RECONSTRUCTION OF NETWORKS OF CHAOTIC SYSTEMS

MEHMET KIRTIŞOĞLU

Abstract. In some areas, recovering the model of dynamics is important to predict the future behavior of them that most likely to happen whereas, in some other areas, it conducts us to the main part of a specific problem. However, it is generally difficult to control and predict complex networks with chaotic elements since a little change in a small part of the network may result in big differences. Also, the phenomenon of generalized synchronization leads to the synchronization of non-aligned identical systems, so keeping track of the similarity of the data will not be solely enough to handle the problem. Pinning one of the synchronized oscillators will help us distinguish them and better understand who is driven by whom. However, it may result in damaging the general system if an unnecessary number of external signal is implemented to the system. Hence, it is essential to successfully reconstruct the network with minimal number of external effect. In this project, networks are created as random Barabasi-Albert models. That requires us to develop some probabilistic methods to find the minimal number of attractors to pin and discover on which attractors pinning will be used with the least external effect on the system. Although it seems hard to find a mathematical analysis method to get an insight understanding, many results might be uncovered by observing and storing a great range of simulation outcomes.

1. Introduction

The current state of technology makes it possible to probe an enormous amount of data from a complex system. These advances have led to a better experimental understanding of transitions to seizures which can be described by a generic phenomenological mathematical model [6], correctly forecast monsoon duration for some anomalous years [13], and connectivity of the neurons in the brain, among many others. However, having this data on its own is not enough. We need to build the accurate model to use this data in order to predict and control the behavior of such systems, in particular catching critical transitions both from a numerical and a theoretical perspective. Recovering an accurate model from data is a notoriously challenging problem due to the huge number of elements involved and the intricacy of their interactions. The paramount importance of this issue has granted it a lot of attention. In this project, we address the problem of reconstructing the structure and type of interactions in complex systems from observations of the dynamics at each site. Even if this problem is ill-posed, for instance, if the interaction is strong and some oscillators are identical in the system, the network parts can transition to different synchronization schemes. In this case, it is impossible to recover the model from data. We aim to show that a lot of information can be recovered by blending techniques from machine learning and dynamical systems such as pinning theory and synchronization on complex graphs.

We consider our network dynamics as follows, the intrinsic dynamics is chaotic, and the interaction structure is heterogeneous. Furthermore, some oscillators can be in synchrony which is an additional challenge since it results in linear dependence of the collected data, which beclouds the identification of dynamics in a network. At this point, the pinning theory takes the role of breaking the synchronous behavior of oscillators for a short period, or in other words, transient dynamics. However, pinning, injecting an external signal, many vertices in a network can cause big troubles such as damaging the general system. In this project, we aim to find an analytical solution for the minimum number of interventions to reconstruct the dynamics of diffusively coupled Rössler [10] and Lorenz [14] *chaotic* systems on randomly generated directed Barabasi-Albert networks where each vertex comes with exactly one edge,

$$\frac{dx_i}{dt} = f_i(x_i) + \alpha \sum_{j=0}^{n} A_{ij}(x_j - x_i) + \delta_i, \tag{1}$$

where $A = (A_{iJ}) \in \mathbb{M}_{n\times n}[\mathbb{F}_2]$ is the adjacency matrix of our Barabasi-Albert network, $\alpha \in \mathbb{R}$ is the coupling strength, and $\delta = (\delta_1, \ldots, \delta_n)$ is the given external input. Let's first study on the conditions of synchronization of non-linear dynamics.

2. Synchronization of Non-linear Chaotic Dynamics

2.1. **Synchronization of Directly Coupled Identical Systems (Complete Synchronization).**

Let $\dot{x}$ and $\dot{y}$ denote two fully diffusively coupled identical Lorenz systems.

$$\dot{x} = \begin{pmatrix} \dot{x_0} \\ \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} \sigma(x_1 - x_0) \\ x_0(\rho - x_2) - x_1 \\ -\beta x_2 + x_0 x_1 \end{pmatrix} + \alpha(y - x), \tag{2}$$

$$\dot{y} = \begin{pmatrix} \dot{y_0} \\ \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} \sigma(y_1 - y_0) \\ y_0(\rho - y_2) - y_1 \\ -\beta y_2 + y_0 y_1 \end{pmatrix} + \alpha(x - y), \tag{3}$$

where $\sigma, \rho, \beta \in \mathbb{R}$ are parameters and $\alpha \in \mathbb{R}$ is the coupling strength. Our aim is to show that if the coupling strength α is sufficiently strong, then the diffusively coupled Lorenz systems get synchronized, i.e. $||x(t) - y(t)|| \to 0$ as $t \to \infty$. Let's define $z = x - y$,so $\dot{z}$ is given as:

$$\dot{z} = \dot{x} - \dot{y} = f(x) - f(y) - 2\alpha z. \tag{4}$$

To find the sufficient condition for the coupling strength, α, we linearize the equation of $\dot{z}$ near $z = x - y = 0$. By Taylor expansion we get

$$f(y) = f(x) - Df(x)(y - x) + \mathcal{O}(||z||^2), \tag{5}$$

where $Df(x)$ is the Jacobian matrix of f near $x(t)$. Now we plug this equation into $\dot{z}$ and get

$$\frac{dz}{dt} = [Df(x) - 2\alpha \mathrm{I}_{3\times 3}]z + \mathcal{O}(||z||^2). \tag{6}$$

Equation (6) is known as the *first variational equation* after ignoring the term $\mathcal{O}(|z|)^2$ [12]. To conclude our analysis we define a new variable ω:

$$\omega(t) = e^{2\alpha t} z(t). \tag{7}$$

Then by differentiating ω with respect to t we get:

$$\dot{\omega}(t) = 2\alpha e^{2\alpha t} z(t) + e^{2\alpha t} \dot{z}(t) \tag{8}$$
$$= 2\alpha\omega + [Df(x) - 2\alpha \mathrm{I}] e^{2\alpha t} z \tag{9}$$
$$= [Df(x)]\omega. \tag{10}$$

Now suppose $\Psi(x)$ is the fundamental matrix for the variational equation, so that any solution of this non-autonomous equation can be written as $z(t) = \Psi(x(t))z(0)$. Let's define $\{\lambda_i(x(t))_{i=1}^n\}$ as the set of positive square roots of the eigenvalues of the symmetric matrix $\Psi(x_1(t))^T \Psi(x(t))$. Now we define

$$\Lambda = \max_i \lim_{t\to\infty} \frac{1}{t} \lambda_i(x(t)). \tag{11}$$

Λ is known as the maximum *Lyapunov exponent* of the orbit $x(t)$ [15]. The conditions for the existence of the limit are given by the Oseledec theorem in 1968 [11].

Remark 1. *Lyapunov exponent measures how quickly an infinitesimally small distance $\delta(0)$ between two initially close states grows over flow of time F:*

$$||\delta(t)|| = ||F(x_0 + \delta(0), t) - F(x_0, t)_{||}$$
$$\approx ||\delta(0)||e^{\lambda t}$$

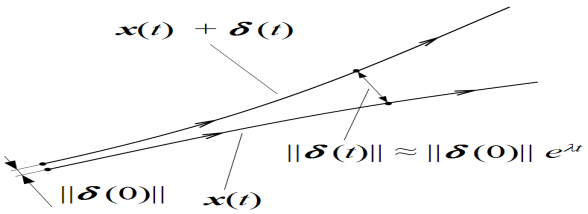


FIGURE 1. Lyapunov Exponent[5]

If the orbit $x(t)$ has maximum Lyapunov exponent Λ, then $\exists C > 0$ such that

$$||w(t)|| \leq Ce^{\Lambda t} \tag{12}$$
$$\implies ||z(t)|| \leq Ce^{(-2\alpha)t} \tag{13}$$
$$\implies \alpha_c = \frac{\Lambda}{2}. \tag{14}$$

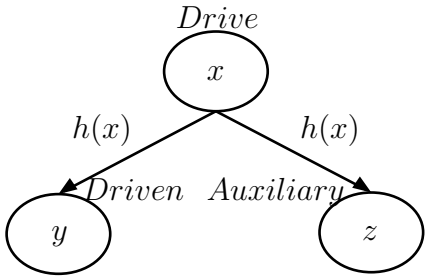
We say α_c is the critical coupling strength for observation of synchronization. It is computed that Lorenz Equation (for $\sigma = 10,\ \rho = 28,\ \beta = 8/3$) has Lyapunov exponent $\Lambda \approx 0.906$, so the critical coupling strength is $\alpha_c \approx 0.453$ [7].

2.2. **Synchronization of Non-directly Coupled Identical Chaotic Systems (Generalized Synchronization).**

Generalized synchronization is a phenomenon that occurs between unidirectionally coupled non-identical dynamical systems. We will consider this drive-driven configuration as follows:

$$\dot{x} = f(x), \tag{15}$$
$$\dot{y} = g(y, h(x)), \tag{16}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$, $g : \mathbb{R}^m \to \mathbb{R}^m$ are nonlinear functions $x \in \mathbb{R}^n, y \in \mathbb{R}^m$, and $h : \mathbb{R}^n \to \mathbb{R}^m$ is a function of x seen as driving the response system (slave system), y.



To detect Generalized Synchronisation between the two systems x and y we add an auxiliary system z and investigate complete synchronization between y and z. In case we observe complete synchronization between the identical copies, then generalized synchronization is observed between the drive x and driven systems y, and z.

More formally, L. Kocarev and U. Parlitz define generalized synchronization between x and y if there exists a transformation $H : \mathbb{R}^n \to \mathbb{R}^m$, a manifold $M = \{(x, y) : y = H(x)\}$, and a subset $B = B_x \times B_y \subset \mathbb{R}^n \times \mathbb{R}^m$ with $M \subset B$ such that all trajectories of (16) with initial conditions in the basin B approach M as time goes to infinity[9].

Theorem 1 (L. Kocarev, U. Parlitz [9])**.** *Generalized Synchronization occurs in system (16) if and only if $\forall(x_0, y_0) \in B$, the basin of attraction, the driven system $\dot{y} = g(y, h(x))$ is asymptotically stable meaning that any two different trajectories of y will synchronize when they are driven by the same trajectory of x.*

Proof. Let $\phi_x^t : \mathbb{R}^n \to \mathbb{R}^n$ be the flow of the system $\dot{x} = f(x)$ and $\phi^t = (\phi_x^t, \phi_y^t)$ the flow of (16) with $\phi_y^t : \mathbb{R}^{n+m} \to \mathbb{R}^m$. In order to construct the map H explicitly we choose an arbitrary point $x_0 \in B_x$ and determine the corresponding image point $y_0 = H(x_0)$. Since all states $y \in B_y$ of the response system converge only asymptotically to the manifold M we consider trajectories starting in the past at the point $(\phi^{-t}(x_0), y_0)$. When this trajectory passes the point x_0 the time t has elapsed and the point $(x_0, \phi^t(y_0))$ is the closer to M the larger t is. Formally we define $\tilde{H}(x_0, y_0) = \lim_{t\to\infty} \phi_y^t(\phi_x^{-t}(x_0), y_0)$. Asymptotic stability implies $\lim_{t\to\infty} ||\phi_y^t(\phi_x^{-t}(x_0), y_{10}) - \phi_y^t(\phi_x^{-t}(x_0), y_{20})|| \to 0$ for all $y_{10}, y_{20} \in B_y$, and therefore $\tilde{H}(x_0, y_0)$ is independent of y_0. The transformation H defining the synchronization manifold M is thus given by $H(x_0) = \tilde{H}(x_0, y_0)$ for arbitrary $y_0 \in B_y$. Furthermore, asymptotic stability implies that M is an attracting manifold [9]. □

Now let y and z denote two identical Rössler systems driven by a Lorenz system, x. Then,

$$\dot{x} = \begin{pmatrix} \dot{x_0} \\ \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} \sigma(x_1 - x_0) \\ x_0(\rho - x_2) - x_1 \\ -\beta x_2 + x_0 x_1 \end{pmatrix}, \tag{17}$$

$$\dot{y} = \begin{pmatrix} \dot{y_0} \\ \dot{y_1} \\ \dot{y_2} \end{pmatrix} = \begin{pmatrix} -(y_1 + y_2) \\ y_0 + a y_1 \\ b + y_2(y_0 - c) \end{pmatrix} + \alpha(x - y), \tag{18}$$

$$\dot{z} = \begin{pmatrix} \dot{z_0} \\ \dot{z_1} \\ \dot{z_2} \end{pmatrix} = \begin{pmatrix} -(z_1 + z_2) \\ z_0 + a z_1 \\ b + z_2(z_0 - c) \end{pmatrix} + \alpha(x - z). \tag{19}$$

where $\sigma, \rho, \beta, a, b, c \in \mathbb{R}$ are parameters and $\alpha \in \mathbb{R}$ is the coupling strength with coupling functions $(x - y)$ and $(x - z)$. Our aim is to show that if the coupling strength α is sufficiently strong, then the response Rössler systems get synchronized, i.e. $||y(t) - z(t)|| \to 0$ as $t \to \infty$. After assuming $\dot{\Delta} = \dot{y} - \dot{z} = f(y) - f(z) - \alpha\Delta$, one can proceed just as in the case of Complete Synchronization and get

$$\implies ||\Delta(t)|| \leq Ce^{(-\alpha)t} \tag{20}$$

$$\implies \alpha_c = \Lambda. \tag{21}$$

where Λ is the Lyapunov exponent of the orbit $y(t)$.

3. Sparse Identification of Non-linear Dynamics from data

3.1. **Sparse Regression.**

To identify the governing equations of dynamics in networks we store the data of all dynamics in each dimension. The SINDY algorithm perfectly works on the dynamical systems which have relatively few terms in their equations. Let's consider a non-linear dynamical system f:

$$\dot{x} = f(x). \tag{22}$$

Then let Θ be a data library of candidate non-linear functions of X which is the stored data matrix of the dynamic

$$\Theta(X) = \begin{bmatrix} 1 & X & X^2 & \dots & \sin(X) & \sin(2X) & \dots \end{bmatrix}, \tag{23}$$

where X is the stored data of a specific trajectory for some time length. Then our dynamical system can be represented as follows:

$$\dot{X} = \Xi\Theta^T(X), \tag{24}$$

where Ξ is the matrix composed of corresponding sparse coefficient vectors.

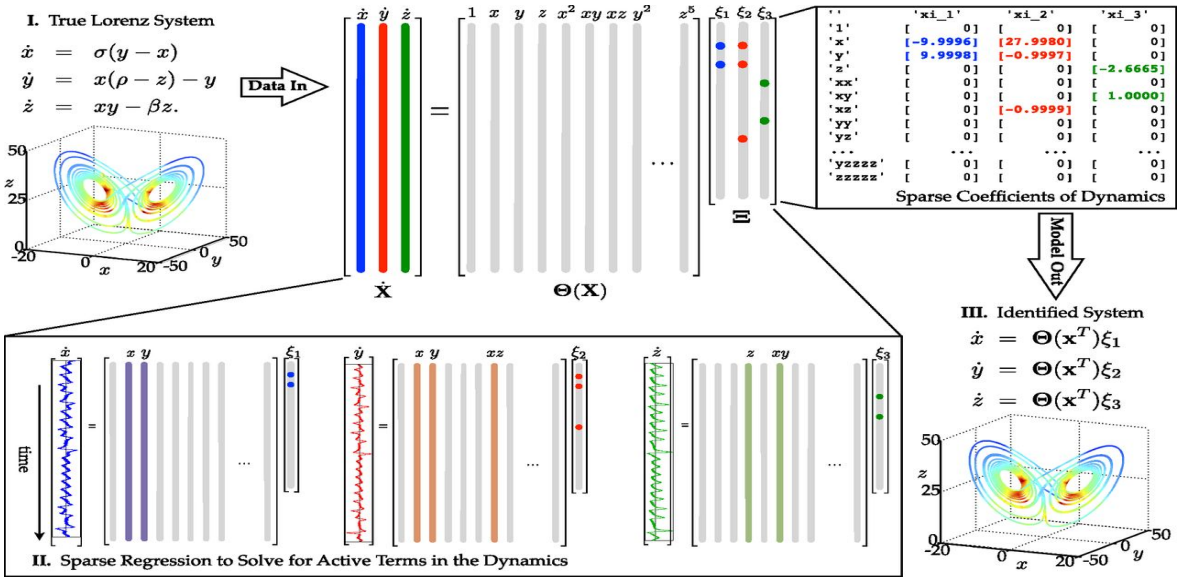


FIGURE 2. Sparse Regression Process [4]

Note that the choice of the library of candidates is crucial for the SINDY algorithm. It is possible to include many different libraries such as polynomials, trigonometric functions, etc. The $k-th$ row of Ξ is found using a sparse regression algorithm, such as Lasso:

$$\xi_k = argmin_{\xi_k}||\dot{X}_k - \Xi_k\Theta^T(X)||_2 + \alpha||\xi_k||_1, \tag{25}$$

where $\dot{X}_k$ represents the $k-th$ row of $\dot{X}$. The $||.||_1$ term promotes sparsity in the coefficient vector ξ_k. The parameter α is selected to identify the Pareto optimal model that best balances low model complexity with accuracy [3].

3.2. **Method.**

Using sparse regression to identify the governing equations of a network works if there is no synchronization. This is because synchronized dimensions of X prevents the recognition of our library. This is handled by adding an external input to the synchronized parts of the networks. The minimum of number external input signals required for a correct reconstruction is found by reassessing the data after implementing external input signals on each combination of synchronized vertices. Finally, when all data can be distinguished the investigation continues with the regression methods, including sparse regression.

3.3. **An Easy Example.**

The most basic example could be to consider a network of 3 vertices where a Lorenz system drives two identical Rössler systems as in equations (17), (18), and (19).

Lorenz system V_0

$$\dot{x_0} = 10(x_1 - x_0)$$

$$\dot{x_1} = x_0(28 - x_2) - x_1$$

$$\dot{x_2} = -\frac{8}{3}x_2 + x_0x_1$$

Coupled Rössler system V_1

$$\dot{x_3} = -(x_4 + x_5) + 2(x_0 - x_3)$$
$$\dot{x_4} = x_3 + 0.2x_4 + 2(x_1 - x_4)$$
$$\dot{x_5} = 0.2 + x_5(x_3 - 5.7) + 2(x_2 - x_5)$$

Coupled Rössler system V_2

$$\dot{x_6} = -(x_7 + x_8) + 2(x_0 - x_6)$$
$$\dot{x_7} = x_6 + 0.2x_7 + 2(x_1 - x_7)$$
$$\dot{x_8} = 0.2 + x_8(x_6 - 5.7) + 2(x_2 - x_8)$$

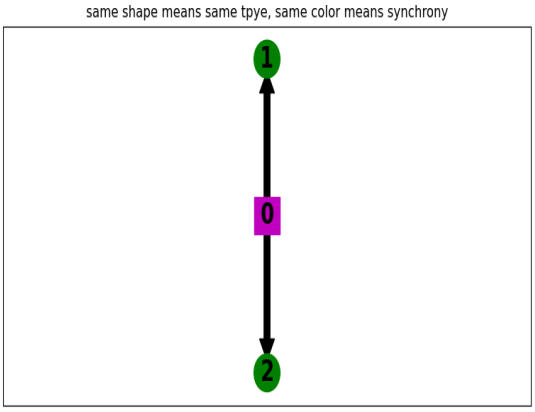


FIGURE 3. Initial Network

In this example, it is ensured that the coupling strength, $\alpha = 2$, is large enough to lead Rössler oscillators to get synchronized. Also, the initial states are randomly chosen from a uniform distribution over the semi-open interval $[0, 1)$. After taking out the data of the transient time required for synchronization, it is tried to identify the dynamics by SINDY algorithm.

```
x0' = -9.999 x0 + 9.999 x1
x1' = 27.992 x0 + -0.998 x1 + -1.000 x0 x2
x2' = -2.666 x2 + 1.000 x0 x1
x3' = 2.000 x0 + -680096307.848 x3 + 219368309.423 x4 + 680096295.849 x6 + -219368299.424 x7
x4' = 2.000 x1 + 1487219.231 x3 + 8586637.721 x4 + -1487174.238 x6 + -8586640.720 x7 + -12494873.894 x3
x5 + 153878128.258 x3 x8 + -158059912.142 x5 x6 + 16676656.778 x6 x8
x5' = 2.000 x2 + 30744146.000 x5 + -30744150.666 x8 + 168107638.517 x3 x4 + 153404997.701 x3 x7 +
-155273679.470 x4 x6 + -166238955.749 x6 x7
x6' = 2.000 x0 + -680096296.251 x3 + 219368299.631 x4 + 680096284.252 x6 + -219368289.632 x7
x7' = 2.000 x1 + 1487219.232 x3 + 8586637.701 x4 + -1487174.239 x6 + -8586640.700 x7 + -12494873.489 x3
x5 + 153878127.915 x3 x8 + -158059911.786 x5 x6 + 16676656.360 x6 x8
x8' = 2.000 x2 + 30744146.055 x5 + -30744150.721 x8 + 168107637.768 x3 x4 + 153404998.362 x3 x7 +
```

FIGURE 4. The result found by SINDY Process without External Signal

It is seen in the figure 2 that the synchronized Rössler oscillators cannot be recovered well. Also, the suggested network by the identification is not the one we have initially started. In figure (3) we see that the equation of $\dot{x}_3$ includes $\dot{x}_0$ and $\dot{x}_6$ which belong to Lorenz system and the second Rössler system, respectively. Hence, the reconstruction fails and it is found as in figure (4).

FIGURE 5. The Wrong Reconstruction of the Initial Network

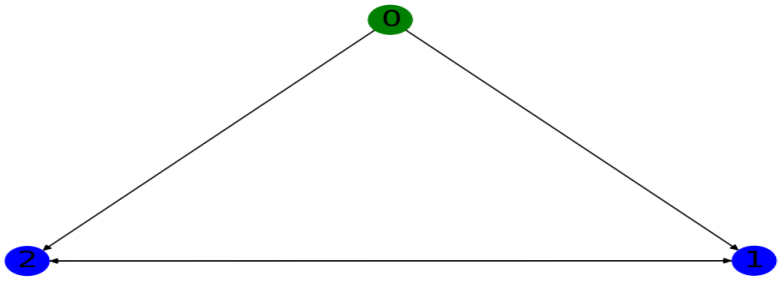


Figure (4) and figure (3) show that we indeed need to implement an external input signal on one of the synchronized vertices so that they can be distinguished by SINDY algorithm. Also, note that there is no problem with the identification of non-synchronized vertices with any others as the Lorenz oscillator given in our example. In figure (5), we see the correct identification where the second Rössler oscillator is given a small external signal $u_0 = \sin(t)$ and it also suggests that the reconstruction should be as same as the initial network.

```
x0' = -9.999 x0 + 9.999 x1
x1' = 27.992 x0 + -0.998 x1 + -1.000 x0 x2
x2' = -2.666 x2 + 1.000 x0 x1
x3' = 2.000 x0 + -11.999 x3 + 9.999 x4
x4' = 2.000 x1 + 44.992 x3 + -2.999 x4 + -1.000 x3 x5
x5' = 2.000 x2 + -4.666 x5 + 1.000 x3 x4
x6' = 2.000 x0 + -11.999 x6 + 9.999 x7 + 4.000 u0
x7' = 2.000 x1 + 44.992 x6 + -2.999 x7 + 4.000 u0 + -1.000 x6 x8
x8' = 2.000 x2 + -4.666 x8 + 3.999 u0 + 1.000 x6 x7
```

FIGURE 6. The result found by SINDY Process with External Signal u_0

4. A few Toy Models

Let's consider a directed star graph S_n, n is a natural number representing the number of leaves, where the internal vertex is a Lorenz oscillator and the leaves are Rössler oscillators and the directions are from the internal vertex to the leaves. In case the coupling is sufficiently strong as in section 3.3, all the Rössler oscillators on the leaves will synchronize. To be able to reconstruct this directed star graph S_n we have to give external signal inputs to any $n-1$ many leaves. It can be directly derived that the more leaves we have the more external signals we need for reconstruction. However, adding link to existing network may not imply that we will need more external signals. The following toy models illustrate this situation. Also, they are given to provide an insight in understanding the synchronization relation.

Suppose that in any case we work on, the coupling parameters are large enough to ensure that synchronization occurs if it is possible and synchronization can only occur between the same type of dynamics.

4.1. **Toy Model 1.**

Let A_1 be the adjacency matrix of our first toy model M_1

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{26}$$

Here A_1 has an entry $a_{ij} = 1$ if there exists an edge $v = (j, i) \in M_1$ and $a_{ij} = 0$ otherwise. Then the graph M_1 of A_1 is given as in figure 6.

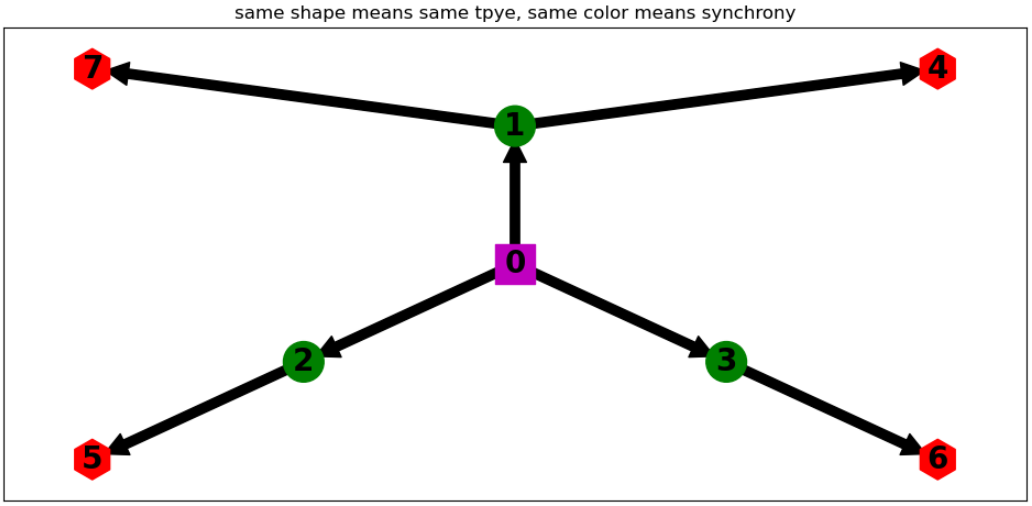


Figure 7. Toy Model 1, M_1

Assume that the vertices given with same shape a square, circle or hexagon have the same dynamics type. The exact type of each oscillator can be found under the section 8 where written codes are provided. Our analysis shows that

$$\text{Synchronized groups} : \{\{v_1, v_2, v_3\}, \{v_4, v_5, v_6, v_7\}\},$$
$$\text{Synchronized vertices regardless of their groups} : \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}.$$

We see that the vertices v_1, v_2, and v_3 get synchronized with each other and the vertices v_4, v_5, v_6, and v_7 get synchronized with each other. It is easy to understand who is synchronized with whom when we keep track on the drivers of each vertices. For instance, the vertex v_7 has driver v_1 which is driven by the vertex v_0 and the vertex v_5 has driver v_2 which has a synchronous motion with vertex v_1 and driven by the vertex v_0. Hence, the vertices v_7 and v_5 should be synchronized. Note that the easiest way of reconstructing this network is to perturb all the vertices except one. However, our aim is to handle this by perturbing minimum number of vertices. In this case, 2 of the vertices v_1, v_2, and v_3 must be given distinct external input signals so that they can be distinguished from each other. Also, note that no matter which 2 combination of those 3 vertices are given external signal the synchronized motion of v_7 and v_4 will not be ruined. That implies another external input signal must be given to one of these. Hence, it is shown that at least 3 vertices must be given an external signal to fully reconstruct the network. It can be ensured by choosing the following 3 vertices:

$$\{v_2, v_3, v_7\}.$$

Now note that v_2, v_3, and v_7 can be distinguished from each other and they will have distinct impact on their children. However, whether the vertex v_1 is perturbed or not it will still have the same impact on its children since the coupling functions do not differ from each other. That is why giving another external input signal to vertex v_7 is suggested.

4.2. **Toy Model 2.**

Let A_2 be the adjacency matrix of our second toy model M_2

$$A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{27}$$

Then the graph M_2 of A_2 is given as in figure 7 and its only difference from M_1 is the extra vertex $(4, 3)$. We see that this difference takes the vertex v_3 out of synchronization with v_1 and v_2 since, now, it has an extra driver that v_1 and v_2 do not have. Also, this causes that v_6 will no longer be in synchrony with the vertices v_4, v_5, and v_7.

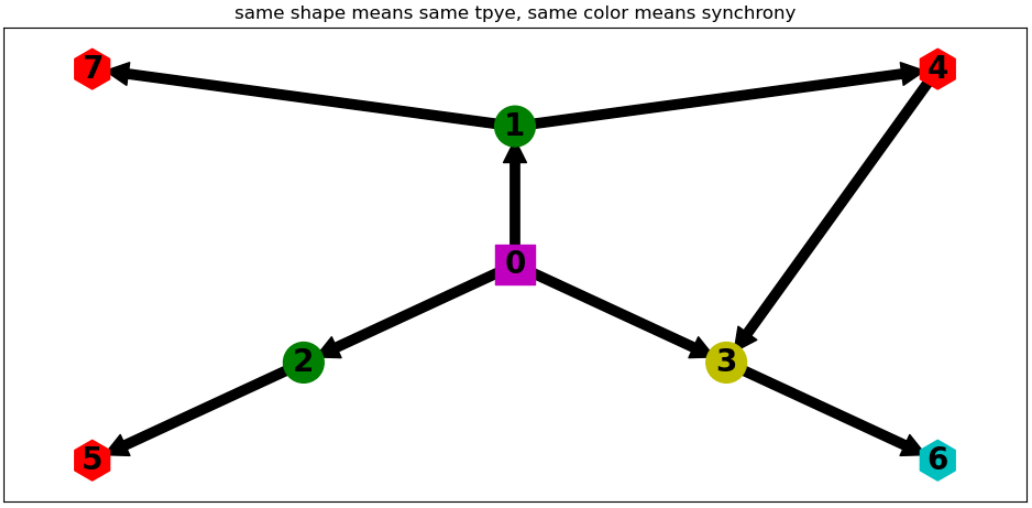


FIGURE 8. Toy Model 2, M_2

Hence, we find the synchronization relations as

$$\text{Synchronized groups} : \{\{1,2\},\{4,5,7\}\},$$
$$\text{Synchronized vertices regardless of their groups} : \{1,2,4,5,7\}.$$

In this case, vertex v_3 should no longer be given an external signal input and the minimum number of vertices to be perturbed diminishes from 3 to 2. Those are the vertices

$$\{v_2, v_7\}.$$

4.3. **Toy Model 3.**

Let A_3 be the adjacency matrix of our third and last toy model M_3

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{28}$$

Then the graph M_3 of A_3 is given as in figure 8 and its difference from M_1 is the extra vertices $(4,2)$ and $(7,3)$. We see that this difference ruins the synchronization of vertices v_1, v_2, and v_3 since, now, the vertives v_2 and v_3 have extra drivers v_4 and v_7, respectively, that v_1 does not have. This change results in the synchronization of v_3 and v_2 since they have either the same driver or drivers that are synchronized. That also causes the synchronization of the vertices v_5 and v_6.

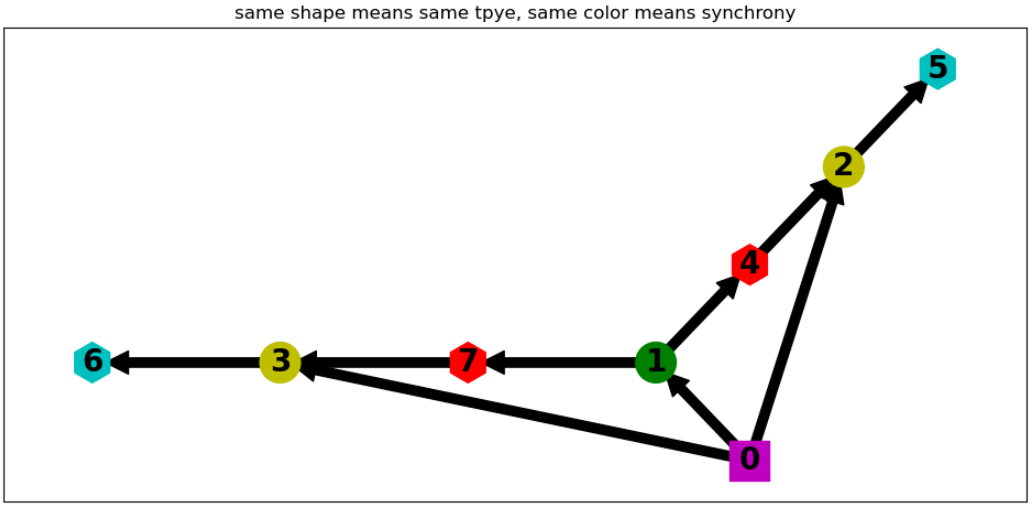


FIGURE 9. Toy Model 3, M_3

Hence, we find the synchronization relations in our last toy model as

$$\text{Synchronized groups} : \{\{2,3\},\{4,7\},\{5,6\}\},$$
$$\text{Synchronized vertices regardless of their groups} : \{2,3,4,5,6,7\}.$$

Although we get a seemingly symmetric graph, the minimum number of vertices to be perturbed surprisingly diminishes from 2 to 1 as we compare it with the second toy model M_2. Giving an external input to only vertex v_7 now provides us with the correct reconstruction of M_3 since it helps us to distinguish all the vertices that are in synchrony from each other.

In the next chapter, we enlarge our analysis for randomly generated Barabasi-Albert models after providing some well-known assertions about it.

5. MATHEMATICAL ANALYSIS

5.1. General Assertions about Barabasi-Albert Model.

A BA model is a model where vertex connectivities follow a scale-free power-law distribution. A scale-free power-law distribution is originated from two properties: (i) networks expand continuously by the addition of new vertices, and (ii) new vertices attach preferentially to sites that are already well connected [1].

Suppose we create a graph by the following rules:

- (Growth) In each step a new vertex with $m \in \mathbb{N}/\{0\}$ edges where each edge links the new vertex to an existing vertex will be added. If we have no initial vertices then the new node will have m loops.
- (Preferential Attachment) An edge of the new vertex at step t connects it to an existing vertex v_i with probability

$$P_i = \frac{k_i}{\sum_{j<N_t} k_j} = \frac{k_i}{2+2m(t-1)}, \tag{29}$$

where $N(t)$ denotes the number existing vertices and k_j is the degree of vertex v_j.

This BA process after $t \in \mathbb{N}^{\geq 0}$ steps creates t new vertices and mt new edges. Also, note that since this is not a directed graph each edge contributes the degree of two vertices. Hence, the new graph at step t has $2mt$ number of total degrees. Let us first analyze the degree distribution of this BA model at time t, denoted by $p_k(t)$. Assume that the number of vertices with degree $k \geq m$ at step t is denoted by $N_k(t)$. Then the degree distribution $p_k(t)$ is found as

$$p_k(t) = \frac{N_k(t)}{t}. \tag{30}$$

Now let us investigate how $N_k(t)$, $k > m$ changes after one further step. It increases if a vertex of degree $k-1$ acquires a new edge and decreases if a vertex of degree k acquires a new edge. Hence, its expected number becomes

$$\begin{aligned}
N_k(t+1) &= N_k(t) + [p_{k-1}(t)N(t)][\frac{k-1}{2mN(t)}]m - [p_k(t)N(t)][\frac{k}{2mN(t)}]m && (31)\\
&= N_k(t) + p_{k-1}(t)\frac{(k-1)}{2} - p_k(t)\frac{k}{2} && (32)\\
&= tp_k(t) + p_{k-1}(t)\frac{(k-1)}{2} - p_k(t)\frac{k}{2} && (33)\\
\implies (t+1)p_k(t) &= tp_k(t) + p_{k-1}(t)\frac{(k-1)}{2} - p_k(t)\frac{k}{2} \qquad \text{by (32)} && (34)\\
\implies p_k(t) &= p_{k-1}(t)\frac{k-1}{k+2} && (35)
\end{aligned}$$

Since there is no vertex with degree less than m and in each step one new vertex with degree m emerges $p_m(t+1)$ is found as

$$\begin{aligned}
N_m(t+1) &= p_m(t) + 1 - p_m(t)\frac{m}{2} && (36)\\
\implies (t+1)p_m(t) &= p_m(t) + 1 - p_m(t)\frac{m}{2} && (37)\\
\implies p_m(t) &= \frac{2}{m+2}. && (38)
\end{aligned}$$

With a recursive approach where k starting from $m+1$ we get the following result by equations (35) and (38)

$$p_{m+1}(t) = p_m(t)\frac{m}{m+3} = \frac{2m}{(m+2)(m+3)} \tag{39}$$

$$p_{m+2}(t) = p_{m+1}(t)\frac{m+1}{m+4} = \frac{2m(m+1)}{(m+2)(m+3)(m+4)} \tag{40}$$

$$p_{m+3}(t) = p_{m+2}(t)\frac{m+2}{m+5} = \frac{2m(m+1)}{(m+3)(m+4)(m+5)} \tag{41}$$

$$p_{m+4}(t)p_{m+3}(t)\frac{m+3}{m+6} = \frac{2m(m+1)}{(m+4)(m+5)(m+6)} \tag{42}$$

$$\implies p_k = \frac{2m(m+1)}{k(k+1)(k+2)}. \tag{43}$$

We can also analyse the degree dynamics in our BA model. Let v_i be any vertex that emerged at time $t = i \in \mathbb{N}$. Then its degree k_i depending on time increases with the rate

$$\frac{dk_i}{dt} = m\frac{k_i}{2mt - m} = \frac{k_i}{2t-1}, \tag{44}$$

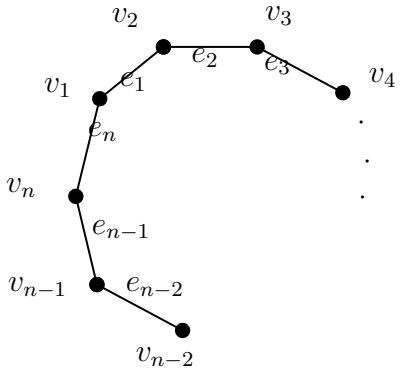
since its probability of having a new link is proportional to its degree divided by the total degree of all vertices except the new one. By integrating and implementing the initial condition $k_i(t = i) = m$ we find [2]

$$k_i(t) = m\sqrt{\frac{2t-1}{2i-1}}. \tag{45}$$

Also, it might be helpful to know that the probability, n_{kl}, of finding a link that connects a node of degree k to an ancestor node of degree l in the BA model for the special case of $m = 1$ is given as [8]

$$n_{kl} = \frac{4(l-1)}{k(k+1)(k+l)(k+l+1)(k+l+2)} + \frac{12(l-1)}{k(k+l-1)(k+l)(k+l+1)(k+l+2)}. \tag{46}$$

A BA model for the special case $m = 1$ where each vertex brings about exaclty 1 edge always generate a tree since assuming the existence of two vertices that are connected by more than one path will result in an $n-cycle$ which is not possible by the following proposition.



Assume without loss of generality that v_1 come into existence earlier than v_2. That implies v_2 brings about e_1 and v_3 comes after v_2. As an inductive result, v_n comes later than v_{n-1} and brings about e_{n-1}. This means it is impossible to have $e_n \in E(V)$. Hence, it is clear that for the special case $m = 1$ a BA model gives us a tree.

The results found in this section will be used in the following section to find the minimum number of vertices to be perturbed for our reconstruction problem where randomly generated BA model $m = 1$ is chosen to be investigated.

5.2. **Minimum Number of External Signals Required for Reconstruction.**

We will only assume that the vertices will represent the same dynamics if they are in the same generation. In this case, note that any two vertices in the same generation will be synchronized since their drivers are totally the same and sufficient strength of coupling is provided. In case two vertices have the same type of dynamics but are in different generation, they never get synchronized since their ancestors at some generation differ.

We can come up with correct analyses for our directed BA network, where direction is implemented from the existing vertex to the new one, by investigating the initial BA model where no direction was introduced yet. For instance, vertices with degree 1 in a tree form the leaves and and the total number of leaves doesn't change after introducing directions on the existing edges. Also, note that the initial vertex will be the only one whose in-degree is 0 which means it is our root according to our process of adding directions whereas the rest will have in-degree 1 which is the link they came with. Hence, in-degree of a vertex rather than v in the directed graph is its degree in a randomly generated BA model subtracted 1.

As a first approach, note that to have a distinct data of synchronized vertices, which is needed for reconstruction of the network, distinct perturbations can be given all the vertices whose parent is the same. However, it is also fine to perturb all children except one. That is same as counting the out-degree of each vertex minus 1. Obviously, the total out-degree of all vertices derived from a BA model (m=1) with N vertices is

$$v_1 + \sum_{i=2}^{N}(v_i - 1) = \sum_{i=1}^{N} out(v_i) = N - 1. \tag{47}$$

Since the leaves are the only ones whose out-degree is 0 any other node will have a possibility to have synchronized children and so will be the ones which we sum up their out-degree subtracted 1 to find the first approximation A_1 of the minimum number of perturbations required P. Let W be the subset of the vertex set V whose elements have out-degree greater than 0. Then what we are looking for is

$$A_1 = \sum_{w\in W}(out(w) - 1) = \sum_{w\in W} out(w) - |W|. \tag{48}$$

If we say U is the subset of V whose out-degree is equal to 0, then (4) becomes

$$\sum_{w\in W} out(w) - |W| = \sum_{i=1}^{N} out(v_i) - \sum_{u\in U} out(u) - (|V| - |X|) \tag{49}$$

$$\approx N - 1 - (N - Np_1) = Np_1 - 1, \tag{50}$$

where p_1 is the approximated number of vertices with degree 1 which is found by the exact degree distribution of BA model (43) as

$$Np_k = N\frac{2m(m+1)}{k(k+1)(k+2)} \tag{51}$$

$$\implies Np_1(m=1) = \frac{4N}{6} \tag{52}$$

$$\implies P \approx A_1 = \frac{4N}{6} - 1. \tag{53}$$

Assuming we have N vertices in our BA model we find another approximation A_2 of (47) by combination of (45) and (51) as

$$P \approx A_2 = \sum_{i=1}^{N-Np_k} (\sqrt{\frac{2t-1}{2i-1}} - 2), \tag{54}$$

since degree dynamics (45) decreases when t is fixed and i ranges from 1 to N we can assume the first $N - Np_k$ vertices will have degree greater than 1.
Since equation (50) tells us to find the vertices with degree 1 we can employee equation (46) and get another approximation, say A_3,

$$A_3 = N\sum_{l=1}^{N} n_{1l} - 1 \approx Np_1 - 1. \tag{55}$$

We will be illustrating these approximations, A_1, A_2, and, A_3, in the next section.

6. Numerical Results

6.1. **N=10.**

- After a thousand of simulations it is found that $P \approx 5.06$
- $A_1 = 5.\overline{6}$
- $A_2 \approx 2.36$
- $A_3 = 3.\overline{72}$

6.2. **N=20.**

- After a thousand of simulations it is found that $P \approx 11.71$
- $A_1 = 12.\overline{3}$
- $A_2 \approx 4.95$
- $A_3 \approx 10.35$

6.3. **N=50.**

- After a thousand of simulations it is found that $P \approx 31.69$
- $A_1 = 30.\overline{3}$
- $A_2 \approx 14.04$
- $A_3 \approx 33.34$

It seems our approximations A_1 and A_3 are strong enough to squeeze the solution in a small range in most of the cases. However, It is seen that the second approximation A_2 is weak according to the simulation results of any number N. The main reason for this is the square root function in its equation and there is no correct way of rounding square root in this case. Also, the degree dynamics suggest that a vertex emerging earlier than another has greater degree than the latecomer which is not always true.

7. Further Discussion

Reconstruction of interacting dynamics from data is a concern of the last decades since it is informative for the future, past and present tenses. This project gives some results and ideas for a particular case. However, the problem quickly gets complicated if there is no notion of generations as in trees or if the edges are not directed since it also ruins the idea of working on generations as we did in this project. Moreover, although we found an approximation of minimum number of required external signals for reconstruction there is a much more complicated problem that is which vertices should be implemented external signal to distinguish distinct synchronized dynamics from each other.

8. Main Code

```
"""
@author: Mehmet Kirtisoglu
"""
import math
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import networkx as nx
from itertools import combinations
import pysindy as ps

mean = 0
r=3
#Parameters of Rossler System
a = 0.2
b = 0.2
c = 5.7

#Parameters of Lorenz System type 1
rho_0 = 28
sig_0 = 10
beta_0 = 8/3

#Parameters of Lorenz System type 2
```

```
rho = 45
sig = 10
beta = 8/3

c_alpha = 1.5 #Coupling Parameter


#Randomly Generated Barabasi-Albert Network without directions
G = nx.barabasi_albert_graph(20,1)
Adj = nx.convert_matrix.to_numpy_array(G) #Adjaceny Matrix of G
for i in range(len(Adj)):
    for j in range(i+1,len(Adj)):
        Adj[i][j]=0

A = Adj # Lower Triangular Adjaceny Matrix of G
n=len(A)

DG = nx.DiGraph() # Directed Graph of G
for i in range(n):
    for j in range(n):
        if A[i][j]==1:
            DG.add_edges_from([(j,i)])

spl = nx.shortest_path_length(DG,source=0)
dynamic_type = []
for i in range(n):
    if spl[i]%3==0:
        dynamic_type.append(0)
    elif spl[i]%3==1:
        dynamic_type.append(1)
    else:
        dynamic_type.append(2)

"""
Out = nx.out_degree_centrality(DG)
Out_inverse = [(value,key) for key, value in Out.items()]
a = max(Out_inverse)[1] #Node having the maximum out degree
"""
"""
#Drawing directed graph DG with colors and labels
color_map = []
for node in DG:
```

```
    if dynamic_type[node]==0:
        color_map.append('green')
    elif dynamic_type[node]==1:
        color_map.append('blue')
    else:
        color_map.append('red')
figure = plt.figure(figsize=plt.figaspect(0.5))
figure.add_subplot(111)
nx.draw_spring(DG, node_color=color_map, with_labels= True)
"""
#Perturbation that should be recognized by pysindy as "u"
def noise_signal(t):
    return np.column_stack([np.sin(t)])

def Rossler(x): #Returns Rossler System
    return np.array([-(x[1]+x[2]), x[0]+a*x[1], x[2]*(x[0]-c)+b])

def lorenz_1(x): #Returns Lorenz Type1 system (rho_0=28)
    return np.array([sig_0*(x[1]-x[0]),
    x[0]*(rho_0-x[2])-x[1], x[0]*x[1] - beta_0*x[2]])

def lorenz_2(x): #Returns Lorenz Type2 system (rho=45)
    return np.array([sig*(x[1]-x[0]),x[0]*(rho-x[2])-x[1],
    x[0]*x[1] - beta*x[2]])

#Returns network according to Graph and external signal
def Network(x,t):
    u = noise_signal(t)
    x = x.reshape(n,3)
    dx = np.zeros_like(x)
    for i in range(n):
        if dynamic_type[i]==0:
            dx[i] = lorenz_1(x[i])
        elif dynamic_type[i]==1:
            dx[i] = lorenz_2(x[i])
        else:
            dx[i] = Rossler(x[i])
    for i in range(n):
        for j in range(n):
            if A[i][j]==1:
                dx[i][0] += c_alpha*(x[j][0]-x[i][0])
                dx[i][1] += c_alpha*(x[j][1]-x[i][1])
```

```
                dx[i][2] += c_alpha*(x[j][2]-x[i][2])
        dx[i][0] += n_coef[i]*math.sin(t)
        dx[i][1] += n_coef[i]*math.sin(t)
        dx[i][2] += n_coef[i]*math.sin(t)
    dx = dx.flatten()
    return dx


def Error_func(D): #Returns synchronized nodes and groups from data
    Err=np.zeros((len(A),len(A)))
    Sync = []
    for i in range(len(A)):
        for j in range(i+1,len(A)):
            for q in range(3):
                x_0 = np.array(D[:, 3*i+q])
                x_1 = np.array(D[:, 3*j+q])
                Err[i,j] += np.mean(np.abs(x_0-x_1))
                Err[j,i]=Err[i,j]
                if Err[i,j]==0:
                    Sync.append((i,j))
    Sync_map = []
    control_map = []
    for i in range(n):
        x = [i]
        for j in range(i+1,n):
            if (i,j) in Sync:
                if i in control_map:
                    break
                else:
                    x.append(j)
                    control_map.append(j)
        if len(x)>1:
            Sync_map.append(x)
        control_map.append(i)
    Sync_nodes = []
    for i in Sync_map:
        for j in i:
            Sync_nodes.append(j)
    return Sync_map, Sync_nodes

#Generate data of initial graph without any external signal
dt = 0.002
```

```
t_train = np.arange(0,600,dt)
xyz0_train = np.random.rand((n*3)) #Inital Conditions
xyz0_train = xyz0_train[:][200000:]
n_coef = np.zeros(n)
xyz_train = odeint(Network,xyz0_train,t_train)


Sync_groups,Sync_nodes = Error_func(xyz_train)
Sync_nodes = np.sort(Sync_nodes)
n_sync_groups = len(Sync_groups)
#print("Number of Synchronized groups:",
n_sync_groups,"=|", Sync_groups,"|")

n_sync_nodes = len(Sync_nodes)
#print("Number of Synchronized Nodes:", n_sync_nodes,"=|", Sync_nodes,"|")

#Combinations of synchronized nodes are given external signals
#and the system is solved again.
g = Sync_groups
v = Sync_nodes
n_g = len(g)
n_v = len(v)
for i in range(1, (n_v-n_g)+1):
    for j in combinations(v,i):
        for e in j:
            n_coef[e] = e*2
            combination=j
        xyz_train_trial = odeint(Network,xyz0_train,t_train)
        xyz_train_trial = xyz_train_trial[:][200000:]
        Sync_group_trial, Sync_node_trial = Error_func(xyz_train_trial)
        if len(Sync_group_trial)==0:
            break
        for e in j:
            n_coef[e] = 0
    if len(Sync_group_trial)==0:
        break

print([combination])
mean += len(combination)
print(i,mean)
#print(n_min)
"""
```

```
#Control inputs
noise_train = noise_signal(t_train)
#Instantiate and fit the sindy model
stlsq_optimizer = ps.STLSQ(threshold=0.15, max_iter=100, alpha=0.01)
model = ps.SINDy(optimizer=stlsq_optimizer)
model.fit(xyz_train_trial,u=noise_train, t=dt)
model.print()


#Validation of Model
M_coef = model.coefficients() #coefficients found by S ndy
M_coef = np.array(M_coef)
M_nonzero_coef = []
for i in M_coef:
    indices = np.where(i==0.0)
    i = np.delete(i,indices)
    for j in i:
        M_nonzero_coef.append(j)

N_coef = [] #real coefficients of each x_i

for i in range(n):
    if dynamic_type[i]==0:
        X_coef = [-sig_0,sig_0,rho_0,-1,-1,-beta_0,1]
    elif dynamic_type[i]==1:
        X_coef = [-sig,sig,rho,-1,-1,-beta,1]
    else:
        X_coef = [-1,-1,1,a,b,-c,1]
    for j in range(n):
        if A[i][j]==1:
            if dynamic_type[i]==0:
                X_coef[0] -= c_alpha
                X_coef.append(c_alpha)
                X_coef[2] -= c_alpha
                X_coef.append(c_alpha)
                X_coef[5] -= c_alpha
                X_coef.append(c_alpha)
            else:
                X_coef.append(c_alpha)
                X_coef.append(c_alpha)
                X_coef[3] -= c_alpha
                X_coef.append(c_alpha)
```

```
                X_coef[5] -= c_alpha
                X_coef.append(c_alpha)
        if n_coef[j]!=0:
            X_coef.append(n_coef[j])
            X_coef.append(n_coef[j])
            X_coef.append(n_coef[j])
            N_coef.extend(X_coef)


if len(M_nonzero_coef)<len(N_coef):
    M_nonzero_coef.extend(np.zeros((len(N_coef)-len(M_nonzero_coef))))
else:
    N_coef.extend(np.zeros((len(M_nonzero_coef)-len(N_coef))))

N_coef = np.array(np.sort(N_coef))
M_nonzero_coef = np.array(np.sort(M_nonzero_coef))
N_coef_len = len(N_coef)

Err_coef = np.sum(np.abs(N_coef-M_nonzero_coef))/N_coef_len
print(Err_coef)

"""
```

References

[1] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". *Science* 286.5439 (1999), pp. 509–512.

[2] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge: Cambridge University Press, 2016. ISBN: 9781107076266 1107076269. URL: http://barabasi.com/networksciencebook/.

[3] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. "Sparse identification of nonlinear dynamics with control (SINDYc)". *IFAC-PapersOnLine* 49.18 (2016), pp. 710–715.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". *Proceedings of the National Academy of Sciences* 113.15 (Mar. 2016), pp. 3932–3937. ISSN: 1091-6490. DOI: 10.1073/pnas.1517384113. URL: http://dx.doi.org/10.1073/pnas.1517384113.

[5] Wikimedia Commons. *File:Orbital instability (Lyapunov exponent).png — Wikimedia Commons, the free media repository*. [Online; accessed 27-December-2021]. 2021. URL: https://commons.wikimedia.org/w/index.php?title=File:Orbital_instability_(Lyapunov_exponent).png&oldid=576462176.

[6] Damien Depannemaecker et al. “A unified physiological framework of transitions between seizures, sustained ictal activity, and depolarization block at the single neuron level”. *BioRxiv* (2021), pp. 2020–10.

[7] Deniz Eroglu, Jeroen Lamb, and Tiago Pereira. “Synchronization of Chaos”. *arXiv preprint arXiv:1703.08296* (2017).

[8] Babak Fotouhi and Michael G Rabbat. “Degree correlation in scale-free graphs”. *The European Physical Journal B* 86.12 (2013), pp. 1–19.

[9] Ljupco Kocarev and Ulrich Parlitz. “Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems”. *Physical Review Letters* 76.11 (1996), p. 1816.

[10] Chunguang Li and Guanrong Chen. “Chaos and hyperchaos in the fractional-order Rössler equations”. *Physica A: Statistical Mechanics and its Applications* 341 (2004), pp. 55–61.

[11] V. I. Oseledec. “A multiplicative ergodic theorem. Characteristic Ljapunov, exponents of dynamical systems”. *Trudy Moskov. Mat. Obšč.* 19 (1968), pp. 179–210. ISSN: 0134-8663.

[12] Tiago Pereira. “Stability of Synchronized Motion in Complex Networks” (Dec. 2011).

[13] Veronika Stolbova et al. “Tipping elements of the Indian monsoon: Prediction of onset and withdrawal”. *Geophysical Research Letters* 43.8 (2016), pp. 3982–3990.

[14] Marcelo Viana. “What’s new on Lorenz strange attractors?” *The Mathematical Intelligencer* 22.3 (2000), pp. 6–19.

[15] Alan Wolf et al. “Determining Lyapunov exponents from a time series”. *Physica D: Nonlinear Phenomena* 16.3 (1985), pp. 285–317.