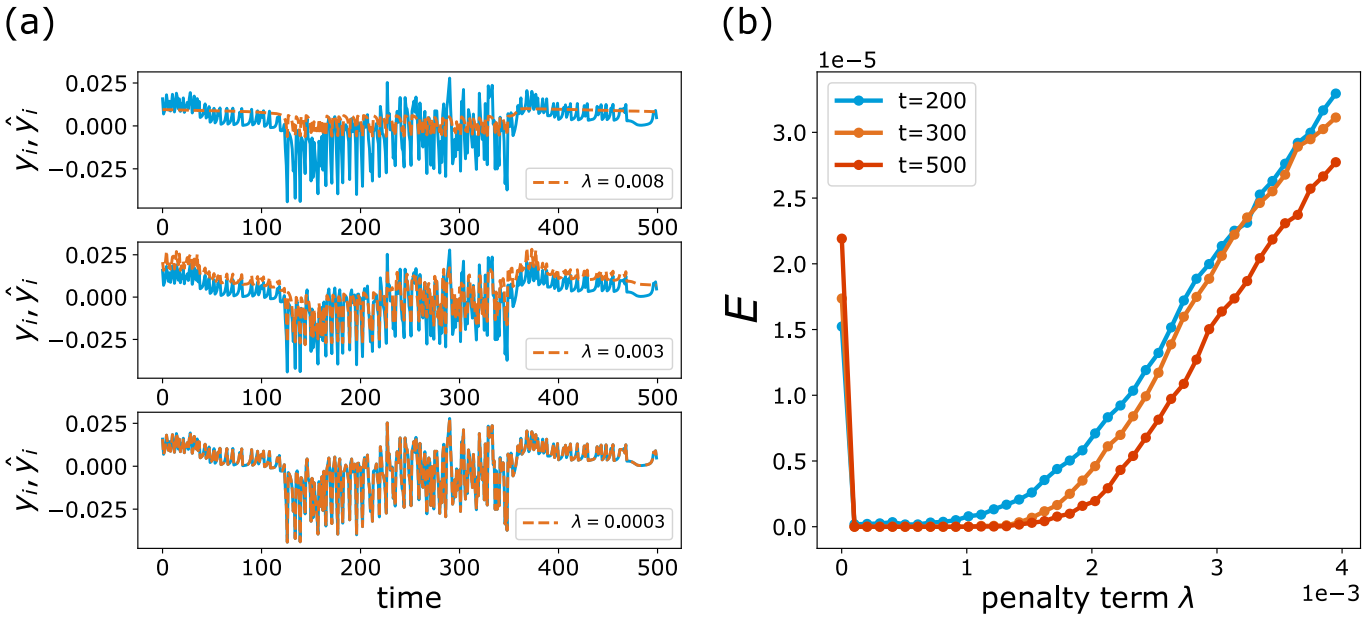

FIG. 12: a) Comparison between measurement of a node and test time series. Each λ parameter corresponds to different connectivity matrices. We simulate the inferred models and obtain test time series. b) Cross validation of the inferred time series based on obtained connectivity matrices for a series of penalty terms.

To find the best hyper-parameters in large network dynamics, we use an error function

$$E = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{T}\sum_{t=1}^{T}\left(y_{i,t} - \hat{y}_{i,t}\right)^2\right) \tag{14}$$

where $y_i = x_i(t+1) - f(x_i)$ and $\hat{y}$ denotes the predicted one. Then we optimize the hyper-parameters, which minimize the error. The parameters yielding the most accurate results can be used as the optimal hyper-parameter (Fig. 12(b)).

XII. WHY IS THE REDUCTION THEOREM CRUCIAL TO RECONSTRUCT LARGE NETWORKS?

A fully-algorithmic approach is to apply any sparse optimization technique directly to the entire multi-variable time series. Only up to a certain size networks can be recovered by this method. By the reduction theorem, we learn local dynamics from low degree nodes and interaction dynamics from the hub. We are not looking for a sparse linear combination of functions within a massive library of all possible functions. Therefore, there is no limit on network size in our approach. In this section, we will show this with a simple example:

Assume we have a network with 5 nodes. If node 1 has links coming from the nodes 0, 4 and 5 as seen in Fig. 13, then the dynamics of node 1 is written simply in Laplacian matrix form:

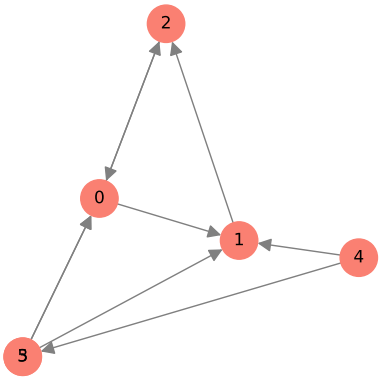

FIG. 13: 5-nodes network as an example.

$$\boldsymbol{x}_1(t+1) = \boldsymbol{f}(\boldsymbol{x}_1(t)) + 3\boldsymbol{H}(\boldsymbol{x}_1(t)) - \boldsymbol{H}(\boldsymbol{x}_0(t)) - \boldsymbol{H}(\boldsymbol{x}_4(t)) - \boldsymbol{H}(\boldsymbol{x}_5(t)) \tag{15}$$

where $\boldsymbol{x}_i \in \mathbb{R}^m$. Eq.(15) is in the form of a linear combination of the connections, allowing any sparse regression algorithm to capture the parameters correctly in case of a well-defined basis library. For one single equation, it seems pretty feasible, but for a 5-nodes network, each has m components, we have $5m$ equations to be discovered. Suppose we set the basis composed of polynomials up to second-degree and all quadratic combinations of all variables due to a piece of prior knowledge about the coupling function. This example setting has 9 functions for $m = 3$. In that case, our basis includes 45 features for this particular example. If we do not have prior knowledge, we use an extensive library that includes more candidate functions. We performed experiments in this direction to see the limitations. The results showed that a purely algorithmic equation-based learning approach fails as the network size, that is, the basis grows. Results of Novaes *et al.* validated our conclusions, they showed that adding new functions to the basis due to increase of network size (up to 20-nodes) effects badly the reconstruction [14].
Furthermore, in the final sparse regression step, our problem turns into a linear equation as $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{y} \in \mathbb{R}^T$, $\boldsymbol{A} \in \mathbb{R}^{T\times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$. Here, $\boldsymbol{y}$ is the data, and the length of the data is T, n is the network size, and we focus on the underdetermined cases ($T < n$) since data availability is always limited. From the perspective of compressed sensing, the measurement matrix $\boldsymbol{A}$ should satisfy some conditions to converge the sparsest solution for $\boldsymbol{x}$ by ℓ_1-norm. One of these conditions can be summarized as the time series length scales with the network size, so if the measurement matrix has more columns, longer measurements are needed. It is given by the relation:

$$T \approx \mathscr{O}(Klog(n/K))$$

where K is the network's sparsity measure (number of nonzero entries) [15]. If the number of columns is determined by the basis functions that include all possible pairwise interactions, not the network size, this means the need for a longer time series. On the other hand, the uncertainty of the interaction function is an obstacle to inferring the exact connection matrix [16]. Finding $\boldsymbol{H}$ and therefore reducing the dimension of the optimization problem for full reconstruction broadens the applicability and distinguishes our work.

XIII. GENERAL DIFFUSIVE COUPLING AND MASTER STABILITY FUNCTION

Consider the function $\boldsymbol{h} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$. We say that $\boldsymbol{h}$ is diffusive if

$$\boldsymbol{h}(x, x) = \boldsymbol{h}(0) = 0 \text{ and } \boldsymbol{h}(x, y) = -\boldsymbol{h}(y, x)$$

We again consider the model to a general diffusive coupling

$$\boldsymbol{x}_i(t+1) = \boldsymbol{f}(\boldsymbol{x}_i(t)) + \gamma \sum_{j=1}^{n} w_{ij}\boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{x}_i) \tag{16}$$

where γ is the coupling strength.

We perform the analysis close to synchronization $\boldsymbol{x}_i = \boldsymbol{s} + \boldsymbol{\xi}_i$ so

$$\boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{x}_i) = \boldsymbol{h}(\boldsymbol{s} + \boldsymbol{\xi}_j, \boldsymbol{s} + \boldsymbol{\xi}_i) = \boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s}) + D_1\boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s})\boldsymbol{\xi}_j + D_2\boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s})\boldsymbol{\xi}_i$$

but since the coupling is diffusive

$$D_2\boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s}) = -D_1\boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s})$$

we get

$$\boldsymbol{h}(\boldsymbol{x}_j, \boldsymbol{x}_i) = \boldsymbol{H}(\boldsymbol{s})(\boldsymbol{\xi}_j - \boldsymbol{\xi}_i) + \boldsymbol{R}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)$$

where $\boldsymbol{H}(\boldsymbol{s}) = D_1\boldsymbol{h}(\boldsymbol{s}, \boldsymbol{s})$ and $\boldsymbol{R}(\boldsymbol{\xi}_i, \boldsymbol{\xi}_j)$ contains quadratic terms. Then, the first variational equation about the synchronization manifold

$$\boldsymbol{\xi}_i(t+1) = D\boldsymbol{f}(\boldsymbol{s}(t))\boldsymbol{\xi}_i(t) + \gamma \sum_{j=1}^{n} w_{ij}\boldsymbol{H}(\boldsymbol{s})(\boldsymbol{\xi}_j(t) - \boldsymbol{\xi}_i(t)) \tag{17}$$

$$= D\boldsymbol{f}(\boldsymbol{s}(t))\boldsymbol{\xi}_i(t) - \gamma\boldsymbol{H}(\boldsymbol{s}) \sum_{j=1}^{n} L_{ij}\boldsymbol{\xi}_j(t). \tag{18}$$

All blocks have the same form which are different only by λ_i, the ith eigenvalue of $\boldsymbol{L}$. Then we obtain the parametric equation for the modes

$$\boldsymbol{u}(t+1) = [D\boldsymbol{f}(\boldsymbol{s}(t)) - \kappa\boldsymbol{H}(\boldsymbol{s}(t))]\boldsymbol{u}(t)$$

where $\kappa = \gamma\lambda_i$. By fixing κ, we compute the maximum Lyapunov exponent $\Lambda(\kappa)$ as

$$\|\boldsymbol{u}(t)\| \leq Ce^{\Lambda(\kappa)t}.$$

The map

$$\kappa \mapsto \Lambda(\kappa) \tag{19}$$

is called master stability function. Notice that if $\Lambda(\kappa) < 0$ when $\kappa > \gamma_c\lambda_2$ then $\|u\| \to 0$. For more details on the master stability function Eq. 19, see e.g., [17, 18].

[1] B. Barzel and A. L. Barabási, Nature Physics **9**, 673 (2013), ISSN 17452481.

[2] T. Pereira, D. Eroglu, G. B. Bagci, U. Tirnakli, and H. J. Jensen, Phys. Rev. Lett. **110**, 234103 (2013).

[3] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, and T. R. J. *et al.*, Cell **162**, 648 (2015), ISSN 10974172.

[4] D. Eroglu, M. Tanzi, S. van Strien, and T. Pereira, Physical Review X **10**, 1 (2020), ISSN 2160-3308.

[5] N. T. Markov, J. Vezoli, P. Chameau, A. Falchier, R. Quilodran, C. Huissoud, C. Lamy, P. Misery, P. Giroud, S. Ullman, et al., Journal of Comparative Neurology **522**, 225 (2014).

[6] J. T. Vogelstein, E. Perlman, B. Falk, A. Baden, W. G. Roncal, V. Chandrashekhar, F. Collman, S. Seshamani, J. L. Patsolic, and K. L. *et al.*, Nature Methods **15**, 846 (2018), ISSN 15487105.

[7] W. X. Wang, R. Yang, Y. C. Lai, V. Kovanis, and C. Grebogi, Physical Review Letters **106** (2011), ISSN 00319007.

[8] S. L. Brunton, J. L. Proctor, J. N. Kutz, and W. Bialek, Proceedings of the National Academy of Sciences **113**, 3932 (2016), ISSN 10916490.

[9] R. Tibshirani, Source: Journal of the Royal Statistical Society. Series B (Methodological) **58**, 267 (1996).

[10] B. M. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, Journal of Open Source Software **5**, 2104 (2020), URL `https://doi.org/10.21105/joss.02104`.

[11] P. Mehta, M. Bukov, C. H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Physics Reports **810**, 1 (2019), ISSN 03701573, URL `https://doi.org/10.1016/j.physrep.2019.03.001`.

[12] D. Donoho, IEEE Transactions on Information Theory **52**, 1289 (2006).

[13] E. J. Candes, J. K. Romberg, and T. Tao, Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences **59**, 1207 (2006).

[14] M. Novaes, E. R. dos Santos, and T. Pereira, Physica D: Nonlinear Phenomena **424**, 132895 (2021), ISSN 01672789.

[15] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2019).
[16] Y. Y. Liu and A. L. Barabási, Reviews of Modern Physics **88**, 035006 (2016), ISSN 15390756.
[17] L. Huang, Q. Chen, Y.-C. Lai, and L. M. Pecora, Phys. Rev. E **80**, 036204 (2009), URL `https://link.aps.org/doi/10.1103/PhysRevE.80.036204`.
[18] P. Schultz, T. Peron, D. Eroglu, T. Stemler, G. M. Ramírez Ávila, F. A. Rodrigues, and J. Kurths, Phys. Rev. E **93**, 062211 (2016).