UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea in Fisica

Tesi di Laurea

Scoprire le leggi del moto dalle dinamiche collettive su reti con il framework SINDy

Discovering the law of motions from collective dynamics on networks with the SINDy framework

Relatore

Prof. Manlio De Domenico

Correlatore

Dott.sa Valeria d'Andrea

Laureando

Andrea Valsecchi

Anno Accademico 2024/2025

Abstract

The scientific method is based on the observation of phenomena, building hypotheses and making measurements to compare theoretical expectation with observation. Is it possible to follow the same process while deducing the underlying model directly from the data without a specific prior expectation? With recent machine learning techniques the answer might be yes and the SINDy framework might be one possible approach to tackle the problem. Sparse Identification of Nonlinear Dynamics (SINDy) is based on machine learning techniques, it gets empirical data as an input and outputs the differential equation(s) that most likely have produced the observed time course of observables of physical interest, by means of an optimization approach. A Thesis work is here proposed to test the SINDy framework, and discover its limits in the context of complex networks dynamics. Specifically, we will explore distinct dynamical processes, such as synchronization via Kuramoto-like models, on the top of a network with varying topology, encoding increasing levels of complexity observed in empirical biophysical systems (such as heterogeneous connectivity, modular and hierarchical structure, spatial embedding). The goal is to understand to which extent machine-assisted approaches like SINDy can help to better understand the structure and the dynamics of empirical complex systems.

Il metodo scientifico si basa sull'osservazione dei fenomeni, la formulazione di ipotesi, e il fare misurazioni per mettere a confronto le aspettative teoriche con i dati osservati. È possibile seguire lo stesso processo mentre si deduce il modello sottostante direttamente dai dati senza una specifica aspettazione a priori? Con le tecniche recenti di machine learning la risposta potrebbe essere sì e il framework SINDy potrebbe essere un possibile approccio per affrontare il problema. Sparse Identification of Nonlinear Dynamics (SINDy) si basa su tecniche di machine learning, prende dati empirici come input e restituisce le equazioni differenziali che con maggior probabilità hanno prodotto le timeseries di osservabili di interesse fisico, attraverso un approccio di ottimizzazione. Viene proposto qui un lavoro di tesi per testare il framework SINDy, e scoprire i suoi limiti nel contesto di dinamiche di reti complessi. In particolare esploreremo diversi tipi di dinamica come la sincronizzazione attraverso modelli basati su quello di Kuramoto, su reti con diverse topologie che racchiudono diversi livelli di complessità osservata nei sistemici empirici biofisici (come la connettività eterogenea, modulare, a struttura gerarchica, o spaziale). L'obiettivo è capire fino a che punto gli approcci di machine-assisted come SINDy possono aiutare per capire meglio la struttura e la dinamica di sistemi complessi empirici.

Contents

Introduction

Have you ever wondered how we can extract the underlying laws of motion and the network structure of a dynamical system from time series data?
This is an important challenge in Network Science, especially when one is interested in characterizing the unknown topology of an interconnected dynamical system from the observation of time series of its units.
In our study, we aim to tackle such an inverse problem. Solving it is important because it allows us to understand the underlying mechanisms that drive the behavior of complex systems, such as social networks, biological systems, and communication networks. To solve this problem, we use a powerful data-driven method called SINDy (the Sparse Identification of Nonlinear Dynamics) and apply it to the Kuramoto model.
In the first chapter, we will see in more detail what a network is, the dynamics at play, and the role of the inverse problem and SINDy.
In the second chapter, we will define some metrics to assess the quality of the reconstruction to solve the inverse problem made by SINDy.
In the third one, we will see different results while varying some parameters, like the coupling in the dynamics or the average degree in the construction process of the network, lastly we will see the complexity analysis made on the SINDy algorithm for the time and the RAM memory used by the algorithm and its limits.

Chapter 1

Complex systems and the SINDy approach to the inverse problem

In this chapter we will examine the essential concepts that will be used in the following chapters, starting with the definition of a complex system, a network and some network models used, following by the dynamics used which in this case is the Kuramoto model, a particular mathematical model used in describing synchronization. Finally, we will see what the SINDy framework is (Sparse Identification of Nonlinear Dynamics), how it works, and the algorithms to solve the sparse regression problem, which will then be used.

1.1 Complex Systems

Before looking into the reconstruction of the dynamics and network structure, it is crucial to define the object of our study. A complex system refers to a system composed of many interconnected and interacting components, giving rise to emergent behavior, which is not easily understood or predicted merely by examining the components in isolation. Typically, complex systems exhibit non-linear behavior, making them challenging to model and understand using traditional reductionist approaches that focus on disassembling the system into smaller, simpler components. Instead, complex systems are commonly studied using interdisciplinary approaches, amalgamating methods from fields such as physics, mathematics, biology, computer science, and social sciences. In our context, we take a particular interest in the approach through networks, which is discussed in the ensuing section.

1.2 Networks

A network in complex systems refers to a set of interconnected entities known as nodes that interact with each other. These entities can represent various systems, including biological organisms, social communities, transportation networks, computer networks, etc. In a network, nodes can be connected by links, also known as edges, which represent the exchange of information or interactions between nodes. For example, consider a social network where each account represents a node and the direct relationships between different accounts symbolize the edges. Networks model how various entities interact, and this concept can be extended to differential equations, where the interaction terms in the dynamics equation rely on the network structure.

The study will employ a range of networks, especially synthetic networks, differing in node size, average number of links per node (referred to as the average degree) and topology, which refers to the specific arrangement or structure of network connections in a complex system. Each network is generated by a particular generative model that accepts various parameters, such as the desired number of nodes in the network, and outputs a network with a certain structure. We will utilize the following models:

- The Erdős–Rényi model [1] (ER) creates random networks where the connections between nodes are made randomly, without any specific underlying structure.

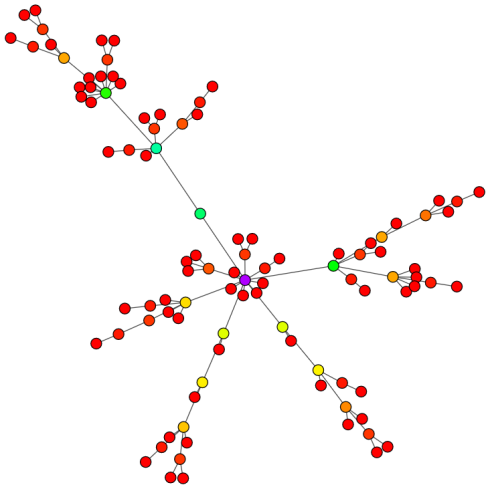


Figure 1.1: Example of a network with 100 nodes, colors not relevant.

- The Lattice model [2] (L) generates regular networks where the nodes are arranged in a regular, repeating pattern. This study will employ the lattice model L where nodes are connected to their nearest neighbors.
- The Watts-Strogatz model [2] (WS) produces small-world networks that blend characteristics of both random and regular networks. In this model, most nodes are connected to their nearest neighbors, but a few connections are established with nodes further away with a certain probability. These "shortcuts" facilitate efficient information dissemination across the network.
- The Barabási–Albert model [3] (BA) constructs scale-free networks where the distribution of connections between nodes adheres to a power law. This implies that a small number of nodes have many connections, while most nodes have only a few connections. Natural and human-made systems following this type of model include the World Wide Web (WWW) and the internet.
- The Stochastic Block model [4] (SBM) is a generative model used for creating networks with a modular or community structure. Nodes are divided into distinct groups or "blocks," and the likelihood of a link between two nodes depends on the blocks to which they belong. Nodes within the same block have a higher probability of being connected to each other than nodes in different blocks.
- The Geometric Random Graphs [5] (GRG) model generates networks by randomly placing nodes in a metric space, such as a plane, and then connecting them based on a distance threshold. In a GRG, the probability of a link between two nodes is determined by their distance from each other. This model is often employed in applications where

An essential tool for network analysis is the adjacency matrix. Represented by the symbol $\hat{A}$, an adjacency matrix is a square matrix that illustrates the connections or links between nodes in a network. This dissertation limits the study to simple, undirected graphs, implying there are no self-loops or multiple edges, and the nodes are connected by edges that lack direction or associated arrow. Therefore, if a connection exists between node A and node B, a reciprocal connection from node B to node A also exists. For these graphs, the adjacency matrix adopts a simpler form, with elements either 0 or 1, depending on whether a link between two nodes exists or not. Specifically, the (i, j) element of the adjacency matrix signifies the connection between node i and node j. If a link between these two nodes exists, the element is set to 1; otherwise, it is set to 0. It is crucial to note that $\hat{A}$ is symmetric, meaning that $\hat{A}(i, j) = \hat{A}(j, i)$ for all i and j.

1.3 Dynamics

In the previous sections, we explored the concept of networks. Now, we will look into the specific dynamics that are used, particularly focusing on the well-known Kuramoto model.

The Kuramoto model [6] is a mathematical model used to investigate the synchronization of coupled oscillators. The model comprises a set of N phase oscillators, each with its own natural frequency. Each

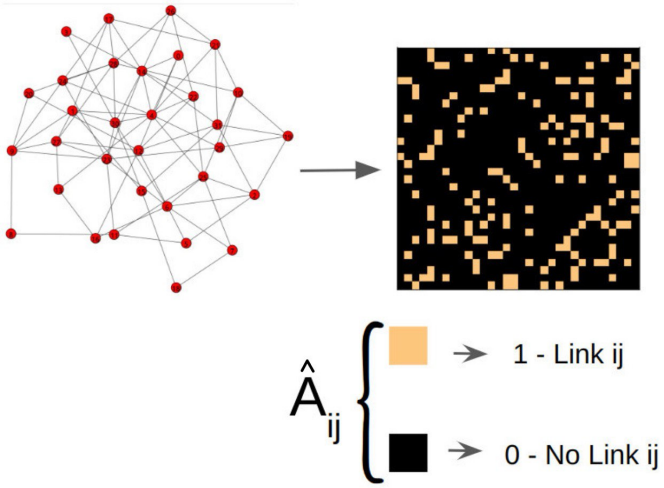


Figure 1.2: Illustrative figure of an adjacency matrix, the color of the relative entries is based if there is a link or not: the color light if there is a link, black otherwise.

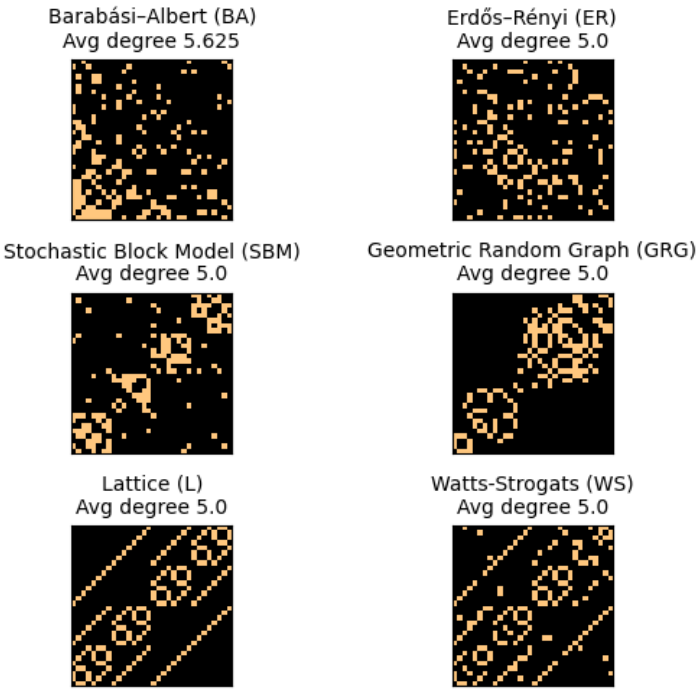


Figure 1.3: 6 different networks, which will be used for the first type of analysis (with a fixed network) in Chapter 3, each with 32 nodes generated with different models, colors means like in Figure 1.2.

oscillator is coupled to some of the other oscillators in the network. The phase of each oscillator evolves according to a system of differential equations that captures the influence of the coupling between the oscillators.

The Kuramoto model has widespread application in the study of synchronization phenomena in various fields such as physics, biology, and engineering. It has been used to analyze firefly synchronization, the coordination of circadian rhythms in cells, and numerous other systems.

This model is often applied to specific networks where not every node is connected to every other node. In particular, the model is defined as follows [7]:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K}{N}\sum_{j=1}^{N} \hat{A}_{ij} \sin\left(\theta_j - \theta_i\right) \quad , \quad i = 1, \ldots, N$$

In this equation, θ represents the phases of the oscillators, ω_i is the natural frequency of the i-th oscillator, K is the coupling strength between nodes (which influences the synchronization velocity between different oscillators), and A_{ij} represents the entry (i, j) of the adjacency matrix of the network being used.

To quantify the degree of synchronization among the oscillators in the network, we employ the "order" parameter r [7], which is the average of the complex unit vectors representing the phase of each oscillator:

$$re^{i\theta} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j} \tag{1.1}$$

In this equation, r represents the magnitude of the complex vector, and θ represents its phase. When the oscillators are perfectly in phase, the order parameter takes a value of 1, indicating complete synchronization. Conversely, when the oscillators are entirely out of phase, the order parameter takes a value of 0, indicating no synchronization.

1.4 SINDy

We have seen the Kuramoto model and how we can apply it to a particular network. Starting from an initial condition, we can integrate the differential equations numerically to observe the temporal evolution of the dynamics in this particular network.

However, we are interested in the opposite: can we predict the network structure and the model starting from temporal time-series, meaning the sequence of data points the model produces over time? This is known as the inverse problem in network science, and it presents an important challenge. One way to approach this is with SINDy, the Sparse Identification of Nonlinear Dynamics.

SINDy (Sparse Identification of Nonlinear Dynamics) [8] is a data-driven method for discovering the governing equations of dynamical systems of the form $\frac{d}{dt}\mathbf{x}(t) = f(\mathbf{x}(t))$ from observational data $\mathbf{x}_i(t)$ using machine learning techniques. In other words, SINDy is a technique for reverse engineering the mathematical equations that describe the behavior of a system, based solely on measurements or observations of the system's behavior over time. SINDy has been successfully applied in a variety of scientific fields, including physics, chemistry, biology, and engineering, to discover the underlying equations governing complex systems from noisy or incomplete data. [15] [16] [17]
The general steps involved in SINDy are as follows:

- Collect data: The first step in applying SINDy is to collect observational data of the system under study. This data consists of time series measurements of the state variables of the system. The data is sampled several times $t_1, t_2, \ldots, t_m$ and then put into two matrices, X and $\dot{X}$, which represent the state of the system at each time point and the rate of change of the state,

respectively.

$$X = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \dots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \dots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \dots & \dot{x}_n(t_m) \end{bmatrix}$$

with $t_1 < t_2 < \cdots < t_m$

- Generate a library of candidate functions $\Theta(X)$: SINDy requires a library of potential functions that can be used to construct candidate terms for unknown equations. This library typically consists of simple mathematical functions, such as polynomials, sine and cosine functions, exponentials, and logarithms.

$$\Theta(X) = \begin{bmatrix} | & | & | & | & | & | & | & | \\ 1 & \mathbf{X} & \mathbf{X}^{P_2} & \dots & \sin \mathbf{X} & \cos \mathbf{X} & \sin(\mathbf{X} - \mathbf{Y}) & \dots \\ | & | & | & | & | & | & | & | \end{bmatrix}$$

  Where $\sin(\mathbf{X} - \mathbf{Y})$ denotes all the combinations:

$$\sin(\mathbf{X} - \mathbf{Y}) = \begin{bmatrix} \sin(x_1(t_1) - x_2(t_1)) & \sin(x_2(t_1) - x_3(t_1)) & \dots & \sin(x_{n-1}(t_1) - x_n(t_1)) \\ \sin(x_1(t_2) - x_2(t_2)) & \sin(x_2(t_2) - x_3(t_2)) & \dots & \sin(x_{n-1}(t_2) - x_n(t_2)) \\ \vdots & \vdots & \ddots & \vdots \\ \sin(x_1(t_m) - x_2(t_m)) & \sin(x_2(t_m) - x_3(t_m)) & \dots & \sin(x_{n-1}(t_m) - x_n(t_m)) \end{bmatrix}$$

  Every column of $\Theta(\mathbf{X})$ is a candidate function for $\mathbf{f}$ of $\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}(t))$ that may appear in the governing equations.
- Solve a sparse regression problem: Given the set of candidate terms and the time series data, SINDy formulates a sparse regression problem to identify the most likely combination of terms that accurately describes the system's dynamics. The sparse regression problem, formulated as $\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi$, involves finding the coefficients $\Xi = \left[\xi_1, \xi_2 \dots \xi_n\right]$ of the candidate terms that minimize the difference between the predicted and observed dynamics, subject to a sparsity constraint.
  The matrix Ξ contains the coefficients of the sparse linear combination of basis functions that approximate the components of the function $\mathbf{f}(x)$. If $\mathbf{f}(x)$ is a vector of length n with components $f_1(x)$, $f_2(x)$, ... , $f_n(x)$, then Ξ is a matrix m × n, where m is the number of basis functions used to approximate each component of $\mathbf{f}(x)$. Each column of Ξ contains the coefficients for the corresponding component of $\mathbf{f}(x)$, expressed as a linear combination of the basis functions.

- Eliminate small coefficients: After solving the sparse regression problem, SINDy prunes away terms with small coefficients if they are smaller than a certain threshold, which correspond to spurious or irrelevant dynamics. The remaining terms correspond to the simplest set of equations that accurately describe the system's behavior.

In the context of the Kuramoto model, we provide SINDy with multiple time series data. The library of functions consists of the sine difference of two variables $\sin(x_i - x_j)$ plus a constant (for the natural frequency). The result is the system identified as a linear combination of the functions we choose in the form of a matrix Ξ, called the coefficient matrix. This process is illustrated in Figure 1.4.

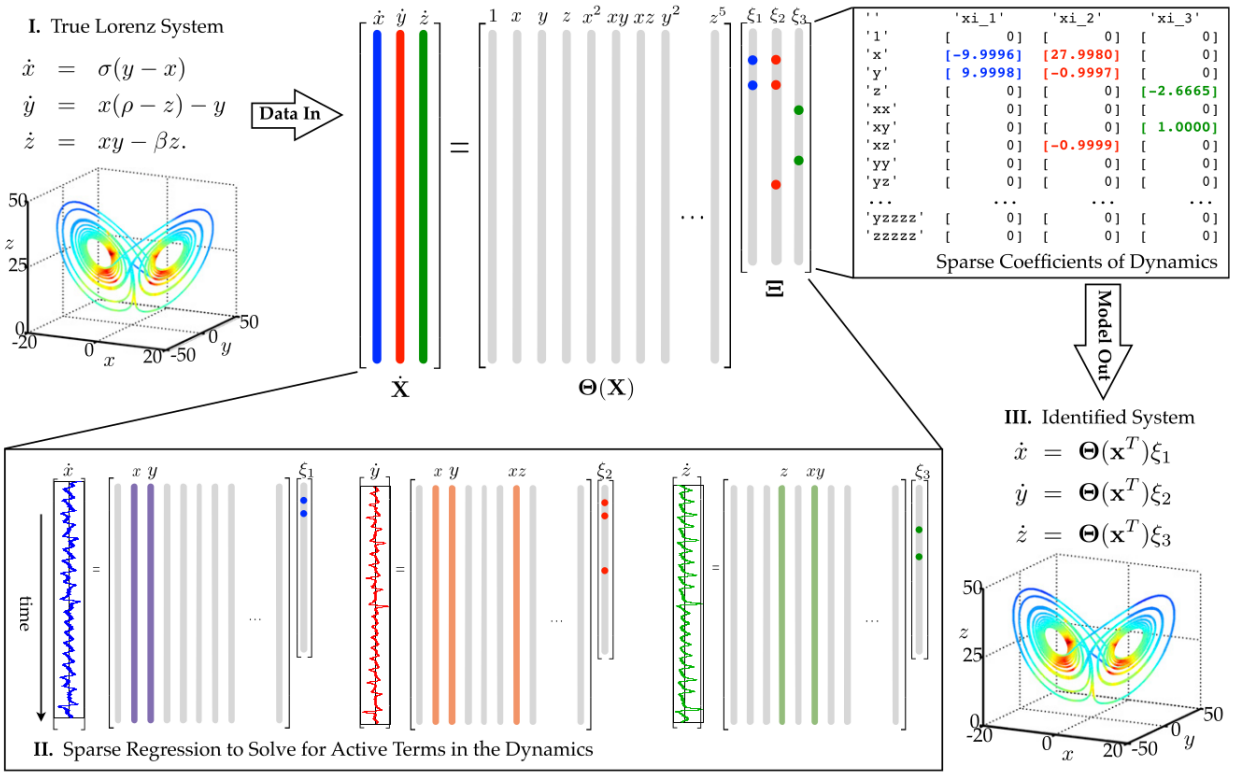


Figure 1.4: "Schematic of the SINDy algorithm, demonstrated on the Lorenz equations. Data are collected from the system, including a time history of the states $\mathbf{x}$ and derivatives $\dot{\mathbf{x}}$; the assumption of having $\dot{\mathbf{x}}$ is relaxed later. Next, a library of nonlinear functions of the states, $\Theta(\mathbf{x})$, is constructed. This nonlinear feature library is used to find the fewest terms needed to satisfy $\dot{\mathbf{x}} = \Theta(\mathbf{x})\boldsymbol{\Xi}$. The few entries in the vectors of $\boldsymbol{\Xi}$, solved by sparse regression, denote the relevant terms in the right-hand side of the dynamics. Parameters values are $\sigma = 10, \beta = 8/3, \rho = 28$, $(x_0, y_0, z_0)^T = (-8, 7, 27)^T$. The trajectory on the Lorenz attractor is colored by the adaptive time step required, with red indicating a smaller time step." Figure and caption from [8]

From this, we can validate the SINDy model by numerically integrating the equations to predict the dynamics of the model in the network, and then compare these predictions to a test time-series. Alternatively, we can confront the coefficient matrix derived from the SINDy model with the original one.

In the following chapter, I will introduce some metrics to assess the quality of this reconstruction process.

For this work, I utilized the PySINDy library [9] [10] as an implementation of SINDy. This library offers excellent support and numerous features. It provides various algorithms to solve the sparse regression problem, including STLSQ, SR3, LASSO, SSR, and Ridge, each with its own characteristics. Although the default optimizer is STLSQ, my focus is primarily on SR3 due to its ability to incorporate additional constraints and perform nonlinear parameter estimation [12]. However, I will also compare how these different algorithms fare against each other.

The default SINDy method, Sequential Thresholded Least Squares (STLSQ) [8], operates by setting a threshold parameter that defines the minimum magnitude of a coefficient in Ξ. The algorithm aims to minimize the objective function

$$\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2^2 + \alpha\|\boldsymbol{\Xi}\|_2^2$$

by iteratively performing a least squares fit and disregarding elements of the weight matrix Ξ that are below the defined threshold. This process of fitting and thresholding continues until convergence is reached or the pre-set maximum number of iterations is met. In particular, this implementation defaults to sequentially thresholded Ridge regression. It also features α, a hyperparameter, which acts as an optional regularization L2 (Ridge) on the weight vector. This signifies that when $\alpha = 0$, the objective function becomes equivalent to ordinary least squares.

The Least Absolute Shrinkage and Selection Operator (LASSO) is a linear regression method that adds an L1 penalty term to the objective function and is implemented in scikit-learn [14]. This method promotes solution sparsity by shrinking some coefficients precisely to zero. The optimization problem for LASSO can be expressed as:

$$\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2^2 + \alpha\|\boldsymbol{\Xi}\|_1^2$$

The Stepwise Sparse Regression (SSR) [13], directly implemented in PySINDy, is a greedy algorithm. These algorithms iteratively truncate model coefficients, seeking the optimal model based on a certain success metric. Numerous strategies exist to decide which terms to truncate; in this case, the implementation used truncates the smallest coefficient at each iteration. The objective function that SSR attempts to minimize by eliminating the smallest coefficient can be written as:

$$\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2^2 + \alpha\|\boldsymbol{\Xi}\|_0^2$$

Ridge regression is a linear regression method that includes an L2 penalty term in the objective function. Implemented in scikit-learn [14], this method helps to prevent overfitting by contracting the coefficients to zero. The optimization problem for Ridge regression can be described as:

$$\min_{\boldsymbol{\Xi}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2^2 + \alpha\|\boldsymbol{\Xi}\|_2^2$$

Here, α is a non-negative regularization parameter that controls the degree of shrinkage. It is worth noting that in the last term, the L2 norm is squared (signified by the superscript '2'), which ensures the penalty term is always positive, contributing to the minimization problem.
SR3 [11] [12], which stands for Sparse Relaxed Regularized Regression, is a method directly implemented in PySINDy. Differing from the previously mentioned algorithms, SR3 introduces an auxiliary variable $\mathbf{W}$ and relaxes the optimization as follows:

$$\min_{\boldsymbol{\Xi},\mathbf{W}} \|\dot{\mathbf{X}} - \boldsymbol{\Theta}(\mathbf{X})\boldsymbol{\Xi}\|_2^2 + \lambda R(\mathbf{W}) + \frac{1}{2\nu}\|\boldsymbol{\Xi} - \mathbf{W}\|_2^2$$

In this equation, $R(\mathbf{W})$ serves as a regularization function, while ν defines the level of relaxation. The threshold determines the intensity of the regularization. When the regularization function R is the L0 norm (as used in my work), the regularization equates to performing hard thresholding, and lambda is chosen to correspond to the threshold value defined by this parameter. This is equivalent to setting $\lambda = \frac{threshold^2}{2\nu}$. Thus, in my case, $R(\mathbf{W}) = |\mathbf{W}|_0$.

The principle behind SR3 involves solving a relaxed version of the regularized regression problem, resulting in a simplified optimization problem that can be solved more efficiently. SR3 boasts several advantages, such as computational efficiency, increased accuracy, faster convergence rates, and greater flexibility, making it a highly effective tool for model discovery in diverse applications.

In this chapter, we explored foundational concepts necessary for understanding the subsequent chapters, beginning with definitions of complex systems, networks, and network models, followed by an examination of the Kuramoto model, a mathematical framework used to describe synchronization. Finally, we looked into the Sparse Identification of Nonlinear Dynamics (SINDy) framework, explaining its functionality and the algorithms it employs to solve sparse regression problems. A complex system consists of numerous interacting components that give rise to emergent behaviors, which are not easily predictable from individual components alone. These systems are typically nonlinear and are best studied through interdisciplinary methods, particularly through network theory. Networks in complex systems are comprised of nodes (representing entities) connected by edges (representing interactions). We discuss various network models, including the Erdős–Rényi (ER), Lattice (L), Watts-Strogatz (WS), Barabási–Albert (BA), Stochastic Block (SBM), and Geometric Random Graph (GRG) models. These models differ in terms of node connectivity and topological features. An adjacency matrix is used to represent these connections, essential for analyzing network structures. The Kuramoto model is introduced to study the synchronization of coupled oscillators within a network. The model uses

differential equations to describe how the phase of each oscillator evolves over time, influenced by the coupling strength and network structure. Synchronization is quantified using the order parameter, which measures the degree to which the oscillators are in phase. The SINDy framework is a data-driven method used to identify the governing equations of dynamical systems from observational data. By collecting time-series data and generating a library of candidate functions, SINDy formulates a sparse regression problem to identify the simplest set of equations that accurately describe the system's dynamics. Various algorithms such as SR3, STLSQ, LASSO, SSR, and Ridge regression are discussed for solving this problem, each with unique characteristics and applications.

This chapter sets the stage for understanding how these concepts are used in reconstructing network structures and dynamics from time-series data, making the foundation for deeper exploration in the following chapters.

Chapter 2

Assessing the quality of the reconstruction

In the preceding chapter, we looked into the concepts of networks and SINDy, its potential for solving inverse problems, and how it operates. This chapter will pivot to evaluating the outputs from SINDy and defining metrics to gauge its performance. To ascertain how accurately SINDy can fit time-series data and to pinpoint the optimal model underlying it, we must establish metrics corresponding to SINDy's outputs. Metrics are quantitative measures employed to evaluate the performance or quality of a model, algorithm, or system. They facilitate the quantification of various aspects of data-related processes, such as prediction accuracy, model interpretability, or classification performance, thereby enabling informed decisions and comparisons of different approaches. SINDy generates two categories of outputs: those associated with network structure and those pertaining to dynamics.

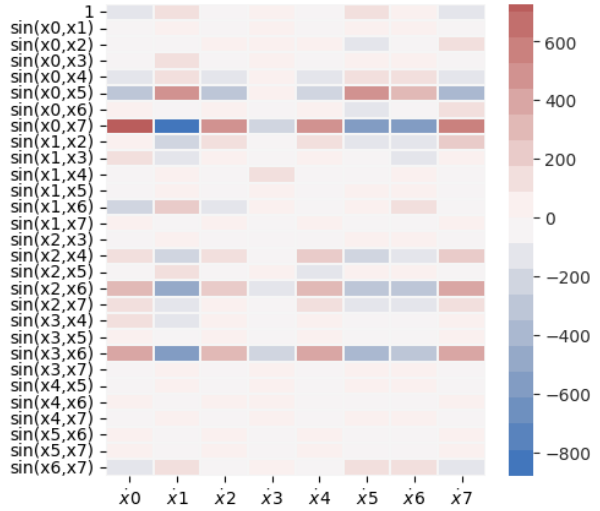
2.1 Network structure

The first output concerning network structure is the coefficient matrix, symbolized by Ξ. This matrix houses non-null coefficients that correspond to the basis functions which best approximate each component of the function $\mathbf{f}$ under the fitted data. The dimensions of the matrix are m $\times$ n, where m represents the number of basis functions, and n is the number of components of $\mathbf{f}$. The coefficient matrix Ξ encompasses all the critical outputs of SINDy.
For the Kuramoto model, however, the coefficient matrix Ξ contains a constant term of 1 and includes all possible $\frac{N\times(N-1)}{2}$ combinations of sine differences for each component of $\mathbf{f}$. This is an issue because the model does not invariably have a relative node and link for every possible combination of terms. Instead, we can only accommodate $\sin(x_k - x_i)\ \forall k \neq i$ for each i-component.
To address this challenge, PySINDy offers a sparse regression method called `ConstrainedSR3` which utilizes the former SR3 algorithm, discussed in the previous chapter under the SINDy section 1.4, with the added advantage of allowing for the direct imposition of constraints on the Ξ matrix. This method fine-tunes the problem by ensuring that the Ξ matrix only contains necessary terms. The application of this method is illustrated in Figures 2.1, which employ the same input data but leverage different algorithms to tackle the sparse regression problem. The left panel of Figure 2.1a presents the result of the `SR3` method, while the right panel of Figure 2.1b showcases the outcome of the `ConstrainedSR3` method. The latter method enforces two types of constraints: zero constraints for the i-entries $\sin(x_j - x_k)\ \forall j, k \neq i$, and the antisymmetry constraints derived from the Kuramoto model, where for the i component we have $\sin(x_j - x_i)$, and for the j component we have $-\sin(x_j - x_i)$. The basis functions yield only one combination of the difference.
As shown in Figure 2.1, the coefficient matrix in the left panel of Fig. 2.1a is filled with numerous non-zero entries that can take very large values. In contrast, the coefficient matrix in the right panel of Fig. 2.1b contains only a few non-zero values that are distinct from each other. This second matrix accurately represents the original model. Unfortunately, the `ConstrainedSR3` method does not scale well as the dimension of the network increases, leading to computational problems even for low-dimensional problems. So I will use other sparse regression methods, especially `SR3`, but at a cost to avoid imposing constraints with the drawback of needing more data to converge [1].

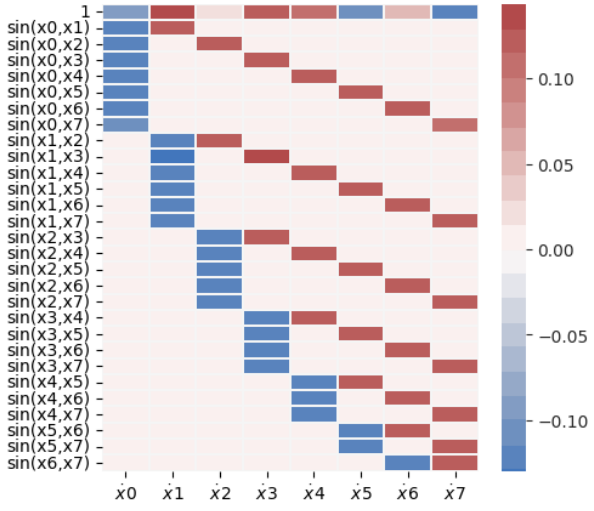
[1] There are different modes to give more data in input to SINDy, by giving multiple time series, by reducing the

Figure 2.1: Example of the use of `ConstrainedSR3` vs `SR3`

(a) Coefficient matrix of a SINDy fit with the SR3 algorithm, applied to a fully-linked network of 8 nodes using the Kuramoto model, without imposing any constraints.



(b) Coefficient matrix of a SINDy fit with the ConstrainedSR3 algorithm. This algorithm is applied to a fully-linked network of 8 nodes using the Kuramoto model, while imposing zero constraints and antisymmetry ones.



Moreover, the coefficient matrix can represent the network structure. Knowing the original coupling value K of the model allows us to extract the matrix entries related to the adjacency matrix, this information can be used to identify the underlying network structure.
So for example if $K = 10$ and the output models gives for a 3 node network:

$$\begin{cases} \dot{x_1} = 1 \times 1 + 10 \sin(x_1 - x_2) \\ \dot{x_2} = 0.4 \times 1 - 10 \sin(x_1 - x_2) + 10 \sin(x_2 - x_3) \\ \dot{x_3} = -0.3 \times 1 - 10 \sin(x_2 - x_3) \end{cases}$$

We have for the relative adjacency matrix:

$$\hat{A}_{SINDy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

I now define the first metric to measure how good the reconstruction of the adjacency matrix is by taking the Frobenius norm of the difference between this reconstructed adjacency matrix and the original adjacency matrix and then normalizing by the Frobenius norm of the original adjacency matrix:

$$\text{metric-frobenius adjacency} := \frac{||\hat{A}_{original} - \hat{A}_{SINDy}||_F}{||\hat{A}_{original}||_F}$$

Where the Frobenius Norm is defined as

$$||A||_F := \sqrt{\sum_i^m \sum_j^n |A_{ij}|^2}$$

It is important to note that this metric takes into account only the important terms of the coefficient matrix related to the network structure, but this may be a bit restrictive for considering how well the SINDy fit has been as a whole, so I then defined another metrics, similar to the one defined before,

sampling interval of the time series or by increasing the maximum time of the time series. It seems that of these three ways to get a better fit from SINDy, the best and simpler way to add more data is by adding more multiple time series.

this time starting from the original model I create the relative SINDy coefficient matrix and then take this matrix and the SINDy coefficient matrix and apply the same operation as before:

$$\text{metric-frobenius coefficients} \coloneqq \frac{||A_{original\ coef.} - A_{SINDy\ coef.}||_F}{||A_{original\ coef.}||_F}$$

This metric takes all the possible coefficient in the model reconstructed by SINDy, this is important because often SINDy returns not only the right coefficients of the dynamics but other ones as seen in Fig. 2.1, so this norm also quantifies those terms.

2.2 Network dynamics

The second type of output can be obtained using a useful PySINDy routine, called `score` which, after fitting the data, predicts the time derivatives using the SINDy model starting from an test time series then it differentiate numerically this time series and confront the two time series derivatives using a determined metric and returns a number according to it. For this analysis, I used the routine `score` with two different types of metrics:

$$\text{score mean} \coloneqq \sqrt{\frac{1}{N}\sum_i^N \frac{\sum_t^{t^{max}}(\dot{y}_i^{test}(t) - \dot{y}_i^{SINDy}(t))^2}{\sum_t^{t^{max}}(\dot{y}_i^{test}(t) - \bar{\dot{y}}_i^{test})^2}}$$

$$\text{score sum} \coloneqq \sqrt{\frac{\sum_i^N\sum_t^{t^{max}}(\dot{y}_i^{test}(t) - \dot{y}_i^{SINDy}(t))^2}{\sum_i^N\sum_t^{t^{max}}(\dot{y}_i^{test}(t) - \bar{\dot{y}}_i^{test})^2}}$$

These metrics are important because they take into account directly the dynamics, in particular the metrics take into account the derivatives and not directly the numerical solution of the differential equation because the resulting SINDy equations are often stiff [2] and not stable, making it very difficult to integrate numerically.
I used another metric, the global order parameter R [7] defined as:

$$R \coloneqq \frac{1}{N}\sum_t^N r_t$$

Where r_t is the order parameter 1.1 calculated at time t. This metric calculated on a certain solution measures the degree of synchronizations between different nodes. In particular, as this metric takes into account directly the solution of the system of ODE, I calculated this metric for both the original equations and the SINDy ones, but for the last ones I calculated it only when the ODEs are more probable to be easy to be numerically integrated and it is set to -1 otherwise. I decided to integrate the SINDy differential equations numerically only when the maximum entries of the reconstructed SINDy adjacency matrix is less than 2 and greater than -1, as I have seen that considering this range, where at maximum the entries distance 2 from the right value (1 if there is a link or 0 if there is none), the number of stiff equations to integrate is much limited. This is done to reduce the time execution of the program in particular when applied to very diverse coupling values in the region of values where there are problems with the SINDy fit, as will be seen in the next chapter.

In summary, in this chapter we focused on evaluating the output of the Sparse Identification of Nonlinear Dynamics (SINDy) algorithm and defining metrics to assess its performance. Specifically,

[2] In the context of ordinary differential equations (ODEs), a stiff equation is one that has solutions that vary rapidly on a small scale and slowly on a larger scale. More precisely, a stiff ODE is one in which there is a large disparity in the characteristic timescales of the solution components, with some components varying much more rapidly than others. Stiff ODEs can be difficult to solve numerically because traditional ODE solvers, such as explicit Runge-Kutta methods, may require very small step sizes to accurately capture the rapid variation of the solution components. This can make the numerical solution of stiff ODEs computationally expensive and time consuming. Moreover, traditional ODE solvers may be unstable or inaccurate when applied to stiff problems.

it examined how well SINDy can fit time-series data and identified the optimal underlying model by establishing relevant metrics for its outputs, including the network structure and dynamics.

In the Network Structure section, the chapter introduces the coefficient matrix Ξ, which represents the network's structure by indicating the non-zero coefficients associated with the basis functions that best approximate each component of the function f. The chapter discusses challenges, such as the overabundance of possible combinations of sine differences in the Kuramoto model and how the `ConstrainedSR3` method can address these by imposing constraints on the Ξ matrix. This method ensures that the matrix contains only the necessary terms, although it struggles with scalability as the dimensions of the network increase. To assess the accuracy of the network reconstruction, two metrics are defined: one based on the Frobenius norm of the adjacency matrix difference and another on the difference between the SINDy and original coefficient matrices.

Chapter 3

Results

In the previous chapter, we explored the outputs of SINDy and discussed various metrics to quantify its performance on a given network. In this chapter, we present results from three types of analysis. For the first two analyses, we considered networks with 32 nodes. Using smaller networks was necessary to conduct a comprehensive analysis since SINDy's fitting time does not scale well as the number of nodes increases as we will see at the end of this chapter. Obtaining a sufficient amount of data for statistical analysis becomes challenging with larger networks.

The first analysis involved fixing a low average degree (~5 connections per node) network of the ones in Figure 1.3 while varying the coupling strength. We performed 30 SINDy fits, each with different time series or trajectories generated from distinct initial conditions. This approach allowed us to obtain reliable statistical results.

The second analysis focused on using different networks with varying average degrees while keeping the coupling strength constant. This analysis provided insights into the influence of the network structure, specifically the average degree, on the SINDy results. By examining multiple networks, we could observe how changes in average degree affected the performance of SINDy.

The third analysis explores the computational limits of SINDy, in particular by measuring the time and the RAM SINDy takes to perform a fit as the number of nodes changes.

Unless explicitly stated, the default sparse regression algorithm used in the analysis was `SR3`. This choice was based on its effectiveness in capturing the underlying dynamics while promoting sparsity. Throughout this chapter, we will present the findings and discuss the implications of the results obtained from these two analyses.

3.1 Diverse coupling

The analysis of varying coupling strength in different network topologies reveals some important information. The metrics used to evaluate SINDy's performance show worse results in both the initial area when the coupling is small and when the coupling is large. This can be observed in Figures 3.1a, 3.1b, 3.1c, and 3.1d.

Figures 3.1e and 3.1f provide additional information by comparing the original global order parameter R of the test time series and the global order parameter of the SINDy reconstruction as the coupling varies. It can be observed that after a certain coupling value, the reconstruction becomes impossible due to stiff equations. In particular, when the value of the reconstructed global order parameter R is set to -1, the reconstruction is incorrect. This explains the previous results in Figures 3.1a, 3.1b, 3.1c, 3.1d, obtained in other metrics in the region where there is a spike.
As the coupling increases, the global order parameter R approaches 1, indicating faster synchronization between different nodes. However, from SINDy's perspective, the useful data are those when there is no full synchronization, allowing the model to capture the different dynamics of each node. When full synchronization occurs, SINDy only sees all nodes oscillating with the same frequency and perfect synchronization, making it impossible to reconstruct the individual dynamics.

Therefore, SINDy performs worse at higher coupling values because of the velocity of the synchronization process. The greater the coupling, the faster the synchronization occurs. However, from

SINDy's perspective, reconstruction is only possible when there is partial synchronization, enabling the differentiation of dynamics among nodes.

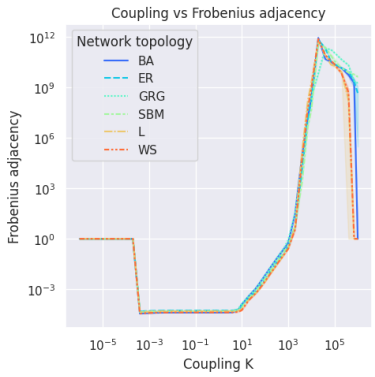
The analysis of different network topologies in Figure 3.1 shows that there are no significant differences in the results obtained, except for some minor variations in the high-coupling region. This suggests that the impact of coupling strength on SINDy's performance is consistent across different network topologies.

Now we see why when the coupling is small in Figures 3.1a, 3.1c and 3.1d the metrics used have a flat value around 1. In particular, the problem becomes apparent in Figure 3.3, where it shows an analysis of the hyperparameter threshold (fixed to 0.00001 in Figure 3.1) and how it affects the behavior of the model `SR3`. The threshold determines the minimum value at which the coefficients in the matrix Ξ are eliminated. So when the coupling or angular velocity ($\frac{K}{N}$ or ω) falls below the threshold, the coefficients are set to zero, resulting in a Frobenius adjacency metric value or a score of 1. However, the Frobenius coefficients metric behaves differently because it includes all contributions from the basis functions, including ω, as seen for higher thresholds.

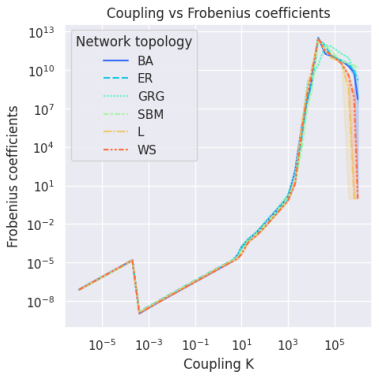
It is important to note that not always all coefficients below the threshold are emptied, as demonstrated in Figure 3.2a. This analysis in Figure 3.2 focuses on a fixed network topology (BA) and examines how the metrics change with a varying number of trajectories in the input data. The results show that higher numbers of trajectories lead to better results, especially in the low-coupling region. However, even with more data, the metrics reach a value of 1 for lower coupling values only when there are larger numbers of trajectories. This explain the problem and it is because the sparse algorithm used, `SR3`, does not always converge when there is a small amount of data.

Figure 3.1: Metric measures as a function of coupling strength K. Different colors refer to different network topologies and different panels show results for different metrics. Results are similar across different network topologies. Random colors.

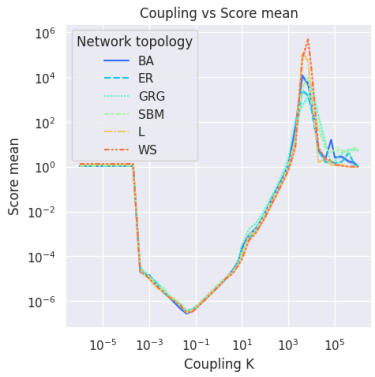
(a) Frobenius adjacency metric.



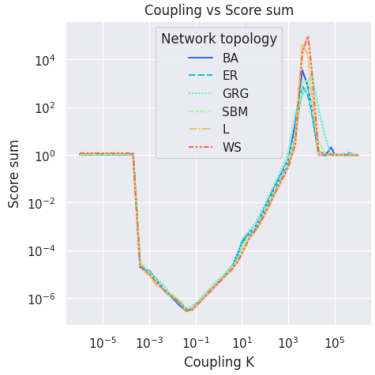
(b) Frobenius coefficient metric.



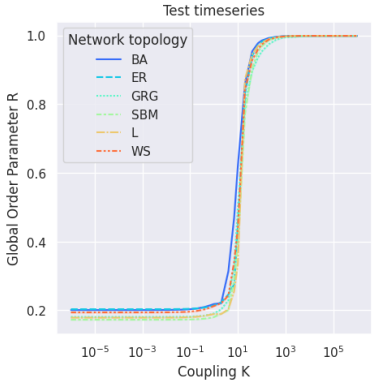
(c) Score mean metric.



(d) Score sum metric.



(e) Global order parameter R for a test time series.



(f) Same of panel (e) but with time series reconstructed by SINDy. For high couplings ($K > 10^3$), R gets a value of 1 and the time series can not be reconstructed.

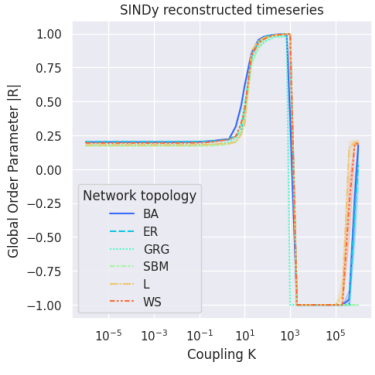
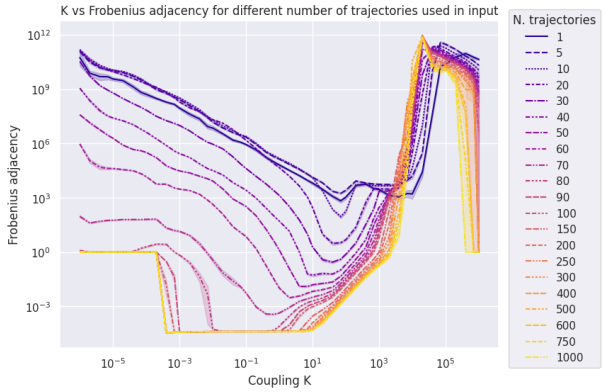
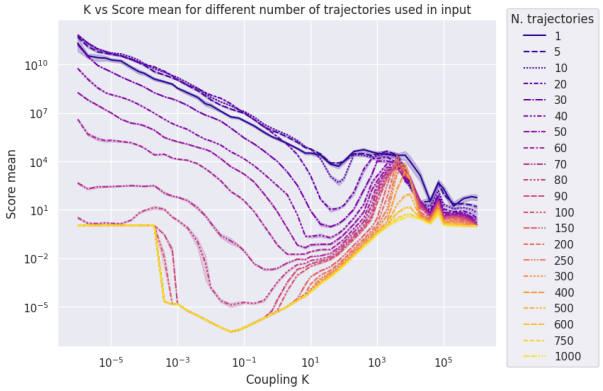


Figure 3.2: Metric measures as a function of coupling strength K. Different colors refer to different number of trajectories used in input and different panels show results for different metrics. Results indicate increasingly better results the more trajectories are in input up to a certain limit. The color scale used for the lines in the graph follows a palette where the colors transition from violet for lower values to yellow for higher values of the number of trajectories used.

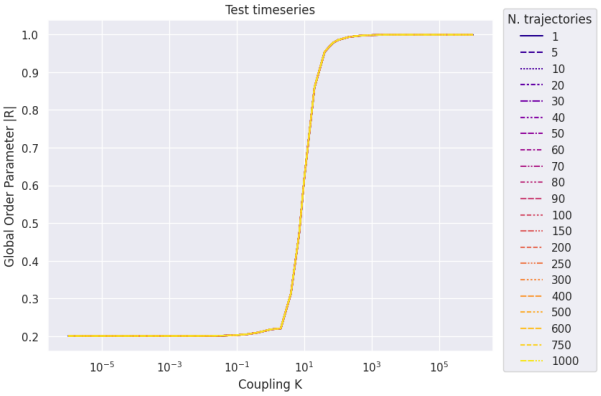
(a) Frobenius adjacency metric.



(b) Score mean metric.



(c) Global order parameter R for a test time series.



(d) Same of panel (c) but with time series reconstructed by SINDy. For higher trajectories the reconstruction is better because R gets a value of -1 for higher values of the coupling.

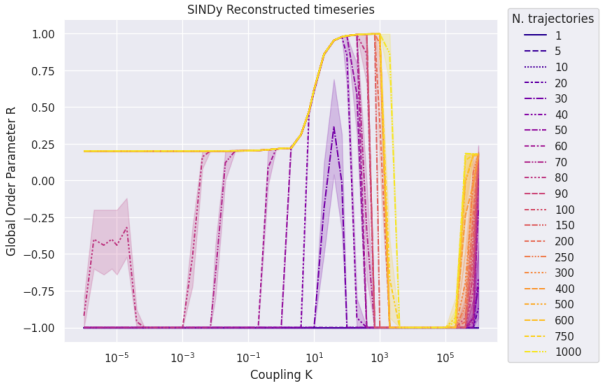
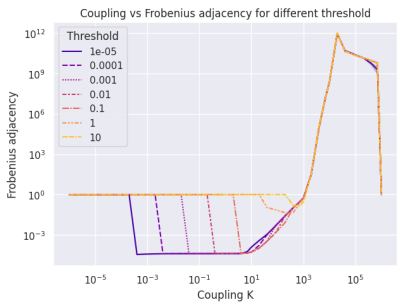
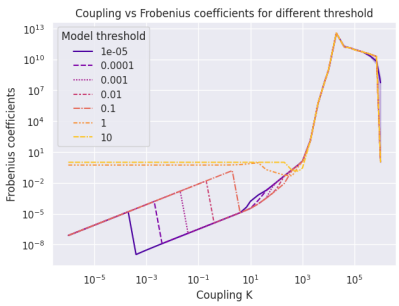


Figure 3.3: Metric measures as a function of coupling strenght K. Different colors refers to different values of the hyperparameter threshold in the `SR3` sparse regression model. The color scale used for the lines in the graph follows a palette where the colors transition from yellow for lower values to violet for higher values of the threshold.

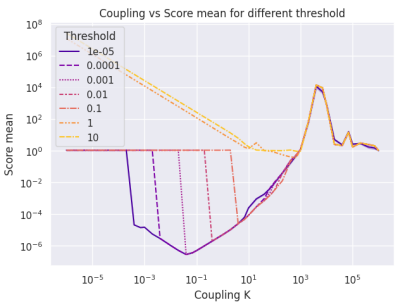
(a) Frobenius adjacency metric.



(b) Frobenius coefficient metric.



(c) Score mean metric.

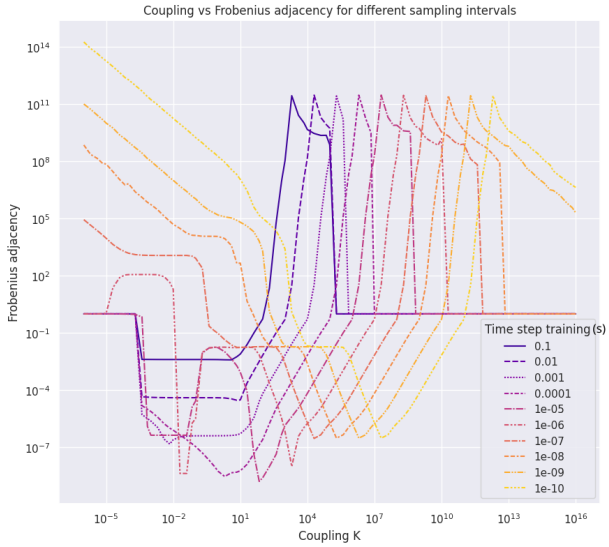


Then I investigated how the sampling interval used to determine the time series affects the results while keeping the number of data points constant. The results are depicted in Figure 3.4. Particularly, for lower intervals, the optimal fitting window shifts towards higher coupling values. However, a similar pattern is observed at the end of the curve. This observation aligns with the notion that, as the time-step interval decreases, larger coupling values are required to achieve full synchronization. In Figure

3.5, we can see that scaling the coupling by multiplying it by the time step yields the same trend in the end. This implies that despite using smaller time steps in an attempt to improve the results for high coupling values, there is a synchronization limit that cannot be overcome.

Figure 3.4: Metric measures as a function of coupling strength K. Different colors refer to different sampling interval of the time series in input. Results show a similar pattern but shifted. The color scale used for the lines in the graph follows a palette where the colors transition from yellow for lower values to violet for higher values of the time step training.

(a) Frobenius adjacency metric.



(b) Score mean metric.

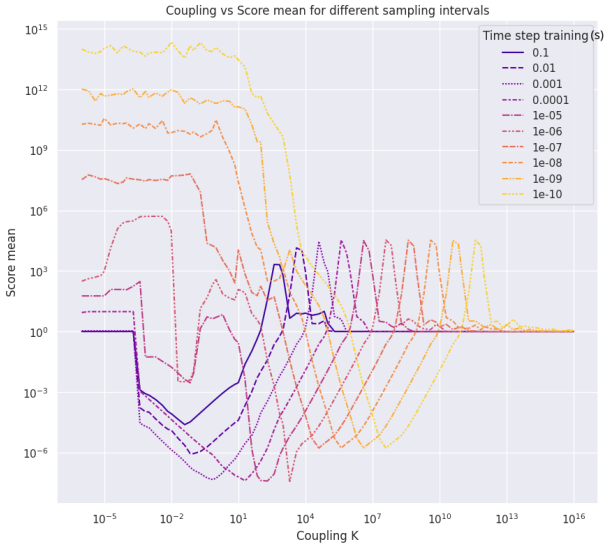
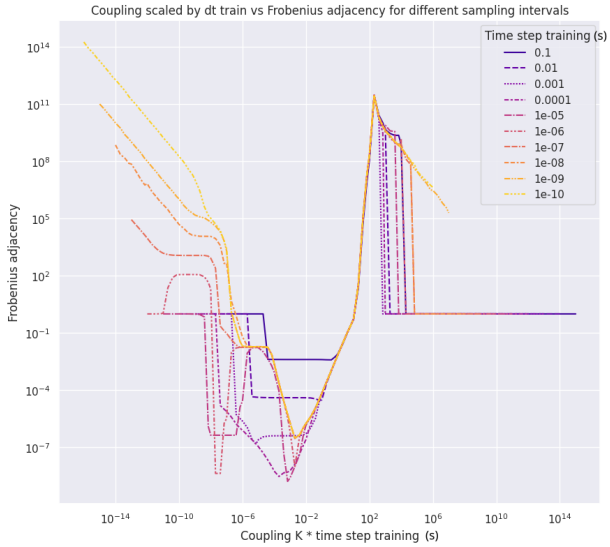
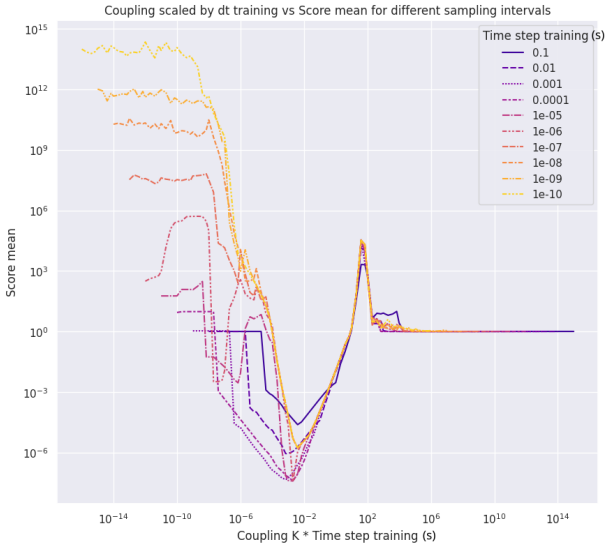


Figure 3.5: Metric measures as a function of coupling strength K times the sampling interval in input. Different colors refer to different sampling interval of the time series in input. Results demonstrates that scaling the coupling on the abscissa by the sampling interval in Figure 3.4 results in the same trend figure at the end of the abscissa. The color scale used for the lines in the graph follows a palette where the colors transition from yellow for lower values to violet for higher values of the time step training.

(a) Frobenius adjacency metric.



(b) Score mean metric.



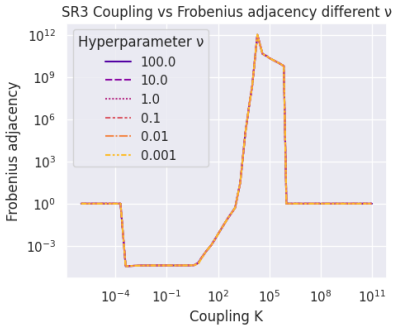
I then wanted to explore how different sparse regression algorithms behave and determine if they all perform poorly at high coupling values, in order to verify if the issue lies with the algorithms themselves. In addition, I wanted to investigate whether the hyperparameters used can influence the

results. For this analysis, I focussed on the `SR3` and `STLSQ` algorithms, as they are commonly used with SINDy.

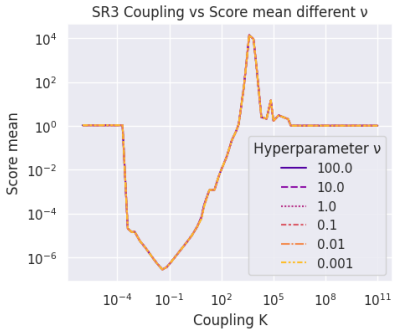
I have already demonstrated the impact of the threshold parameter in `SR3` in Figure 3.3, where it played a crucial role in the truncation of small coefficients in the coefficient matrix. However, when considering the hyperparameter ν, the results are as follows in Figure 3.6. It is evident that varying the value of ν does not have a significant impact on the results.

Figure 3.6: Metric measures as a function of coupling strength K. Results are similar across different hyperparameter ν values of the `SR3` algorithm. Hyperparameter threshold fixed to 0.00001. The lines in the graph are color-coded using a palette where the colors transition from yellow for lower values to violet for higher values of the hyperparameter ν.

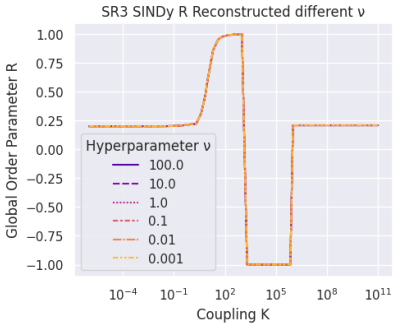
(a) Frobenius adjacency metric.



(b) Score mean metric.



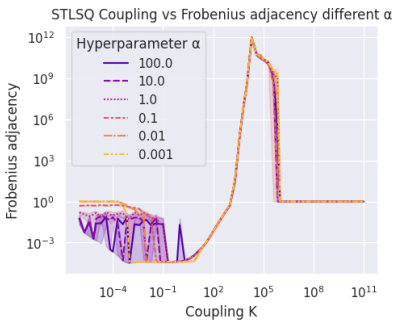
(c) SINDy reconstructed global order parameter R metric.



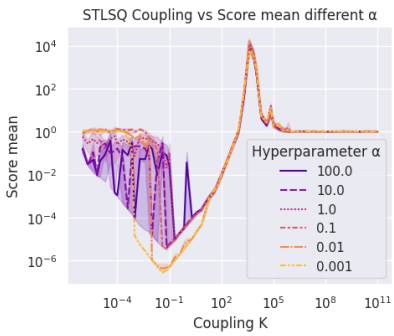
When examining the impact of the hyperparameter α on the `STLSQ` algorithm, I obtained the results depicted in Figure 3.7. It is evident that, while the low-coupling region exhibits changes, the high-coupling area remains unaffected. Consequently, modifying the hyperparameter α does not enhance the reconstruction in the region characterized by strong synchronization.

Figure 3.7: Metric measures as a function of coupling strength K. The results are different for various hyperparameters α of the `STLSQ` algorithm for low coupling values ($10^{-6} < K < 10$) and similar for higher values. The color scale used for the lines in the graph follows a palette in which the colors transition from yellow for lower values to violet for higher values of the hyperparameter.

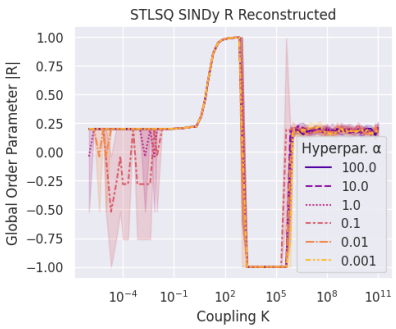
(a) Frobenius adjacency metric.



(b) Score mean metric.



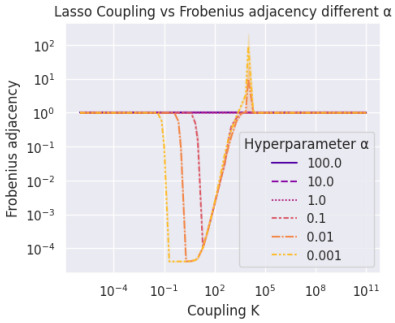
(c) SINDy reconstructed global order parameter R metric.



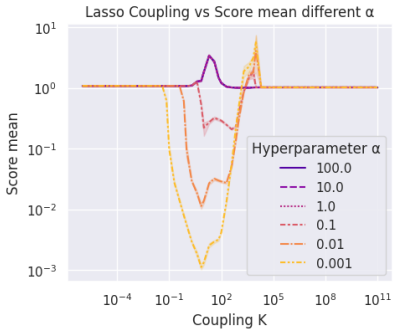
Using now the LASSO algorithm and changing the hyperparameter α I obtained the following results, with better results on the low coupling area for the lower parameter.

Figure 3.8: Metric measures as a function of coupling strength K. Different colors refer to different values of the hyperparameter α of the LASSO algorithm. The results are different for $10^{-1} < K < 10^4$. The color scale used for the lines in the graph follows a palette in which the colors transition from yellow for the lower values to violet for higher values of the hyperparameter α.

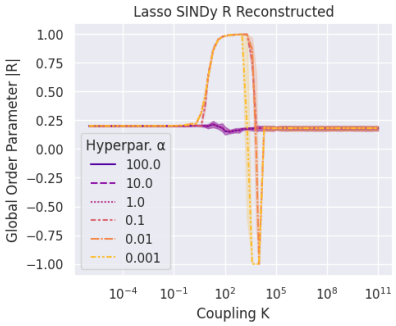
(a) Frobenius adjacency metric.



(b) Score mean metric.



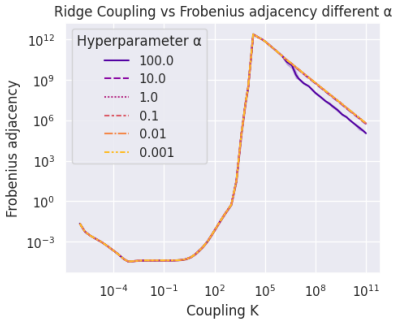
(c) SINDy reconstructed global order parameter R metric.



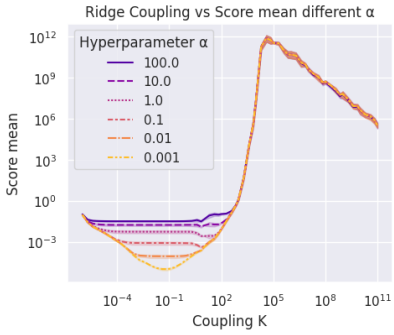
Instead, for the Ridge method for changing α I obtained better results in the score mean for lower values of the parameter in the low coupling area:

Figure 3.9: Metric measures as a function of coupling strength K. Different colors refer to different values of the hyperparameter α for the Ridge algorithm. The color scale used for the lines in the graph follows a palette in which the colors transition from yellow for the lower values to violet for the higher values of the hyperparameter α.

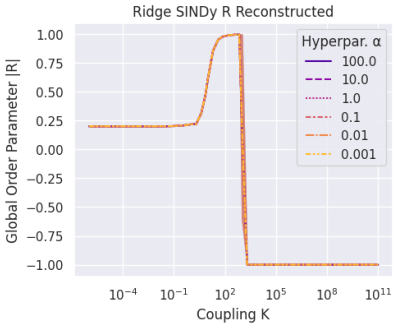
(a) Frobenius adjacency metric.



(b) Score mean metric.



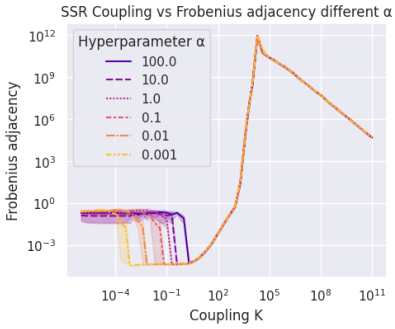
(c) SINDy reconstructed global order parameter R metric



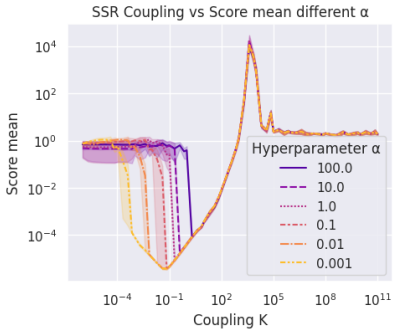
Lastly, for the `SSR` algorithm while changing α I obtained the following results, with better results in the low coupling area for lower values of the parameter:

Figure 3.10: Metric measures as a function of coupling strength K. Different colors refer to different values of the hyperparameter α for the `SSR` algorithm. The color scale used for the lines in the graph follows a palette where the colors transition from yellow for lower values to violet for higher values of the hyperparameter α.

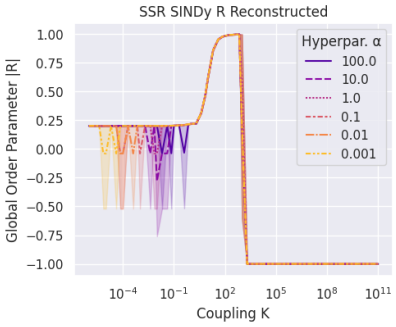
(a) Frobenius adjacency metric.



(b) Score mean metric.



(c) SINDy reconstructed global order parameter R metric.

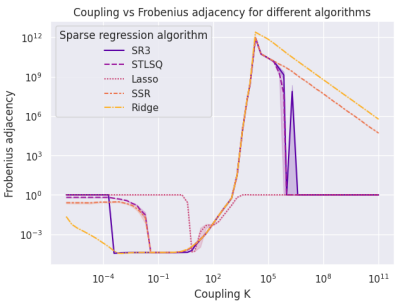


It is important to note that for every algorithm used, no choice of hyperparameter helped the reconstruction at high coupling when the global order parameter of the original time series is very close to one. This means that it is an independent phenomenon from the algorithm used; this strengthens the hypothesis that the reconstruction at high coupling becomes impossible to do because of the velocity of the synchronization which leaves SINDy with not enough data to reconstruct the dynamics.

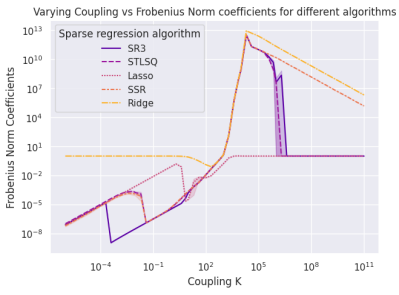
Now we can see a comparison between the use of different algorithms to approach the sparse regression problem. In particular, it can be seen from the R reconstruction in Fig. 3.11f and from all the other metrics that every algorithm behaves poorly at high coupling values.

Figure 3.11: Metric measures as a function of coupling strength K for different sparse regression algorithms used. Different colors refer to different sparse regression algorithm used. Fixed hyperparameter: `LASSO`'s $\alpha = 0.05$, `STLSQ`'s $\alpha = 0.05$, `Ridge`'s $\alpha = 0.05$, `SSR`'s $\alpha = 0.05$ and `SR3`'s $\nu = 1.0$, threshold= 0.00001.

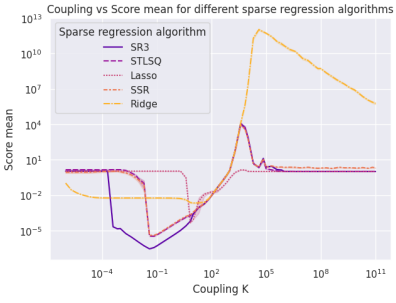
(a) Frobenius adjacency metric.



(b) Frobenius norm coefficients metric.

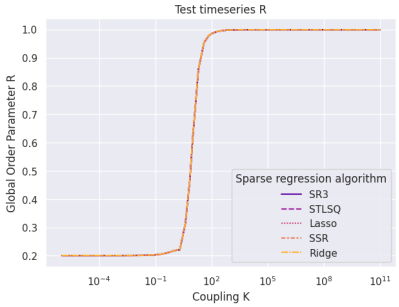


(c) Score mean metric.

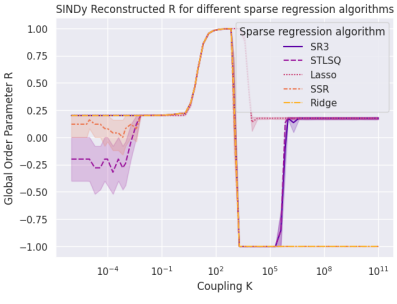


(d) Score sum metric.

(e) Global order parameter for the original test time series used.



(f) Same of panel (e) but with time series reconstructed by SINDy.

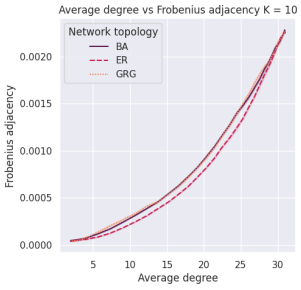


3.2 Different average degree

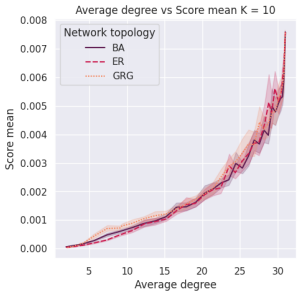
After examining how a model parameter, the coupling, can influence the SINDy reconstruction, in this section I will show the results for the effect of varying instead a parameter intrinsic in the network, the average degree, which means that I will use many different networks while keeping every other parameter constant, the time step of the time series in input fixed to 0.01s, the time maximum of the time series to 1s and using it as a method for solving the sparse regression problem `SR3`. In particular, I examine the effect of using different topologies, BA, ER and GRG for different fixed coupling K. but as we can see from the Figures 3.12, 3.13 for different metrics the results do not change significantly for K = 10, 100, 1000 by the use of a different topology; instead for K = 10000 there are substantial changes in the Frobenius adjacency. The thing to note is that for K = 100 and K = 1000, the SINDy reconstruction is possible only for a low average degree, as a higher average degree encourages synchronization.

Figure 3.12: Metric measures as a function of average degree of the network. The coupling strength K is fixed to 10. Different panels refer to different metrics. Different colors refer to different network topologies.

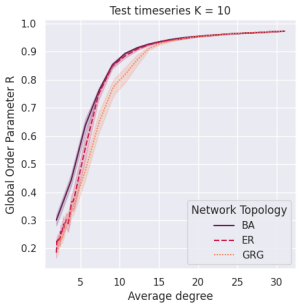
(a) Frobenius adjacency metric.



(b) Score mean metric.



(c) Global order parameter for the original test time series used.



(d) Same of panel (c) but with time series reconstructed by SINDy.

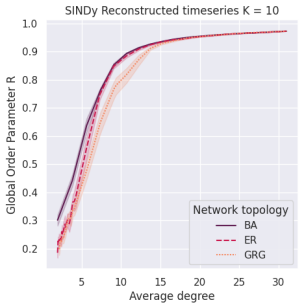
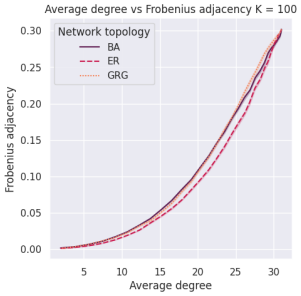
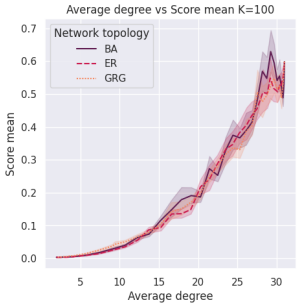


Figure 3.13: Metric measures as a function of average degree of the network. The coupling strength K is fixed to 100. Different panels refer to different metrics. Different colors refer to different network topologies.

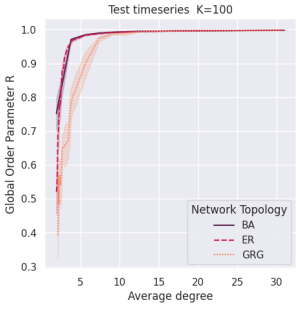
(a) Frobenius adjacency metric.



(b) Score mean metric.



(c) Global order parameter for the original test time series used.



(d) Same of panel (c) but with time series reconstructed by SINDy.

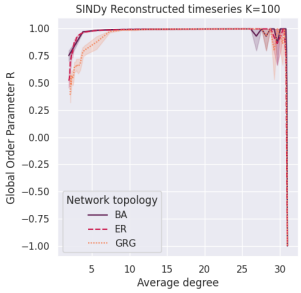
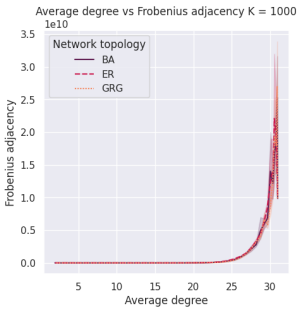
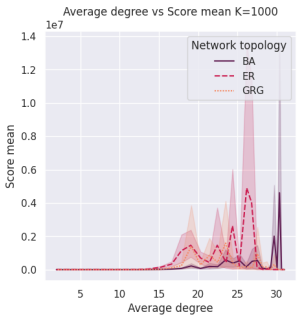


Figure 3.14: Metric measures as a function of average degree of the network. The coupling strength K is fixed to 1000. Different panels refer to different metrics. Different colors refer to different network topologies.

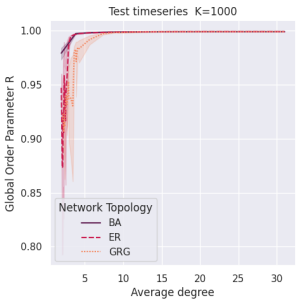
(a) Frobenius adjacency metric.



(b) Score mean metric.



(c) Global order parameter for the original test time series used.



(d) Same of panel (c) but with time series reconstructed by SINDy.

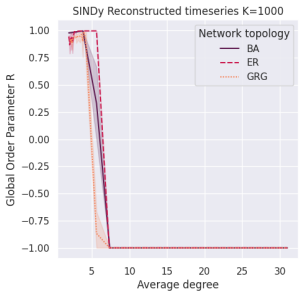
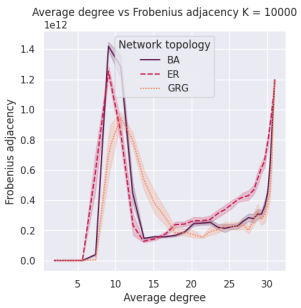
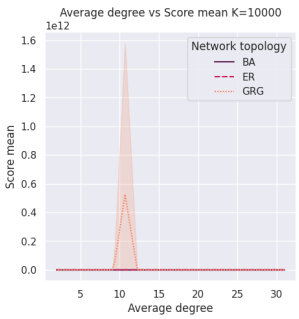


Figure 3.15: Metric measures as a function of average degree of the network. The coupling strength K is fixed to 10000. Different panels refer to different metrics. Different colors refer to different network topologies.

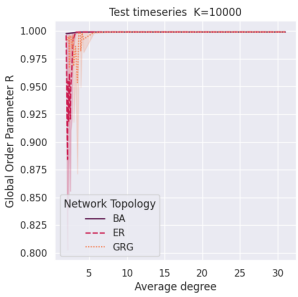
(a) Frobenius adjacency metric.



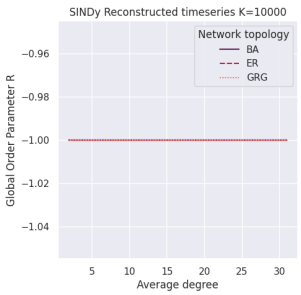
(b) Score mean metric.



(c) Global order parameter for the original test time series used.



(d) Same of panel (c) but with time series reconstructed by SINDy.

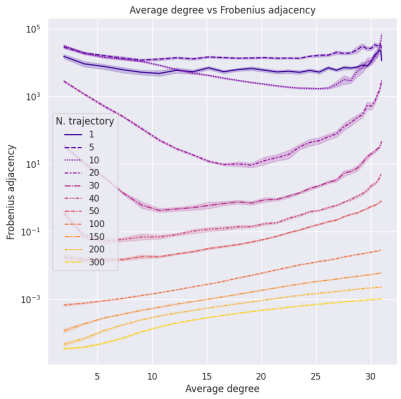


Upon examination of the Figures 3.16, where the BA topology is used with a fixed coupling strength $K = 10$ while varying the number of different trajectories in the input, we can observe the impact of the number of trajectories on the results. Notably, it becomes apparent that to obtain a satisfactory fit across the entire range of average degrees, which entails having both the Frobenius adjacency and score values below 1, a minimum of 100 trajectories in the input is required.

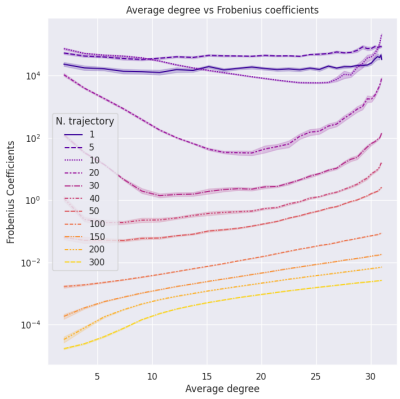
Specifically, in Figure 3.16f, which depicts the SINDy reconstruction of the global order parameter seen in Figure 3.16e, we can observe that a better reconstruction is achieved with an increased number of trajectories. This is particularly evident in cases with higher average degrees, where synchronization poses challenges. As discussed earlier, higher average degrees result in larger global order parameters, making the reconstruction more difficult. Thus, if there are insufficient data points in the form of different trajectories, reconstruction issues arise.

Figure 3.16: Metric measures as a function of average degree of the network. The coupling strength K is fixed to 10. The network topology used is BA. Different colors refers to different number of trajectories used in input. The color scale used for the lines in the graph follows a palette where the colors transition from violet for lower values to yellow for higher values of the the number of trajectory used.

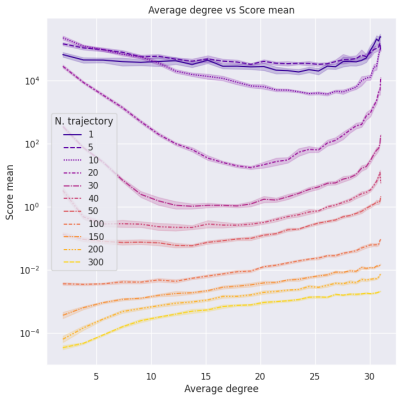
(a) Frobenius adjacency metric.



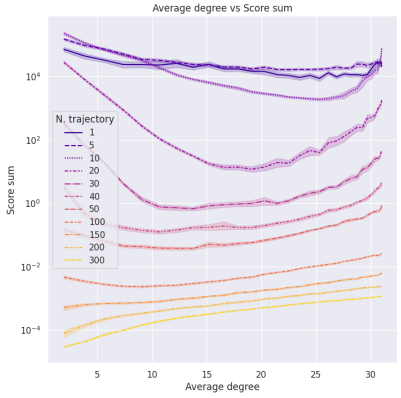
(b) Frobenius coefficient metric.



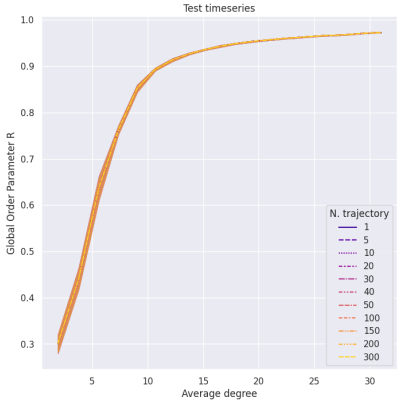
(c) Score mean metric.



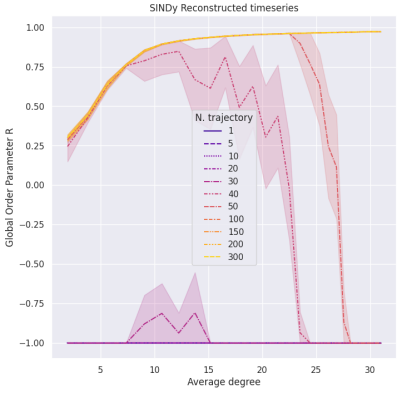
(d) Score sum metric.



(e) Global order parameter for the original test time series used.



(f) Same of panel (e) but with time series reconstructed by SINDy.



3.3 Complexity analysis and computational limits

In this section, I will explore the computational limits of SINDy, specifically focusing on the RAM memory usage and the time required for fitting as the number of nodes increases. To determine the appropriate number of trajectories to use, we employed a method that evaluates the convergence of the algorithms by comparing the Frobenius adjacency matrix reconstructed from SINDy's output with the original adjacency matrix. To accomplish this, I used the confusion matrix, which takes two arrays as input. To ensure compatibility, I transformed both matrices into arrays by "flattening" them.

The comparison process involves sequentially examining each entry and checking if they are equal or not. Since the reconstructed matrix may contain values other than 1 and 0, I introduced a threshold set at 10%. If a value differs from 1 or 0 by more than this threshold, it is replaced by 1 or 0 accordingly. Otherwise, it is set to -1. By constructing the confusion matrix, I can identify any non-coinciding values. If there are non-coinciding values, it indicates that the fit is not ideal and requires more trajectories as input to achieve better results. On the other hand, if all values coincide, indicating the presence of only true 1s and true 0s in the confusion matrix, then the minimum number of trajectories that produces this outcome is determined.

To summarize the algorithm applied:

- Obtain the original adjacency matrix representing the actual classification and the reconstructed matrix representing the predicted classification.
- Transform both matrices into vectors by flattening them.
- Create a new matrix from the reconstructed matrix, setting values to 1 or 0 if they differ less

from 1 or 0 by the threshold or set -1 otherwise.
- Create the confusion matrix.
- If any values do not coincide in the confusion matrix, increase the number of trajectories in the input.
- If all values coincide, minimize the number of trajectories required to achieve this output.
- Once the minimum number of trajectories is determined, record the results, including the RAM usage and the time required for fitting.

By following this algorithm, I can systematically evaluate the computational performance of SINDy, considering both the usage of RAM memory and the fitting time based on the varying number of nodes.

In particular, for this method, all parameters were fixed except for the number of trajectories in the input. Specifically, I set $t_{max} = 1$, $dt = 0.01$, and $K = 10$. Then I applied this method to different sparse regression algorithms to observe the differences and identify the best-performing algorithm. In particular, I studied the `SR3` case, as it was the most widely used.

The results obtained with this method for RAM usage, shown in Figure 3.17, while varying the number of nodes, indicate that all algorithms perform similarly, except for `Lasso`. Furthermore, by fitting the `SR3` data with the variable N^4 in Figure 3.18a, I achieved a good linear interpolation with a r^2 value of 0.999. Hence, the RAM usage for the `SR3` method scales as N^4. Looking at the residuals in Figure 3.18b it confirms that the data scale as N^4 linearly.

Figure 3.17: Comparison across RAM usage of different algorithms, as a function of number of nodes of underlying network. Results show that every algorithm except `Lasso` performs the same. Different colors for different algorithm used.

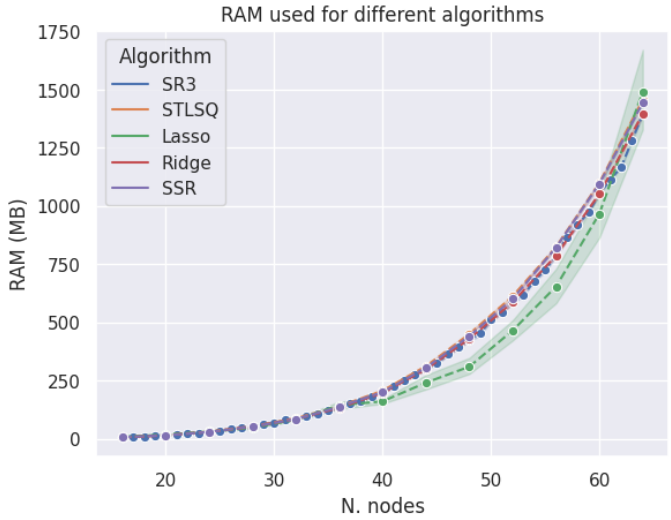
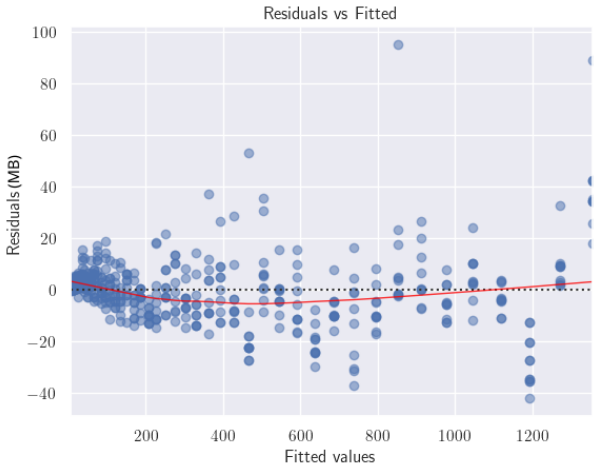


Figure 3.18: Fit and residual plot of the memory RAM used with the SR3 algorithm as a function of number of nodes of underlying network.

(a) RAM usage as a function of the number of nodes to the fourth.

(b) Residual plot of panel (a). The horizontal red line is an indicator that the residual has a linear pattern.



Instead, by looking at the time necessary for a fit in Figure 3.19 we can clearly see that the algorithm Lasso performs the best while the algorithm STLSQ performs the worst. Using the SR3 data with the variable N^6, we can see in Figure 3.20a that the data is approximately linear with $r^2 = 0.999$ with a non-linear trend, especially in the initial area. Looking at the residuals in Figure 3.20b we can clearly see that they do not follow a linear pattern, so we can say that the data scale as N^6 only in the first approximation.

Figure 3.19: Comparison across the time necessary for SINDy to do a fit of different algorithms, as a function of the number of nodes of the underlying network. Results show that Lasso performs the best while SSR performs the worst. Different colors for different algorithm used.

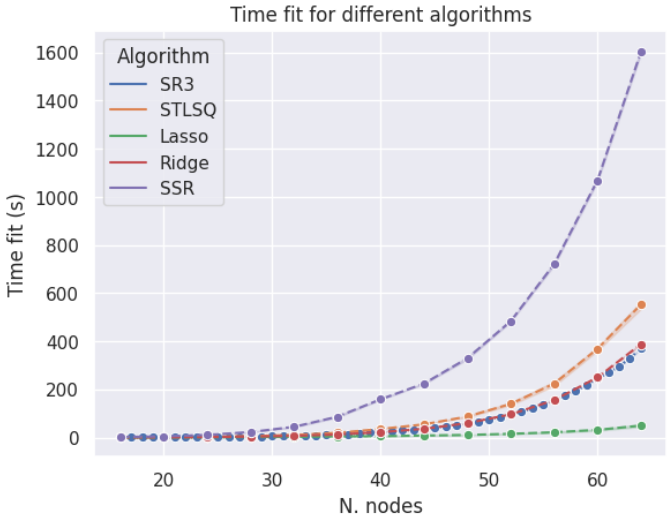
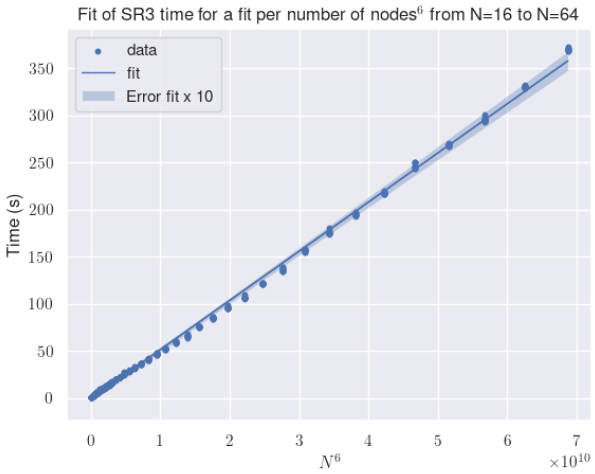
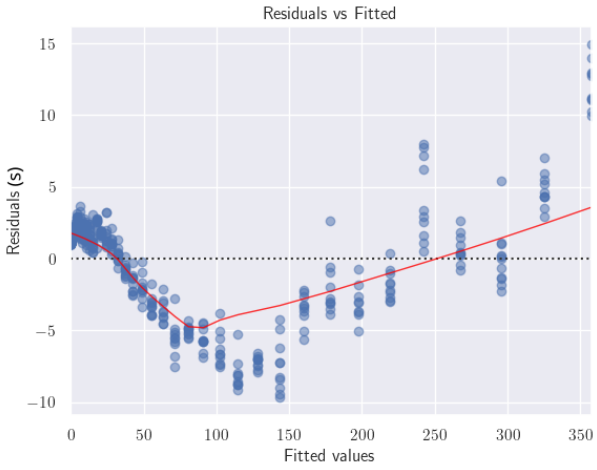


Figure 3.20: Fit and residual plot of the time necessary for a SINDy fit with the `SR3` algorithm for different number of nodes of the underlying network used.

(a) Fit of the time necessary for the SINDy fit in function of N^6.



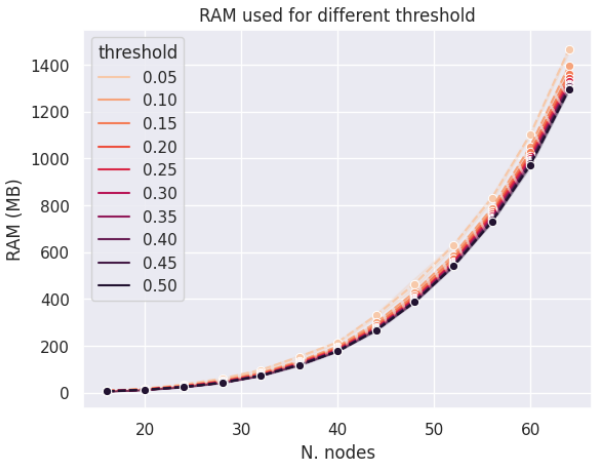
(b) Residuals plot of panel (a). Horizontal red line is an indicator that the residual does not have a linear pattern.



To see that the algorithm used is consistent for different choices of the initial threshold for the confusion matrix, I obtained other results for the RAM used and time necessary for a SINDy fit obtaining the following results in Figure 3.21 for the RAM usage where we can see that there is little variance. Instead in 3.22 we can see that even for the time for the SINDy fit there is little to no variance. This confirms that the method used is stable.

Figure 3.21: Comparison across RAM usage of different threshold used, as a function of the number of nodes of the underlying network used. The color scale used for the lines in the graph follows a palette where the colors transition from pink for lower values to black for higher values of the threshold used in the algorithm.

(a) RAM usage in function of the number of nodes for different thresholds.



(b) Fit of the memory RAM usage in function of the number of nodes to the fourth for different thresholds.

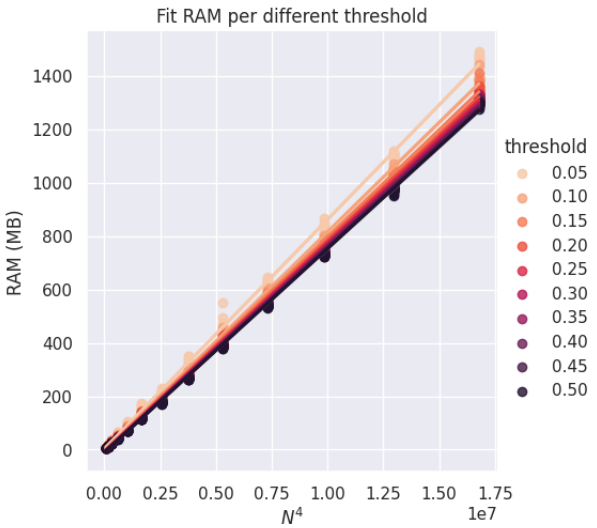
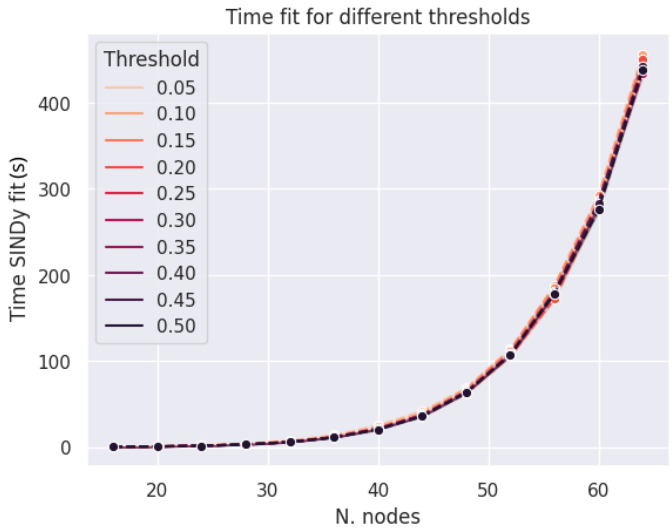


Figure 3.22: Comparison across different time necessary for SINDy to do a fit for different threshold, as a function of the number of nodes. Different colors for different thresholds. The color scale used for the lines in the graph follows a palette where the colors transition from pink for lower values to black for higher values of the the threshold used in the algorithm.



In this chapter, the performance of the SINDy (Sparse Identification of Nonlinear Dynamics) algorithm has been analyzed using three approaches: varying the coupling strength in a fixed network, varying network structures with constant coupling strength, and analyzing the performance of SINDy while varying the number of nodes of the underlying network by measuring the time necessary for doing a fit and by measuring the RAM used during the fit. In the first two approaches, only networks with 32 nodes were used due to the computational challenges that SINDy faces with larger networks.

The first analysis varied the coupling strength within a low average degree network (~5 connections per node), performing 30 SINDy fits with different initial conditions. The results indicate that SINDy performs poorly at low and high coupling strengths due to full synchronization, which prevents SINDy from differentiating the dynamics of individual nodes. The effect of different hyperparameters on the sparse regression algorithms used by SINDy, including `SR3`, `STLSQ`, `LASSO`, `Ridge`, and `SSR`, was also explored, showing that no hyperparameter settings could improve performance at high coupling strengths. In addition, the impact of different sampling intervals on time series while varying the coupling strength has been analyzed, and the results show that the Frobenius adjacency metric and the score mean metric have a similar pattern but shifted for high values of the coupling strength ($K > 10^3$). By scaling the coupling in the abscissa by the sampling interval, we get the same trend at the end of the abscissa analyzed.

In the second section, the impact of varying the average degree, an intrinsic network parameter, on the SINDy reconstruction is analyzed while keeping other parameters constant, such as the time step, the time maximum, and the use of `SR3` to solve sparse regression problems. The analysis is performed in different topologies of the network (BA, ER and GRG) and fixed coupling strengths (K). The findings indicate that for low to moderate coupling strengths ($K = 10, 100, 1000$), changes in network topology do not significantly affect SINDy reconstruction on different metrics. However, for a high coupling strength ($K = 10000$), substantial variations in the Frobenius adjacency metric are observed depending on the topology used. Additionally, reconstruction is feasible only with low average degrees for moderate coupling strengths, as higher average degrees lead to synchronization, making reconstruction challenging.

Furthermore, the effect of the number of input trajectories on reconstruction was explored using the BA topology with a fixed coupling strength of $K = 10$. The results reveal that at least 100 trajectories are needed to achieve a satisfactory reconstruction across the full range of average degrees. The quality of reconstruction improves with an increasing number of trajectories, especially at higher

average degrees, where synchronization complicates the process. This demonstrates the importance of having sufficient data points for accurate SINDy reconstruction, particularly when dealing with complex networks.

In the final section, we investigate the computational limits of the SINDy algorithm, specifically focusing on RAM usage and fitting time as the number of nodes in the network increases. To determine the optimal number of trajectories needed for accurate reconstruction, a method is used that involves comparing the Frobenius adjacency matrix reconstructed by SINDy with the original adjacency matrix. This comparison is facilitated by creating a confusion matrix that identifies discrepancies between the original and reconstructed matrices, guiding the adjustment of the number of input trajectories to minimize errors.

The results indicate that while all the sparse regression algorithms except `Lasso` exhibit similar RAM usage, the RAM consumption of the `SR3` algorithm scales with the fourth power of the number of nodes (N^4), as confirmed by a strong linear fit ($r^2 = 0.999$) and the corresponding residuals, considering $x = N^4$ as the independent variable. In terms of fitting time, `Lasso` outperforms other algorithms, while `STLSQ` performs the worst. The fitting time for `SR3` scales approximately with the sixth power of the number of nodes (N^6), although some nonlinearity is observed, particularly in the early stages.

To ensure the stability of these findings, the study tested the algorithm with different thresholds for the confusion matrix, confirming that variations in the threshold had minimal impact on both RAM usage and fitting time. Thus, the method is robust and reliable for assessing SINDy's computational performance across different network sizes and sparse regression algorithms.

Conclusions

In this thesis, I have examined the functionality of SINDy (Sparse Identification of Nonlinear Dynamics), established certain metrics to evaluate the quality of the network and dynamics reconstructed by SINDy, and explored its application to the Kuramoto model across various coupling values and average degrees, employing different sparse regression algorithms, time step intervals of the provided time series data, and different number of trajectories used.
Specifically, we observed that SINDy's ability to rebuild the network and dynamics is influenced by the synchronization speed between nodes as well as the amount of input data; more data yield better results.
Moreover, the outcomes resulting from changes in coupling (which represents the strength of interaction between oscillators) are dependent on the sparse regression algorithm utilized, but all the algorithms were unsuccessful in reconstructing the network and dynamics beyond a coupling threshold of approximately 1000. Analyzing the results of varying the average degree for different network topologies (GRG, BA, ER) revealed that the results for the three network topologies are consistent for fixed coupling values. Furthermore, by fixing both a network topology and a coupling value, we see that the results vary with the number of trajectories used; more trajectories improve the results.
Finally, I evaluated the computational performance of SINDy with respect to both RAM usage and fitting time for different algorithms. The findings indicate that most algorithms exhibit similar requirements for memory capacity (here, we measured this in terms of RAM usage), except Lasso, which behaves slightly distinctively.
In contrast, when examining the fitting time, the algorithms perform differently, Lasso being the most efficient. Analyzing the data from the SR3 algorithm shows that the RAM usage scales as N^4 and the fit time scales as N^6, where N represents the number of nodes in the network.

Returning to our initial research question—whether we can extract the laws of motion and the network structure of a dynamical system from time series data—we find that, in the case of the Kuramoto model with 32 nodes, SINDy successfully reconstructs the laws of motion across various network topologies, provided sufficient input data. This holds for a wide range of coupling values, $10^{-4} < K < 10^3$. However, inferring the network structure requires prior knowledge of the coupling strength used.

To further enhance this project and build upon the findings of this thesis, a promising area is the investigation of other complex dynamical systems beyond the Kuramoto model to evaluate the generalizability of SINDy. This could include systems with higher-order interactions or more intricate coupling functions, providing a broader validation of the methodology.

Bibliography

[1] Erdős, P., & Rényi, A. (1959). *On random graphs I.* Publicationes Mathematicae, 6, 290-297.

[2] Watts, D. J., & Strogatz, S. H. (1998). *Collective dynamics of 'small-world' networks.* Nature, 393(6684), 440-442. DOI: 10.1038/30918

[3] Barabási, A.-L., & Albert, R. (1999). *Emergence of scaling in random networks.* Science, 286(5439), 509-512. DOI: 10.1126/science.286.5439.509

[4] Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). *Stochastic blockmodels: First steps.* Social Networks, 5(2), 109-137. DOI: 10.1016/0378-8733(83)90021-7

[5] Penrose, M. D. (2003). *Random Geometric Graphs.* Oxford University Press.

[6] Kuramoto, Y. (1975). *Self-entrainment of a population of coupled non-linear oscillators.* In H. Araki (Ed.), International Symposium on Mathematical Problems in Theoretical Physics (pp. 420-422). Springer.

[7] Arenas, A., Diaz-Guilera, A., Kurths, J., Moreno, Y., & Zhou, C. (2008). *Synchronization in Complex Networks.* Cambridge University Press.

[8] Brunton, Steven L., 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 (2016): 3932-3937.

[9] Brian M. de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Nathan Kutz, and Steven L. Brunton., (2020). *PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data. Journal of Open Source Software*, 5(49), 2104, https://doi.org/10.21105/joss.02104

[10] Kaptanoglu et al., (2022). *PySINDy: A comprehensive Python package for robust sparse system identification.* Journal of Open Source Software, 7(69), 3994, https://doi.org/10.21105/joss.03994

[11] Peng Zheng and Travis Askham and Steven L. Brunton and J. Nathan Kutz and Aleksandr Y. Aravkin (2018) *A Unified Framework for Sparse Relaxed Regularized Regression: SR3*, arXiv, 1807.05411

[12] Kathleen Champion and Peng Zheng and Aleksandr Y. Aravkin and Steven L. Brunton and J. Nathan Kutz *A unified sparse optimization framework to learn parsimonious physics-informed models from data*, arXiv, 1906.10612

[13] Boninsegna, Lorenzo, Feliks Nüske, and Cecilia Clementi. *Sparse learning of stochastic dynamical equations.*, The Journal of chemical physics 148.24 (2018): 241723.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). *Scikit-learn: Machine learning in Python.*, The Journal of machine Learning research, 12, 2825-2830.

[15] Mangan, N. M., Kutz, J. N., Brunton, S. L., & Proctor, J. L. (2016) *Inferring biological networks by sparse identification of nonlinear dynamics.*, IEEE Transactions on Molecular, Biological and Multi-Scale Communications, 2(1), 52-63.

[16] Moritz Hoffmann, Christoph Fröhner, Frank Noé. *Reactive SINDy: Discovering governing reactions from concentration data.* J. Chem. Phys. 14 January 2019; 150 (2): 025101. https://doi.org/10.1063/1.5066099

[17] Kaiser, E., Kutz, J., & Brunton, S. (2018). *Sparse identification of nonlinear dynamics for model predictive control in the low-data limit.* Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474(2219), 20180335.