UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Spectral identification of networks

Using cross-validation to improve the Dynamic Mode Decomposition algorithm

Dissertation presented by
Andine HAVELANGE

for obtaining the Master's degree in
Mathematical Engineering

Supervisors
Julien HENDRICKX, Alexandre MAUROY

Reader
Michel GEVERS

Academic year 2016-2017

Abstract

Spectral network identification allows to infer global topological properties of a network system from measurements of its dynamics at a few nodes. The Dynamic Mode Decomposition algorithm is used in this framework to estimate the dynamics spectrum, or Koopman eigenvalues, from which those topological properties can be deduced. However, it is not always effective and accurate and, most importantly, not optimized to capture the Koopman eigenvalues. We propose a method that aims at enhancing the Dynamic Mode Decomposition algorithm performance with respect to the estimation of the dynamics spectrum, based on the cross-validation technique. We show that our method does not enable any improvement in case of linear local dynamics but allows a slight enhancement in case of nonlinear local dynamics. We finally expose a modified version of our method which leads to significantly better results but only for small networks.

Acknowledgement

I sincerely thank my supervisors Julien Hendrickx and Alexandre Mauroy for their support, help and guidance throughout this work. I could not have achieved this without their numerous comments and suggestions, as well as the interesting discussions we have had all along this year. I also wish to thank my parents and all of those who supported and encouraged me throughout my years of study and through the process of researching and writing this thesis.

Contents

Introduction

Networks such as the neuronal system, the electricity power grid, social networks and transportation ones, to only cite a few, are part of our everyday life. They differ widely in a number of aspects such as their structure, composition and purpose. Yet, they all share one common characteristic in that they form complex networks of dynamical agents interacting with each other. These can be modeled using system networks, defined as a graph of which each node is a dynamical system, interacting with the other systems through the graph edges. Those networks of systems occur in numerous engineering and scientific fields and have received an increasing interest over the past 15 years. However, we often possess a limited knowledge about them in real-world applications. Their exact structure might be unknown, simply because the network is too large, or the dynamics describing the agents interaction might not be available.

Several frameworks have been developed to overcome this issue, allowing to derive information about the network system from available knowledge about it. One of those methods is the so called *spectral network identification* which enables to infer global information about the network topology, such as its mean degree, from measurements of the dynamics of a few of its nodes. More precisely, global information about the network structure is derived from its spectral properties, linked with the spectrum of the Laplacian matrix of the network. Those properties are related to the eigenvalues of the network collective dynamics, obtained by means of the Dynamic Mode Decomposition algorithm from measurements of the dynamics. Spectral network identification is very promising as it allows to infer global information about the structure of very large networks by measuring a few nodes, possibly only one. But it is also a very recent method and, as such, it is still being developed and improved. This work has been performed in this perspective and aims more specifically at enhancing the numerical algorithms involved in this method.

In particular, the Dynamic Mode Decomposition algorithm that is used to obtain the spectrum of the network dynamics is also recent, not always fully reliable and has not been optimized with respect to the estimation of the eigenvalues of the dynamics. The purpose of this work is to improve the approximation of the dynamics spectrum by enhancing the performance of the Dynamic Mode Decomposition algorithm. This would allow to obtain more accurate information about the topology of the network, which is the final goal of spectral network identification. The method we developed to achieve this goal is based on cross-validation, a technique borrowed from the machine learning framework.

This work is organized as follows. The first chapter is dedicated to a brief introduction to network systems and a thorough overview of spectral network identification. The Dynamic Mode Decomposition algorithm and how it is used in the spectral identification framework are also presented. The end of the first chapter exposes the aim of this work in more details. In the second chapter, we focus on the method we developed to enhance the Dynamic Mode Decomposition

algorithm performance regarding the estimation of the dynamics spectral properties, based on the cross-validation technique. We first briefly introduce supervised learning, which is the field of machine learning in which cross-validation has been developed. We then expose the cross-validation method and draw a parallel between supervised learning and the dynamics spectrum estimation, to finally arrive at the detailed presentation of our method. In the third and last chapter, we aim at validating our method and examine its performance in different settings before finally drawing conclusions and further prospects.

Chapter 1

Spectral network identification

1.1 Network systems

A loose definition
A network system can be loosely defined as a system of systems, or as a complex network of interacting dynamical units. A dynamical system is attached to each of the nodes of the network and two systems are linked by an edge if one of them (or both) is influencing the other one.

Real-world illustrations
Network can be used to model processes such as biological and social contagions [1] for instance. In the first case, nodes correspond to individuals, each of them being in an epidemic state, i.e. susceptible, infected or recovered. Two of them are linked by an edge if they are in contact and can thus potentially infect each other. The epidemic state of each "vertex" is dynamically evolving over time and is influenced by the states of other vertices in the network. Social contagion can be modeled in the same way but in this case, information, ideas or rumors are spread amongst people, rather than a disease and edges represent the social influence between people. The dynamic state of a vertex reflects whether its associated social actor has heard about the rumor or not, or whether he or she agrees with an idea or not. A third illustration is the neuronal system [2], where neurons are the nodes and edges the synapses connecting them and where the dynamics of a neuron can be described by its membrane voltage. Such a model could be used for instance to study the propagation of neural signals through the brain. A last example is the electrical power grid [3], of which the generators, transformers and substations are attached to vertices while edges represent high-voltage transmission lines between them.

A wide range of applications
As illustrated by the above-mentioned examples, network systems occur in divers engineering and scientific fields such as communication networks, sensor networks, water irrigation networks, mutli-robot networks, camera networks, transportation networks, social networks, and chemical and biological networks [4]. Their applications vary from environmental monitoring, ocean sampling and marine energy systems through search and rescue missions, high-stress development in disaster recovery and health monitoring to science imaging, the smart grid and cybersecurity. The modeling and analysis of such networks of dynamical systems is thus playing an increasingly important role in the scientific and engineering domains. It is precisely this variety that makes it difficult to define more accurately network systems as this definition should be at the same time comprehensive enough to capture their diversity and simple enough to express their main characteristics. In the definition given just above, the dynamical units attached to nodes could

be humans, biological agents or even engineered ones and the type of interactions defining the collective dynamics depends on the application that is dealt with.

1.2 Network system identification

Incomplete knowledge about the network
Networks of dynamical systems have been extensively studied over the last 15 years or so but, till recently, the network was assumed known or to be designed [5]. Only by looking at the example of the human neural system, it is evident that this hypothesis is not valid in many applications. Mapping the neuronal system in our brain is indeed a challenge that has still not been met. Network systems encountered in real-world are frequently not perfectly known. We may have to face several unknowns, such as the number of vertices or edges, the link structure or the dynamics of the units attached to the nodes.

1.2.1 Two main frameworks in inferring information about network systems

Inferring the dynamics from the topology
One of the challenges related to this issue is the identification of the dynamics, when the topology of the network, i.e. its number of nodes, edges and link structure, is known. The underlying question being what kind of collective dynamics arises from a given network structure. For example, in [6–8], the dynamic network is defined as an interconnection of transfer functions where the interconnecting signals are the nodes and the proper transfer functions the edges. The topology of the network is known and the goal is to identify one or several transfer functions, on the basis of data measurements.

Inferring the topology from the dynamics
In many applications, it is however desirable to address the inverse question, i.e. how to identify the topology of the network from its dynamics. This is of interest in various fields such as [9] biology (gene and protein regulatory networks and metabolic and chemical reactions), neuroimaging (capturing the brain configuration) and engineering (electric power grids and wireless sensor networks). It is indeed often possible to access and measure the dynamics of individual units whereas the topology of the network is unknown. This happens when considering very large real-world system networks, such as the neural one, for instance. The problem of reconstructing the network structure from data capturing its collective dynamics has thus received more and more interest over the past years.
Several techniques have been proposed to address this challenge (for more information about these, see [10–20]). Every method assumes different levels of pre-knowlege about the system network and strive to identify different information about it (see [9]). One could be interested in whether the network is directed or undirected, in whether an edge between two specific nodes exists or not, in the strength of the interactions between units, in the type of links and so on.

1.2.2 Complete topology reconstruction versus inferring global information

Complete topology reconstruction
One common thread of the existing network identification methods is that their goal is to identify the exact topology of the network. This means they aim at inferring the exact number of nodes and edges along with the exact structure of it, i.e. which pairs of nodes are connected by an edge and the weights of these edges. As so, they can hardly be used to study real networks. It can indeed be shown that fully reconstructing the network requires to measure all of its

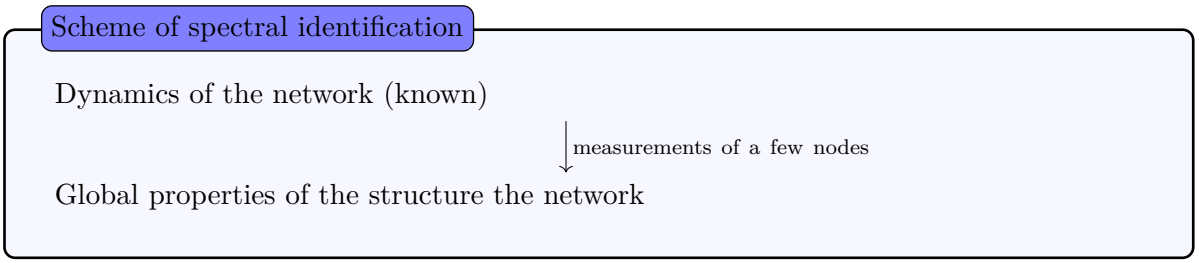
nodes and this is in most cases impossible for large networks. Remember that the number of neurons in the human brain is estimated to be between 86 and 100 billions, which makes it out of reach to measure them all at the same time. Additionally, it might be that not all the nodes are measurable or that only the averaged dynamics of a subset of nodes is available. Another drawback of the above-mentioned methods is that most of those are invasive, i.e. they require to modify the dynamics structure of the network, or must be used "online", that is they necessitate to dynamically interact with the network while proceeding.

Identification of global structure properties
These limitations motivate the network identification framework developed by Julien Hendrickx and Alexandre Mauroy in [2] and [21]. They note that reconstructing the exact connection structure of the network is not only often impossible but also unnecessary. When considering large networks, knowing whether two specific nodes are linked to each other by an edge or not is of relatively minor interest. Being able to somehow infer information about the global structure of the network is, on the contrary, more relevant. The authors therefore focus on identifying global properties of the network structure (also called spectral properties) based on few measurements of its dynamics, a framework that they named *spectral network identification*. The reason for choice of the term *spectral* will be made clear thereafter. The idea of approximating the spectral properties of a network has already been studied in the literature but in the case of specific consensus imposed at each vertex (see [22–24]). The approach of Julien Hendrickx and Alexandre Mauroy is innovative in that only a few measurements of the network dynamics are required and that it can be applied to a broader class of networks.

Main features of spectral identification
The main drawback of spectral identification it only allows to infer the spectral properties of the network, not its exact topology. But spectral properties provide information capturing the global structure of the network, such as its mean, minimum and maximum degree, and connectivity. Other advantages of spectral identification is that it can be used offline and is not invasive. And maybe its greatest strength compared to the above-mentioned techniques is that it requires to measure only a small number of nodes. Inferring global properties of the structure of large networks by only measuring a few nodes may be surprising at first sight. This is intuitively explained by the fact that each node of a connected network "feels" the influence of all other nodes, through the edges representing the interaction between the vertices. Only a few nodes (possibly one in some cases) need thus to be measured and they do not have to be representative of the vertices set. Also, measurements can be defined by a function (possibly non-linear) depending on the states of several nodes, such as the average dynamics of a group of units.

1.2.3 Summary

The scheme of spectral network identification can, for now, be roughly summarized as



Throughout the rest of this work, we will consider spectral network identification as it is defined in [2] and we will formally define its framework in the remaining part of this chapter. We will first express the problem of spectral identification more accurately.

1.3 Problem statement of spectral network identification

1.3.1 Classical versus spectral network identification

Formal description of the network system
A network system is composed of n interconnected dynamical systems, attached to the n nodes of a graph $\mathcal{G} = (V, E, w)$, interacting through the (possibly weighted) edges of $\mathcal{G}$. The dynamical systems are deterministic since no stochastic perturbation is considered. Each system (or unit) is attached to a node $k \in V$ and is influenced by another unit i if there exists an edge connecting both nodes, i.e. if $(i, k) \in E$. The strength of the interactions between units is defined by the function $w : E \to \mathbb{R}^+$ which assigns a weight to each edge of $\mathcal{G}$. The graph $\mathcal{G}$ is static, i.e. its number of nodes and its edges structure do not change over time.

Put briefly, the networked dynamical system is fully defined by

(1) the graph $\mathcal{G}$

(2) the local dynamics of the states of the units attached to the nodes

(3) the type of coupling between the pairs of interacting units

Matrix representations of the graph $\mathcal{G}$
The (weighted) degree of a vertex i is given by

$$d_i = \sum_{k=1}^{n} w(i, k)$$

where

$$w(i,k) = \begin{cases} w((i,k)) & \text{if } (i,k) \in V \\ 0 & \text{if } (i,k) \notin V \end{cases}$$

and it is assumed that there is no self-loop in $\mathcal{G}$, i.e. $(i, i) \notin E \ \forall i \in V$. The degree matrix is defined as

$$D = \text{diag}(d_1, d_2, ..., d_n)$$

Two important other matrices can be used to describe the graph. The first one is its (weighted) adjacency matrix, W, defined by

$$W_{ik} = w(i, k)$$

The second matrix is the (weighted) Laplacian matrix of $\mathcal{G}$

$$L = D - W \tag{1.1}$$

In the following, we consider graphs that can be weighted and directed, that is $W \neq W^T$ and $L \neq L^T$ given that $(i, k) \in V$ does not imply $(k, i) \in V$.

Problem statement
Network identification focuses on the following problem: under the hypothesis that the local dynamics of the units (2) and the type of coupling between pairs of units (3) are known, infer the topology of the graph (1) from measurements of the local dynamics of the units. Two sub-problems arise from this one, referred to as classical and spectral network identification, and are defined as:

Classical identification	assume that (2) and (3) are known. From measurements of the dynamics of all units of the network, infer the set of its edges E and the weight function w, i.e. the exact topology of $\mathcal{G}$
Spectral identification	assume that (2) and (3) are known. From $p \ll n$ measurements of the dynamics of the states of a small subset $\bar{V} \subset V$ of units, estimate the spectrum of the Laplacian matrix, $\sigma(L)$, or its first spectral moments (which will be defined just below) to then infer information about the global structure of $\mathcal{G}$

Assumptions
In this work, we consider network systems such that the dynamical units interact through a diffusive coupling, i.e. each node influences and/or is influenced only by its neighbors, that is the nodes it is connected to by an edge. In other words, a unit i interacts with unit j if and only if $(i, j) \in V$ or $(j, i) \in V$. This kind of interaction is in opposition to global or "all-to-all" coupling, where each node interacts with equal strength with all other nodes in the network.
We also assume that studied network systems converge to a stable equilibrium corresponding to the synchronization of all the units, i.e all units reach the same value at equilibrium. This behavior can be observed with excitable neurons, cardiac cells, opinion dynamics and epidemics.

1.3.2 Relation between the spectral properties of L and the topology of $\mathcal{G}$

As exposed above, even if spectral network identification does not allow to recover the exact topology of the graph, it only requires a few measurements to retrieve spectral information about the Laplacian matrix, from which properties about the global structure of $\mathcal{G}$ can be inferred. But what spectral properties exactly are we talking about? And how are they related to the topology of the network?

Spectrum of L
The spectrum (eigenvalues) of L, denoted by $\sigma(L)$, allows to deduce various topological properties of the graph. For instance, in the case of a connected graph, the second smallest eigenvalue λ_2 (also called the algebraic connectivity) reveals information about the connectivity of the graph and provides a bound on its diameter, i.e. the length of the longest path between any pair of nodes.
Additionally, the algebraic connectivity λ_2 and the spectral radius λ_n (the largest eigenvalue) of the Laplacian can be used to derive bounds on the minimum and maximum degree of the graph. In the case of undirected networks,

$$d_{\min} \geq \frac{n-1}{n}\lambda_2 \qquad\qquad d_{\max} \leq \frac{n-1}{n}\lambda_n$$

Spectral moments of L

Another spectral property of L we are interested in is its spectral moments, the k^{th} one being defined as

$$\mathcal{M}^{(k)}(L) = \frac{1}{n}\sum_{\lambda\in\sigma(L)}\lambda^k = \frac{1}{n}tr(L^k) \quad k \in \mathbb{N}$$

where $tr(L)$ denotes the trace of L. The spectral moments of the Laplacian are related to the moments of the degree distribution as follows. The first one is equal to the mean degree of the graph, denoted by $\mathcal{D}^{(1)}(\mathcal{G})$ i.e.

$$\mathcal{M}^{(1)}(L) = \mathcal{D}^{(1)}(\mathcal{G}) \triangleq \frac{1}{n}\sum_{i=1}^{n} d_i \tag{1.2}$$

and the first and the second spectral moments enable to derive bounds on the quadratic mean degree distribution, denoted by $\mathcal{D}^{(2)}(\mathcal{G}) \triangleq \frac{1}{n}\sum_{i=1}^{n} d_i^2$

$$\max\left((\mathcal{M}^{(2)}(L))^2, \frac{\mathcal{M}^{(2)}(L)}{2}\right) \leq \mathcal{D}^{(2)}(\mathcal{G}) \leq \mathcal{M}^{(2)}(L) \tag{1.3}$$

The proof of (1.3) is given in [2].

1.3.3 Summary

The underlying problem of spectral network identification can be summarized as follows

Problem statement of spectral identification

Measurements of the local dynamics of a few nodes

$\downarrow$?

Spectral properties of the Laplacian ($\sigma(L)$ and $\mathcal{M}^{(k)}(L)$)

$\downarrow$ spectral graph theory

Topological properties of the network ($d_{\max}, d_{\min}, ...$)

So far, we have exposed how information about the graph structure can be derived from spectral properties of its Laplacian matrix. However, when considering real-world, large networks, their Laplacian matrix is unknown as it depends on the adjacency and degree matrices of the network (see (1.1)), which can not be computed since the topology of the network is unknown... But as said previously, the spectrum and spectral moments of L can be inferred from measurements of the dynamics of the states of a few nodes of the network. The methods developed for that purpose in [2] are exposed in details in the following sections.

1.4 Exact spectral identification

1.4.1 Linear systems with identical units

In this section, we present the spectral network identification framework that has been developed for identical units. Linear dynamics is first considered and results are then extended to nonlinear dynamics.

Collective dynamics of the networks
We consider a network of n interconnected, identical units that are each described by m states evolving according to the linear dynamics

$$\begin{aligned} \dot{x}_i &= Ax_i + Bu_i \\ y_i &= C^T x_i \end{aligned} \qquad i = 1, .., n \tag{1.4}$$

with $x_i \in \mathbb{R}^m$ the state vector of the dynamical unit attached to node i and $y_i \in \mathbb{R}$. $A \in \mathbb{R}^{m \times m}$, $B \in \mathbb{R}^{m \times 1}$ and $C \in \mathbb{R}^{m \times 1}$ are the matrices describing the local dynamics of the units and are assumed to be known.

The units interact though a diffusive coupling given by

$$u_i = \sum_{j=1}^{n} W_{ij}(y_i - y_j) \tag{1.5}$$

Defining the state vector of the entire network as $X = [x_1 \ x_2 \ ... \ x_n]^T \in \mathbb{R}^{nm}$, the collective dynamics of the network is described by

$$\dot{X} = (I_n \otimes A - L \otimes BC^T)X$$

where I_n is the $n \times n$ identity matrix and $\otimes$ denotes the Kronecker product.
For the sake of simplicity, we denote

$$K \triangleq I_n \otimes A - L \otimes BC^T \in \mathbb{R}^{nm \times nm} \tag{1.6}$$

The analytical solution to $\dot{X} = KX$ is given by

$$X(t) = \sum_{j=1}^{mn} V_j e^{\mu_j t} \tag{1.7}$$

with V_j the eigenvectors of matrix K and μ_j their corresponding eigenvalue. Note that V_j depend on the initial condition $X(0)$. It is assumed that the matrices A, B and C are such that the units synchronize, that is

$$\lim_{t \to \infty} X(t) = \mathbf{0} \quad \text{or equivalently} \quad \Re\{\mu_j\} < 0 \ \forall j$$

Measurements
Even though the dynamics of the local units, i.e. the matrices A, B and C, is assumed to be known, the matrix K describing the collective dynamics of the network cannot be computed as it also depends on the Laplacian matrix L (see (1.6)), which is unknown. Yet, we can to a certain extent obtain information about K by means of measurements of the dynamics.

In case of spectral network identification, measurements are performed through an observation function, the value of which is called an observable. An observable is "an experimentally accessible quantity" [25] or, in other words, a dynamic variable that can be measured. In dynamical systems governed by classical mechanics, if the state of the system is defined as the position, possible observables are the velocity, the energy or the position itself. When studying fluid flows, observables can be the velocity, pressure or vorticity fields. More generally, an observable is a function of the state of the dynamical system that can somehow be measured. It will be

mathematically defined in subsection 1.4.3 (The Koopman operator).

We will first consider a linear observation function

$$f(X(t)) = Q^T X(t) \quad \in \mathbb{R}^p \tag{1.8}$$

where $X(t)$ is a trajectory of (1.4)-(1.5) associated with the initial condition $X(0) = X_0$, $Q \in \mathbb{R}^{nm \times p}$ is a sparse matrix and $p \ll n$ is the number of measurements. The case of a nonlinear observation function will be presented in subsection 1.4.2 (Nonlinear systems with identical units), together with nonlinear dynamics.

Estimating $\sigma(K)$ from measurements
It is worth noting that, substituting (1.7) in (1.8), we obtain

$$f(X(t)) = Q^T(\sum_{j=1}^{mn} V_j e^{\mu_j t}) = \sum_{j=1}^{mn} Q^T V_j e^{\mu_j t} \tag{1.9}$$

provided that $f(V_j) = Q^T V_j \neq 0 \; \forall j$. Note that the eigenvalues μ_j of the matrix of the collective dynamics of the network K appear in this expression of the observable. Approximations $\tilde{\mu}_j$ of the exact eigenvalues μ_j of K can therefore be computed from measurements, i.e. from snapshots of (1.9), even if only one state of a single vertex is measured, which amounts to choose

$$Q = e^l \quad \Rightarrow \quad f(X(t)) = (e^l)^T X(t) = X_l \quad \in \mathbb{R} \tag{1.10}$$

where e^l is the l^{th} unit vector, i.e. the vector whose components are all zero except the l^{th} one which is equal to 1. Estimates $\hat{\mu_j}$ of exact eigenvalues μ_j of K are obtained using a method called the Dynamic Mode Decomposition (DMD) algorithm.

The DMD algorithm will be presented in details in section 1.6 (Dynamic Mode Decomposition algorithm). For now, we assume that the spectrum of the matrix K, that is the spectral properties of the dynamics, can be estimated from measurements of the dynamics, i.e. the values of $f(X(t))$.

Inferring $\sigma(L)$ from $\sigma(K)$
What remains to expose is how the spectrum of the Laplacian matrix of the graph, L, can be inferred from the (estimated) spectrum of the dynamics matrix, K, assuming that the local dynamics, i.e. A, B and C, is known. The relationship between $\sigma(K)$ and $\sigma(L)$ is summarized in the following proposition.

Proposition 1. *Assume that the local dynamics* (1.4) *is controllable and observable (i.e. rank*$[B \; AB \; ... \; A^{m-1}B] = m$ *and rank*$[C \; A^TC \; ... \; (A^T)^{m-1}C] = m$ *respectively). Then,*

$$\sigma(L) = \{g(\mu) | \mu \in \sigma(K)\} \tag{1.11}$$

with

$$g(\mu) = \begin{cases} 1/(C^T(A - \mu I_m)^{-1}B) & \text{if } \mu \notin \sigma(A) \\ 0 & \text{if } \mu \in \sigma(A) \end{cases}$$

Moreover,

$$\sigma(K_1) = \sigma(K_2) \Leftrightarrow \sigma(L_1) = \sigma(L_2) \tag{1.12}$$

with $K_1 = I_n \otimes A - L_1 \otimes BC^T$ *and* $K_2 = I_n \otimes A - L_2 \otimes BC^T$.

We are mainly interested in the results presented in proposition 1 rather than in its proof and how it has been derived but these are exposed in [2].

1.4.2 Nonlinear systems with identical units

We now examine the case of networks of which the local dynamics of the units and the observation function are nonlinear. The spectral identification framework also applies to a nonlinear dynamics associated with a linear observation function and conversely, of course.

Collective dynamics of the network
The nonlinear dynamics of the (identical) units is given by

$$\begin{aligned} \dot{x}_i &= F(x_i) + G(x_i)u_i \\ y_i &= H(x_i) \end{aligned} \qquad i = 1, .., n \tag{1.13}$$

with the analytic functions $F : \mathbb{R}^m \to \mathbb{R}^m$, $G : \mathbb{R}^m \to \mathbb{R}^m$ and $H : \mathbb{R}^m \to \mathbb{R}$ that are assumed to be known. The units interact through the same diffusive coupling as in the case of linear local dynamics

$$u_i = \sum_{j=1}^{n} W_{ij}(y_i - y_j) \tag{1.14}$$

It is assumed that (1.13) admits a stable fixed point x^* and that the units synchronize, so that the trajectory $X(t)$ converges to the stable fixed point $X^* = [x^* \ ... \ x^*]^T$.

The Jacobian matrix associated with (1.13)-(1.14) linearized at X^* is given by

$$K \triangleq I_n \otimes \underbrace{\frac{\partial F}{\partial x}(x^*)}_{A} - L \otimes \underbrace{G(x^*)}_{B}(\underbrace{\nabla H(x^*)}_{C})^T \in \mathbb{R}^{nm \times nm} \tag{1.15}$$

(where ∇ denotes the gradient).

Inferring $\sigma(L)$ from spectral properties of the dynamics
(1.15) being very similar to (1.6), the relationship between the spectrum of the Laplacian matrix L and the eigenvalues of the dynamics matrix K can be obtained from proposition 1, as stated in the following proposition

Proposition 2. *Consider the local dynamics* (1.13) *and assume that*

$$(A, B) = \left(\frac{\partial F}{\partial x}(x^*), G(x^*)\right) \quad \textit{and} \quad (A, C) = \left(\frac{\partial F}{\partial x}(x^*), \nabla H(x^*)\right)$$

are controllable and observable pairs, respectively.
Then the relationship between the spectrum of the Laplacian matrix L and the spectrum of the Jacobian matrix K (1.15) *is a bijection, i.e.* (1.12) *holds. Moreover, the spectrum of L is given by* (1.11).

Proposition 2 implies that $\sigma(L)$ can be retrieved from the estimated eigenvalues $\tilde{\mu}_j$ of the Jacobian matrix K. Moreover, $\sigma(K)$ can be obtained from sparse measurements of the network dynamics, again using the Dynamic Mode Decomposition algorithm. But these measurements need then to be realized close enough to the equilibrium X^* for the DMD algorithm to be able to capture the eigenvalues of the Jacobian matrix of the dynamics linearized at X^*. However, this might be complicated in real applications. That is why we resort to the Koopman operator framework, into which we will give an insight in what follows, based on [26] and [27]. How to estimate $\sigma(K)$ from measurements in the case of nonlinear dynamics and nonlinear observation function will be exposed after introducing this new concept.

1.4.3 The Koopman operator

Various methods to deal with nonlinear dynamics
A commonly used technique to deal with nonlinear systems is to linearize their dynamics at a stable equilibrium. However, the so obtained dynamics approximation is valid only in a certain region around this equilibrium point. Another method often applied to handle complicated functions (such as nonlinear ones) is to expand them into an infinite weighted sum of simpler components. Examples of such approaches are the Taylor and Fourier expansions. For these decompositions, a predetermined set of functions (corresponding to the simpler components) is chosen and the complicated function is projected onto those to obtain the coefficients (weights) of the sum. An alternative scheme has recently emerged and tackles the problem of nonlinear dynamics from the perspective of operator theory, using the Koopman operator.

Main features of the Koopman operator
In the Koopman operator framework, the dynamics is studied by looking at the evolution of functions of the state space, rather than directly looking at state space trajectories as it is usually done. Those functions of the state space are called observables. The evolution of the dynamics is analyzed by expanding the observable into the basis of eigenfunctions of the Koopman operator. The weights of the decomposition are obtained by projecting the observable onto the corresponding eigenspaces of the Koopman operator. They are called the Koopman modes.

The Koopman operator is a linear operator allowing to capture the complete nonlinear dynamics within a linear setting. This is probably its most significant advantage as it enables to apply the spectral analysis to nonlinear systems without losing any information, as required by other linearization methods. Moreover, it is possible to construct, approximate or analyze the Koopman operator using only simulations or experimental measurements data.

But there is also a price to pay for being able to capture the full nonlinear dynamics in a linear setting. It is indeed possible due to the fact that the Koopman operator is infinite-dimensional even when the state space is finite-dimensional. Because of this, the implementation of any approximation is more challenging. And the associated numerical methods are currently underdeveloped.

Formal definition of the Koopman operator
The definition we will expose here is based on [26]. We consider a dynamical system whose dynamics is described by

$$\dot{X} = G(X) \tag{1.16}$$

defined on a state space M (i.e. $X \in M$), where X is a vector and G is a (possibly nonlinear) vector-valued function of the same dimension as X. $X(t, X_0)$ denotes the position at time t of the trajectory of (1.16) that starts at point X_0 at time zero.
We define an observable to be a function $f : M \to \mathbb{R}^p$, where f is an element of a function space which is a vector space (this assumption needs to be verified for the Koopman operator to be linear). If the system starts at X_0 in time zero, the value of f at time t is given by

$$f(t, X_0) = f(X(t, X_0))$$

A concrete interpretation of an observable is that of a sensor for the dynamical system: information about the state of the system is accessible via the evolution of the observable value. So, instead of following the trajectory $\{X_0, G(X_0), G^2(X_0), ..., \}$, the trace $\{f(X_0), f(G(X_0)), f(G^2(X_0)), ..., \}$

is tracked.

The Koopman operator of the system (1.16) is the family of operators U^t acting on the space of observables parametrized by time t, such that

$$(U^t f)(X_0) = f(X(t, X_0))$$

For a fixed value of time τ, U^τ maps the vector-valued observable $f(X_0)$ to its value at that time, $f(\tau, X_0)$. Put otherwise, the value $(U^\tau f)(X_0)$ is the value that will be observed at time τ via f for a trajectory of the system that starts at X_0 at time zero. When the state space M is a finite set, U^t is finite-dimensional and it can be represented by a matrix. But when M is finite- or infinite-dimensional, U^t is generally infinite-dimensional.

Koopman mode analysis
As the Koopman operator is linear, it is possible to consider its spectral properties to analyze the dynamics of the system (1.16), similarly to the case of linear finite-dimensional systems. An eigenfunction $\gamma(X)$ of the Koopman operator is a special observable such that

$$(U^t\gamma)(X_0) = \gamma(X(t, X_0)) = e^{\psi t}\gamma(X_0)$$

where ψ is the eigenvalue of the Koopman operator associated to the eigenfunction $\gamma(X)$ (in the discrete case, $(U^t\gamma)(X_0) = \gamma(X(t, X_0)) = \psi\gamma(X_0)$).

Let γ_j be an eigenfunction for the Koopman operator corresponding to the eigenvalue ψ_j. Given a vector-valued observable f, the Koopman mode $k_j(f)$ corresponding to γ_j is the vector of the coefficients of the projection of f onto span$\{\gamma_j\}$.
Koopmand modes are of interest because they allow for an decomposition akin to the eigenvectors expansion used in linear dynamics (see (1.7) and (1.9)). Indeed, an observable f can be expanded as

$$f(X_0) = \sum_{j=1}^{N} k_j(f)\gamma_j(X_0)$$

where N might be finite or infinite, depending on the dimension of the Koopman operator. Given that it is generally infinite, so will be N in most cases. The dynamics of f can then be described very easily using the Koopman operator setting, as

$$\begin{aligned}(U^t f)(X_0) &= f(X(t, X_0)) \\ &= \sum_{j=1}^{N} k_j(f)\gamma_j(X(t, X_0)) \\ &= \sum_{j=1}^{N} k_j(f)(U^t\gamma_j)(X_0) \\ &= \sum_{j=1}^{N} k_j(f)e^{\psi_j t}\gamma_j(X_0) \end{aligned} \tag{1.17}$$

Koopman modes, eigenfunctions and eigenvalues can be computed analytically but it is far from being easy in practice, especially because an explicit representation of the Koopman operator is generally not available. Its behavior can only be analyzed through its action on an observable and at a finite number of initial conditions. This leads us thus to consider data-driven algorithms to estimate the spectral properties of U. These data are a sequence of observations of an

observable along a trajectory of the system, $X(t, X_0)$. This sequence of observation is then used by these algorithms to estimate the eigenvalues, the modes and the eigenfunctions of the Koopman operator. One of these algorithms is precisely the Dynamic Mode Decomposition algorithm.

For a detailed review of the definition, computation, estimation and applications of the Koopman operator and Koopman mode analysis, see [27].

Inferring $\sigma(K)$ from measurements
Let us now revert to the measurements of the dynamics in the case of nonlinear systems and to the estimation of the spectral properties of the Jacobian matrix of the dynamics, K, from those.

Let $X(t, X_0)$ be a trajectory of (1.13)-(1.14) that starts at X_0 in time zero. $p \ll n$ measurements of $X(t, X_0)$ have been obtained through a possibly nonlinear observation function $f : \mathbb{R}^{nm} \to \mathbb{R}^p$, which depends on a few local states in the spectral identification framework. Applying the Koopman operator on f gives

$$(U^t f)(X_0) = f(X(t, X_0))$$

i.e. the value that will be observed at time t via the observation function f for a trajectory of which the initial condition is X_0, that is the measurements. As explained above, the Koopman operator is linear so it can be characterized by spectral properties and, provided that f is analytic, the spectral expansion of the operator leads to

$$f(X(t, X_0)) = X^* + \sum_{(j_1,...,j_{nm}) \in \mathbb{N}^{nm}} V_{(j_1,...,j_{nm})} e^{(j_1\mu_1 + ... + j_{nm}\mu_{nm})t} \tag{1.18}$$

where

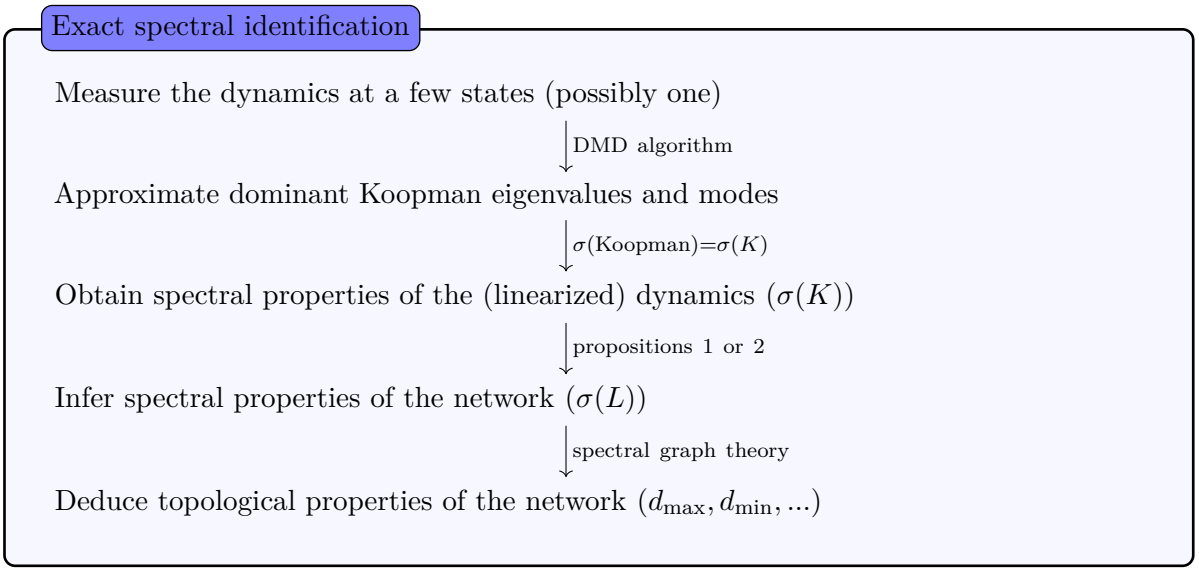
- X^* is the stable fixed point towards which all the units converge
- $j_1\mu_1 + ... + j_{nm}\mu_{nm}$ are the eigenvalues of the Koopman operator, i.e. ψ_j in (1.17)
- $V_{(j_1,...,j_{nm})} \triangleq k_j(f)\gamma_j(X_0) \in \mathbb{R}^p$ in(1.17), i.e. the product of the modes and associated eigenfunctions of the Koopman operator. In the rest of this document, we will refer to them as the modes of the Koopman operator. This is a slight abuse of notation given that $\gamma_j(X_0)$ is a constant when the initial condition is known.

As exposed previously, the Koopman modes and eigenvalues can be estimated from measurements of the dynamics, i.e. from snapshots of (1.18) ,by means of the DMD algorithm (which will be presented in details in section 1.6 (Dynamic Mode Decomposition algorithm)). It can be shown that the spectral properties of the dynamics can be inferred from the spectral properties of the Koopman operator. Indeed, provided that the dominant Koopman modes (the ones associated with the dominant Koopman eigenvalues) are nonzero, the DMD algorithm yields the dominant Koopman eigenvalues (denoted, with a slight abuse of notation, by σ(Koopman)), which are the ones of the matrix K. That means that the eigenvalues of the linearized dynamics, i.e. of the Jacobian matrix K, can be captured by the measurements whether these are close to the linearization point or not. This is obviously also valid in the case of linear dynamics as the Jacobian matrix of the dynamics (1.15) is simply equal to (1.6).

Once the spectrum of K has been estimated, proposition 2 is used to infer the spectral properties of the Laplacian matrix L and, from these, information about the topology of the network.

1.4.4 Summary

The scheme defined by exact spectral network identification can be summarized as follows



This framework is termed as *exact* spectral network identification because it allows to infer the exact connection between the spectrum of the dynamics, i.e. of matrix K, and the spectrum of the Laplacian matrix, i.e. of matrix L, which is related to the topology of the network. In other words, it allows to recover each individual eigenvalue of L.

In what follows, we will consider networks such that the assumptions of propositions 1 and 2 are verified, i.e. such that the (linearized) local dynamics of the units is controllable and observable. However, the spectrum of L can also be inferred from the spectrum of K even if this condition is not fulfilled. How it is achieved can be found in [2].

Non-identical units
It has been shown in [2] that, except for some very restrictive cases, the proble of spectral network identification cannot be solved when the units are non-identical, i.e. when their local dynamics differ. Note also that the linearized dynamics of identical units that do not synchronize but converge to distinct equilibria are in general equivalent to the (linear) dynamics of non-identical units. Similar issues are then encountered when trying to infer the exact spectral properties of the network. Fortunately, there exists a statistical approach developed in [2] that allows to get around these limitations in the case of large networks.

1.5 Statistical approach to large networks

Motivation
Each of the Laplacian eigenvalues of a large network taken individually do not significantly impact on its dynamics, which makes them hard to identify them accurately. Moreover, each of them captures only little information about the network structure. For these two reasons, identifying exactly all the eigenvalues of L is not only practically out of reach but also irrelevant in the case of large networks. It is, instead, more convenient and relevant to study statistical measures of the spectral density of the Laplacian matrix, namely its first spectral moments.

They are related to statistical information on the degree distribution of the network vertices and, as so, yield information about its structure. The first spectral moments of L can be, as in the case of exact spectral identification, approximated from sparse measurements of the network dynamics. We will expose how it is achieved in this section. This approach is also well-suited to networks with non-identical units and allows to obtain some information on the underlying unweighted graph.

Spectral moments: reminder
The k^{th} spectral moment of an arbitrary matrix D is defined as

$$\mathcal{M}^{(k)}(D) = \frac{1}{N}\sum_{\lambda\in\sigma(D)}\lambda^k = \frac{1}{N}tr(D^k) \quad k \in \mathbb{N} \tag{1.19}$$

where $tr(D)$ denotes the trace of D and N is the number of rows (or columns) of matrix D.

The first two spectral moments of the Laplacian are worth studying since they are related to the moments of the degree distribution of $\mathcal{G}$. The first one is equal to the mean degree of the graph and, together with the second one, they provide bounds on the quadratic mean distribution (see (1.2) and (1.3)).

1.5.1 Dynamical systems with identical units

Inferring $\mathcal{M}^{(k)}(L)$ from $\mathcal{M}^{(k)}(K)$
Let us assume, for now, that the spectral moments of the (linearized) matrix of the dynamics of the network, K, are known. (We will revert to how they can be estimated later on). The spectral moments of the Laplacian matrix L can be obtained from them as summarized in the following proposition (of which the proof is given in [2]).

Proposition 3. *Suppose that*

$$K = I_n \otimes A - L \otimes BC^T$$

Then the spectral moments of L are given by

$$\mathcal{M}^{(k)}(L) = \frac{(-1)^k}{(C^TB)^k}\left(m\mathcal{M}^{(k)}(K) + \sum_{j=0}^{k-1}\binom{k}{j}(-1)^{j+1}\mathcal{M}^{(j)}(L)tr(A^{k-j}(BC^T)^j)\right)$$

with $\binom{k}{j} = \frac{k!}{j!(k-j)!}$

(as a reminder, m is the number of states describing the local dynamics of each unit). This result holds for both linear and nonlinear dynamics. In the second case, the matrix K is given by (1.15).

We will mainly focus on the first two moments as their meaning is more intuitive. Following from proposition 3, they are given by

$$\mathcal{M}^{(1)}(L) = \frac{-m\mathcal{M}^{(1)}(K) + tr(A)}{C^TB} \tag{1.20}$$

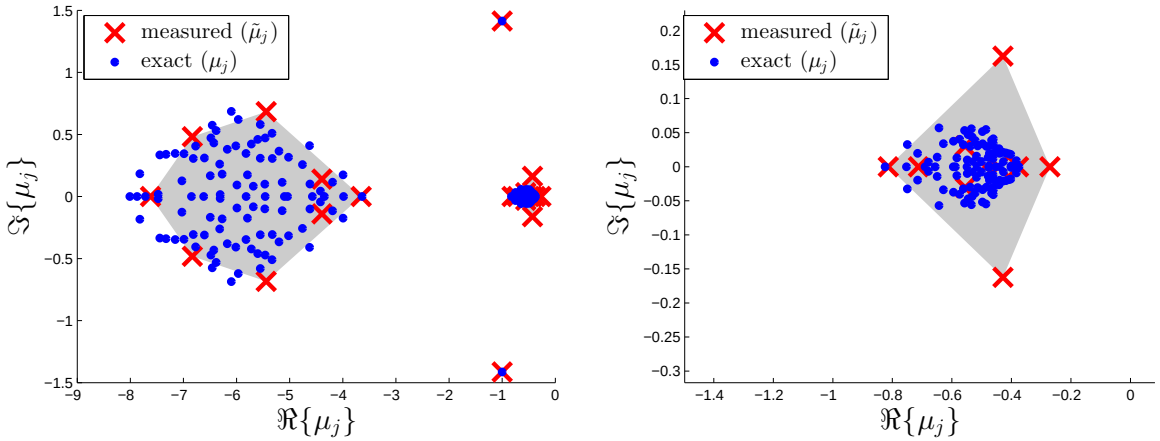
and

$$\mathcal{M}^{(2)}(L) = \frac{m\mathcal{M}^{(2)}(K) - tr(A^2) + 2\mathcal{M}^{(1)}(L)tr(ABC^T)}{(C^TB)^2} \tag{1.21}$$

Estimating $\mathcal{M}^{(k)}(K)$

In the case of large networks, the number of estimated eigenvalues $\tilde{\mu}_j$ of the dynamics matrix K, obtained through the DMD algorithm, is much smaller than the number of its true eigenvalues μ_j (the reason for this will be exposed in section 1.6 (Dynamic Mode Decomposition algorithm)). However, the spectral moments of K can be estimated using a heuristic method.

It is indeed observed that the convex hulls of the clusters of the eigenvalues $\tilde{\mu}_j$ form a good approximation of the clusters of exact eigenvalues μ_j, as illustrated in Figure 1.1. There are typically $n_c = m$ (i.e. the number of states of each dynamical unit) clusters which can be identified using k-means clustering. If you are not familiar with clustering, and more specifically k-means clustering, we refer you to section 2.1 (Supervised learning in machine learning) and [28] first and then to [29].



(a) Entire estimated and exact spectra

(b) Zoom on the cluster lying near the origin

Figure 1.1: Estimation of the eigenvalues of K in the case of a large network using measurements at one vertex. The convex hulls (in gray) of the clusters of measured eigenvalues (red crosses) provide a good approximation of the clusters of exact eigenvalues (blue dots). The local dynamics of each units is characterized by 2 states and there are 2 clusters of estimated eigenvalues.

Once the n_c clusters of the $\tilde{\mu}_j$ have been identified, their respective convex hulls are computed. The spectral moments of K can then be estimated using the moments of area of these convex hulls. Let $\mathcal{S}_c$ $(c = 1, ..., n_c)$ denote the convex hulls of the clusters of measured eigenvalues $\tilde{\mu}_j$. The moments of the area of the convex hulls are given by

$$\mathcal{A}(\mathcal{S}_c) = \int_{\mathcal{S}_c} dxdy \qquad \text{area of the convex hull}$$

$$\mathcal{I}_x(\mathcal{S}_c) = \int_{\mathcal{S}_c} y\, dxdy \qquad \text{first moment of area (with respect to the } x\text{-axis)}$$

$$\mathcal{I}_y(\mathcal{S}_c) = \int_{\mathcal{S}_c} x\, dxdy \qquad \text{first moment of area (with respect to the } y\text{-axis)}$$

$$\mathcal{I}_{xx}(\mathcal{S}_c) = \int_{\mathcal{S}_c} y^2\, dxdy \qquad \text{second moment of area (with respect to the } x\text{-axis)}$$

$$\mathcal{I}_{yy}(\mathcal{S}_c) = \int_{\mathcal{S}_c} x^2\, dxdy \qquad \text{second moment of area (with respect to the } y\text{-axis)}$$

Let an eigenvalue μ in $\mathcal{S}_c$ be assigned to the coordinates $(x, y) = (\Re\{\mu\}, \Im\{\mu\})$. If we knew the

exact spectrum of K, i.e. the eigenvalues μ_j, the first spectral moment of K would be equal to (with $N = |\sigma(K)|$)

$$\begin{aligned}\mathcal{M}^{(1)}(K) &= \frac{1}{N}\sum_{j=1}^{N}\mu_j \\ &= \frac{1}{N}\sum_{j=1}^{N}(\Re\{\mu_j\} + i\Im\{\mu_j\}) \\ &= \frac{1}{N}\sum_{j=1}^{N}\Re\{\mu_j\} + i\underbrace{\frac{1}{N}\sum_{j=1}^{N}\Im\{\mu_j\}}_{0}\end{aligned} \tag{1.22}$$

The second term in the sum is equal to zero given that all complex eigenvalues are complex conjugates so the sum of their imaginary part is zero. We do not know neither the eigenvalues μ_j of K nor their number, N. But the first spectral moment can be approximated using the areas and the moments of area of the convex hulls. We assume that

(1) the exact eigenvalues μ_j are uniformly distributed in the clusters

(2) there is the same number of eigenvalues in each cluster

(3) that the number of μ_j in each cluster is high enough

(4) each eigenvalue and its complex conjugate belong to the same cluster so that the second term in (1.22) is zero for each cluster taken individually

Note that, if $N = \sum_c N_c$ and N_c denotes the number of eigenvalues μ_j in convex hull $\mathcal{S}_c$, $N = n_c N_c$ as (2) is assumed. The first spectral moment can be estimated as follows

Provided assumptions (1) and (3), we have

$$\frac{1}{N_c}\sum_{\mu_j\in\mathcal{S}_c} g(\mu_j) \approx \frac{1}{\mathcal{A}(\mathcal{S}_c)}\int_{\mathcal{S}_c} g(\mu)d\mu \qquad c = 1, ..., n_c \quad \text{for any function } g$$

In particular, choosing g as being the identity function,

$$\frac{1}{N_c}\sum_{\mu_j\in\mathcal{S}_c}\mu_j = \frac{1}{N_c}\sum_{\mu_j\in\mathcal{S}_c}\Re\{\mu_j\} \approx \frac{1}{\mathcal{A}(\mathcal{S}_c)}\int_{\mathcal{S}_c} x dx dy = \frac{\mathcal{I}_y(\mathcal{S}_c)}{\mathcal{A}(\mathcal{S}_c)}$$

Thus,

$$\begin{aligned}\mathcal{M}^{(1)}(K) = \frac{1}{N}\sum_{j=1}^{N}\mu_j &= \sum_{c=1}^{n_c}\frac{1}{N}\sum_{\mu_j\in\mathcal{S}_c}\mu_j = \sum_{c=1}^{n_c}\frac{1}{n_c N_c}\sum_{\mu_j\in\mathcal{S}_c}\mu_j \\ &= \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{1}{N_c}\sum_{\mu_j\in\mathcal{S}_c}\mu_j \\ &\approx \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{\mathcal{I}_y(\mathcal{S}_c)}{\mathcal{A}(\mathcal{S}_c)}\end{aligned}$$

Adopting the same approach for the second spectral moment, we have

$$\begin{aligned}\mathcal{M}^{(2)}(K) &= \frac{1}{N}\sum_{j=1}^{N}\mu_j^2 \\ &= \frac{1}{N}\sum_{j=1}^{N}(\Re\{\mu_j\}+i\Im\{\mu_j\})^2 \\ &= \frac{1}{N}\sum_{j=1}^{N}(\Re\{\mu_j\})^2 + \frac{1}{N}\sum_{j=1}^{N}(i\Im\{\mu_j\})^2 + \underbrace{\frac{1}{N}\sum_{j=1}^{N}2i\Re\{\mu_j\}\Im\{\mu_j\}}_{0} \\ &= \frac{1}{N}\left(\sum_{j=1}^{N}\Re\{\mu_j\}^2 - \sum_{j=1}^{N}\Im\{\mu_j\}^2\right)\end{aligned} \tag{1.23}$$

Again, the third term in (1.23) is equal to zero because two complex conjugate numbers have equal real parts and opposite imaginary ones. Then

$$\begin{aligned}\mathcal{M}^{(2)}(K) = \frac{1}{N}\sum_{j=1}^{N}\mu_j^2 &= \sum_{c=1}^{n_c}\frac{1}{N}\sum_{\mu_j\in\mathcal{S}_c}\mu_j^2 \\ &= \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{1}{N_c}\sum_{\mu_j\in\mathcal{S}_c}(\Re\{\mu_j\}^2 - \Im\{\mu_j\}^2) \\ &\approx \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{\mathcal{I}_{yy}(\mathcal{S}_c)-\mathcal{I}_{xx}(\mathcal{S}_c)}{\mathcal{A}(\mathcal{S}_c)}\end{aligned}$$

In brief,

$$\begin{aligned}\mathcal{M}^{(1)}(K) &\approx \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{\mathcal{I}_{y}(\mathcal{S}_c)}{\mathcal{A}(\mathcal{S}_c)} \\ \mathcal{M}^{(2)}(K) &\approx \frac{1}{n_c}\sum_{c=1}^{n_c}\frac{\mathcal{I}_{yy}(\mathcal{S}_c)-\mathcal{I}_{xx}(\mathcal{S}_j)}{\mathcal{A}(\mathcal{S}_j)}\end{aligned} \tag{1.24}$$

The μ_j are generally not uniformly distributed in the convex hulls so (1.24) is of course an estimation.

Dynamical systems with non-identical units

Dynamical systems with heterogeneous units are such that the local dynamics is not the same at each node. If the local dynamics are randomly distributed and that their mean and variance are known, spectral network identification also applies to these network systems.

In this case, the Jacobian matrix associated with the collective dynamics linearized around its stable fixed point is denoted by $K + \delta K$. Since the measurements are related to the dynamics described by this matrix, the DMD algorithm yields the dominant eigenvalues of $K + \delta K$ and not those of K as for identical units. We can thus estimate the spectral moments of $K + \delta K$ using (1.24). The spectral identification problem in the case of heterogeneous units is thus to infer the spectral properties, in particular the spectral moments, of the Laplacian matrix L from the estimated spectral moments of $K + \delta K$. We will not expose here how it can be achieved as we will not consider this case in the rest of this work, but it is explained in [2].

Note that, in case of unknown local dynamics, it is still possible to estimate the spectral moments of L. It is also detailed in [2].

1.5.2 Spectral moments of the unweighted Laplacian matrix

Thus far, we have analyzed the spectral properties of weighted graphs $\mathcal{G}$. However, the spectral network identification framework allows to infer those of unweighted graphs $\bar{\mathcal{G}}$ as well.

Definition of an unweighted graph
Given a weighted graph $\mathcal{G}$, its associated unweighted graph $\bar{\mathcal{G}}$ is defined as having the same edges and vertices sets as its weighted counterpart but a weight function $\bar{w}(i,j) = 1 \; \forall (i,j) \in E$. The degree $\bar{d}_i$ of a node i of $\bar{\mathcal{G}}$ is its number of connections.

Inferring the spectral properties of $\bar{\mathcal{G}}$ from those of $\mathcal{G}$
As in the case of weighted networks, the spectral moments of the unweighted Laplacian matrix $\bar{L}$, are related to the distribution of this number of connections. The spectral moments of $\bar{L}$ can be approximated from the moments of L as will be exposed in what follows.

Let us say we want to determine the spectral moments of an unweighted graph $\bar{\mathcal{G}}$. We assume that the weights W_{jk} of the corresponding weighted graph $\mathcal{G}$ are independent random variables of mean $r > 0$ and standard deviation $s > 0$, i.e. for all $(j,k) \in E$,

$$\begin{aligned} \mathbb{E}(W_{jk}) &= r \\ \mathbb{E}(W_{jk}^2) &= r^2 + s^2 \\ \mathbb{E}(W_{jk} W_{j'k'}) &= r^2 \quad \forall (j,k) \neq (j',k') \end{aligned} \tag{1.25}$$

In this instance, the Laplacian matrix L can be seen as a random matrix. Provided that the mean r and the standard deviation s are known, the spectral moments of $\bar{L}$ can be estimated from the expectation of the spectral moments of L.

Proposition 4. *If the edges of the network have random weights distributed according to*(1.25)*, then*

$$\begin{aligned} \mathcal{M}^{(1)}(\bar{L}) &= \mathbb{E}\left(\frac{\mathcal{M}^{(1)}(L)}{r}\right) \\ \mathcal{M}^{(2)}(\bar{L}) &= \mathbb{E}\left(\frac{\mathcal{M}^{(2)}(L)}{r^2} - \frac{s\mathcal{M}^{(1)}(L)}{r^3}\right) \end{aligned} \tag{1.26}$$

Moreover, we have

$$Var\left(\frac{\mathcal{M}^{(1)}(L)}{r}\right) = \frac{s^2}{r^2}\frac{\mathcal{D}^{(1)}(\bar{\mathcal{G}})}{n} < \frac{s^2}{r^2}$$

Note that, when $s/r \ll 1$, the variance of $\frac{\mathcal{M}^{(1)}(L)}{r}$ is small and (1.26) is a good estimation of the mean number of connections of the network.

1.5.3 Summary

In this section, we have exposed how to estimate the spectral moments, in particular the first two ones, of the (possibly unweighted) Laplacian matrix from measurements of the network dynamics. The method that should be applied to achieve this depends on the network we are dealing with (is it weighted or not, are the units identical or not?). The general schemes of these methods are summarized below.

Spectral identification for large networks (identical units)

Measure the dynamics at a few states (possibly one)

↓ DMD algorithm

Approximate dominant Koopman eigenvalues and modes

↓ $\sigma(\text{Koopman})=\sigma(K)$

Obtain a small subset of the estimated spectrum $\sigma(K)$

↓ relations (1.24)

Obtain spectral properties of the (linearized) dynamics $(\mathcal{M}^{(k)}(K),\ k=1,2)$

↓ proposition 3

Infer spectral properties of the network $(\mathcal{M}^{(k)}(L),\ k=1,2)$

↓ spectral graph theory

Deduce topological properties of the network $(d_{\text{mean}}, ...)$

Spectral identification for large networks (unweighted graph $\bar{\mathcal{G}}$)

Measure of the dynamics of the weighted network $\mathcal{G}$ at a few states (possibly one)

↓ use the above scheme

Infer spectral properties of the weighted network $(\mathcal{M}^{(k)}(L),\ k=1,2)$

↓ proposition 4

Infer spectral properties of the unweighted network $(\mathcal{M}^{(k)}(\bar{L}),\ k=1,2)$

↓ spectral graph theory

Deduce topological properties of the unweighted network $(d_{\text{mean}}, ...)$

1.6 Dynamic Mode Decomposition algorithm

We will now introduce the method that is used in the framework of spectral network identification to estimate the eigenvalues of the Koopman operator, that have been shown to be the same as the eigenvalues of the matrix of the collective dynamics, K.

1.6.1 Why use DMD in spectral identification

Data-driven algorithms to obtain the Koopman spectral properties

As explained above, computing the Koopman eigenvalues and modes analytically is most of the time out of reach as we generally do not have an explicit representation of the Koopman operator. However, its behavior can be analyzed through its action on an observable, i.e from measurements of the evolution of this observable over time. There exist algorithms allowing to directly estimate the Koopman modes and eigenvalues from these data. One of those techniques is the Dynamic Mode Decomposition (DMD) algorithm.

DMD in fluid field
DMD was originally introduced in the fluid mechanics field, in 2008, and it has not ceased gaining popularity ever since. Its success is due to the fact that it allows to describe the dynamics of complex nonlinear flows [30]. Flows are often analyzed using modal decomposition techniques that expand the fluid state (such as the velocity of vorticity field), which is most of the time a complicated function, into a superposition of simpler components, which are empirically computed basis vectors, called modes. DMD is one of those methods and its strength is its ability to extract the dynamics information from snapshots of the flow, generated either by numerical simulations or measured in a physical experiment. The dynamics information in question yields the so-called DMD modes and eigenvalues that, taken together, describe the dynamics captured in the data sequence. DMD is nowadays used in a broader field of applications, as it provides a powerful mean of analyzing nonlinear dynamics in general.

DMD and Koopman spectral analysis
It has been shown in [31] that DMD is related to the spectral analysis of the Koopman operator, as there exists a mathematical relation between the eigenvalues and modes obtained trough DMD from measurements of an observable and the modes and eigenvalues of the Koopman operator associated to that observable. DMD can thus be considered as a numerical approximation of the Koopman spectral analysis, what will be made clearer in what follows.

1.6.2 Using DMD to approximate the Koopman eigenvalues

The modes and eigenvalues of the Koopman operator can be estimated by computing the best approximation of the Koopman operator on some finite-dimensional subspace and then obtaining modes and eigenvalues of the resulting finite-dimensional linear operator, as will be detailed below.

Measurements and data matrix
As said above, a strong point of the DMD algorithm is that it only requires a sequence of measurements of the dynamics as input. In the case of spectral identification, the dynamics of the network is measured through the observation function $f : \mathbb{R}^{nm} \to \mathbb{R}^p$, for r different initial conditions. If the dynamics is measured over a time interval of length T with fixed sampling period Δt, the measurement at time $s\Delta t$ yields the data point, or snapshot,

$$z_s = \begin{bmatrix} f(X^{(1)}(s\Delta t)) \\ f(X^{(2)}(s\Delta t)) \\ \vdots \\ f(X^{(r)}(s\Delta t)) \end{bmatrix} \in \mathbb{R}^{rp}$$

where $X^{(j)}(t)$ $j = 1, ..., r$ is a trajectory of the system associated with the initial condition $X_0^{(j)}$ and $s = 0, 1, ..., S$ with $S + 1$ being the total number of measurements. From the data points obtained for each measurement, we construct the data matrix

$$Z = \begin{bmatrix} f(X^{(1)}(0)) & f(X^{(1)}(\Delta t)) & f(X^{(1)}(2\Delta t)) & \dots & f(X^{(1)}(S\Delta t)) \\ f(X^{(2)}(0)) & f(X^{(2)}(\Delta t)) & f(X^{(2)}(2\Delta t)) & \dots & f(X^{(2)}(S\Delta t)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(X^{(r)}(0)) & f(X^{(r)}(\Delta t)) & f(X^{(r)}(2\Delta t)) & \dots & f(X^{(r)}(S\Delta t)) \end{bmatrix} \triangleq \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix} \tag{1.27}$$

Finite-dimensional approximation of the Koopman operator
Note that

$$f(X^{(j)}(s\Delta t)) = (U^{s\Delta t} f)(X_0^{(j)})$$

with U denoting the Koopman operator associated to the observation function f. So,

$$z_s = (U^{s\Delta t} f)(X_0) = U^{s\Delta t} f(X_0) = U^{s\Delta t} z_0 \quad \text{where } X_0 \triangleq [X_0^{(1)}\ X_0^{(2)}\ ...\ X_0^{(r)}]^T \in \mathbb{R}^{rnm} \quad (1.28)$$

Given that we consider measurements of the dynamics, we would like to reformulate (1.28) in a discrete time setting. By denoting $U \triangleq U^{\Delta t}$, observe that (1.28) can be rewritten as

$$z_s = U^s z_0 \quad s = 0, ..., S$$

The data matrix (1.27) can thus be reformulated as

$$Z = \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix} = \begin{bmatrix} z_0 & Uz_0 & U^2 z_0 & \dots & U^S z_0 \end{bmatrix}$$

Notice also that

$$z_s = U^s z_0 = U(U^{s-1} z_0) = U z_{s-1} \quad s = 1, ..., S$$

In order to estimate the spectral properties of the infinite-dimensional Koopman operator U, it is assumed that there exists a linear, finite-dimensional operator (its associated matrix being denoted by P) that approximates the action of U restricted to the Krylov subspace spanned by the snapshots z_s [26]. A Krylov subspace is defined as follows [32]:

Definition 1. *Given a matrix P and a vector z_0, the j^{th} Krylov subspace $\mathcal{K}_j(P, z_0)$ generated by P from z_0 is*

$$\mathcal{K}_j(P, z_0) \triangleq span\{z_0, Pz_0, P^2 z_0, ..., P^{j-1} z_0\}$$

In other words, we assume that there exists a matrix P such that

$$z_s \approx P z_{s-1} \quad s = 1, ..., S \quad (1.29)$$

i.e. such that it approximates the action of the Koopman operator on every snapshot of the data matrix Z, and consequently on any linear combination of the snapshots since U is linear. This is equivalent to considering the action of the Koopman operator restricted to the space generated by the snapshots, which span a Krylov subspace, as

$$Z = \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix} \approx \begin{bmatrix} z_0 & Pz_0 & P^2 z_0 & \dots & P^{S-1} z_0 \end{bmatrix}$$

DMD allows to approximate the eigenvalues and eigenvectors of the unknown matrix P, based on the known data z_s. Thereafter, the modes and eigenvalues of U can be estimated from respectively the computed eigenvectors and eigenvalues of P.

DMD algorithm
DMD is a recent method and is continuously being improved so there exist several "versions" of it (see [30, 33–35]). We will consider here the standard one developed in [30] and used in [2], summarized in algorithm 1.

Algorithm 1 Standard DMD

1. Arrange data matrix $Z = \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix}$ into matrices

$$X \triangleq \begin{bmatrix} z_0 & z_1 & \dots & z_{S-1} \end{bmatrix} \quad \text{and} \quad Y \triangleq \begin{bmatrix} z_1 & z_2 & \dots & z_S \end{bmatrix}$$

2. Compute the reduced Singular Value Decomposition of X

$$X = R\Sigma V^*$$

3. Define the matrix

$$\tilde{P} \triangleq R^* Y V \Sigma^{-1}$$

4. Compute the eigenvalues π_j and eigenvectors w_j of $\tilde{P}$, i.e.

$$\tilde{P} w_j = \pi_j w_j$$

5. The DMD modes ν_j corresponding to the DMD eigenvalue π_j are then given by

$$\nu_j = R w_j$$

The DMD modes ν_j and DMD eigenvalues π_j are respectively the eigenvectors and eigenvalues of matrix P in (1.29) and it follows that

$$z_s \approx \sum_{j=1}^{rp} \pi_j^s \nu_j \tag{1.30}$$

Comparing this expression with the expansion of measurements using the spectral properties of the Koopman operator, given by (1.18)

$$f(X(t)) = X^* + \sum_{(j_1,\dots,j_{nm}) \in \mathbb{N}^{nm}} V_{(j_1,\dots,j_{nm})} e^{(j_1\mu_1 + \dots + j_{nm}\mu_{nm})t} \tag{1.31}$$

it can easily be seen that the Koopman modes are equal to the DMD modes, i.e. $V_j = \nu_j$ and that the Koopman eigenvalues can be approximated by $\tilde{\mu}_j = \frac{\log(\pi_j)}{\Delta t}$. As the Koopman operator is infinite-dimensional, it is obvious that not all its eigenvalues and modes can be retrieved by means of the DMD algorithm but only the dominant ones. Using DMD in the framework of spectral identification allows consequently to estimate the dominant eigenvalues of the matrix of the dynamics, K, since those are equal to the dominant Koopman eigenvalues.

Note that, the number of DMD eigenvalues is equal to the number of rows of Z, i.e. rp. The number of exact eigenvalues μ_j of K is equal to nm and, in the case of large networks, is typically much larger than rp.

Increasing the number of estimated Koopman modes and eigenvalues
The number of DMD modes and eigenvalues is equal to the number of rows of the data matrix $Z \in \mathbb{R}^{rp \times S+1}$. If this number is too small, it should be increased in order to improve the efficiency

of the DMD algorithm and compute more eigenvalues. This, in turn, enhances the estimation of the spectral properties of the network.

It can be done without performing additional experiments or simulations, by using the fact that, if $X(t)$ is a trajectory of the system associated with the initial condition $X(0)$, then $X(t+\delta\Delta t)$ is another trajectory associated with the initial condition $X(\delta\Delta t)$. It is therefore possible to construct a modified data matrix with crp rows by considering c sequences of data points shifted in δ increments. The data matrix Z

$$Z = \begin{bmatrix} z_0 & z_1 & z_2 & \dots & z_S \end{bmatrix} \in \mathbb{R}^{rp \times S+1}$$

then becomes

$$\tilde{Z} = \begin{bmatrix} z_0 & z_1 & \dots & z_{S-(c-1)\delta} \\ z_\delta & z_{\delta+1} & \dots & z_{S-(c-2)\delta} \\ \vdots & \vdots & \ddots & \vdots \\ z_{(c-1)\delta} & z_{(c-1)\delta+1} & \dots & z_S \end{bmatrix} \in \mathbb{R}^{crp \times S-(c-1)\delta+1}$$

For instance, if $S = 20$ and we choose $c = 4$ and $\delta = 5$,

$$Z = \begin{bmatrix} z_0 & z_1 & z_2 & \dots & z_{20} \end{bmatrix} \quad \text{becomes} \quad \tilde{Z} = \begin{bmatrix} z_0 & z_1 & \dots & z_{15} \\ z_5 & z_6 & \dots & z_{10} \\ \vdots & \vdots & \ddots & \vdots \\ z_{15} & z_{16} & \dots & z_{20} \end{bmatrix}$$

Note that $\tilde{Z}$ has less columns than Z.

It is obvious that using a single time series and delaying it N times does not provide more information about the network dynamics but, in theory, it allows to compute N approximated eigenvalues $\tilde{\mu}_j$ with one single experiment. Yet, better results are obtained in practice when considering several times series and less delayed snapshots sequences. If the number of rows of Z is too small, then it should be preprocessed that way before performing DMD on it.

1.6.3 Summary

The numerical scheme for spectral network identification is summarized in algorithm 2

Numerical scheme of spectral identification

Algorithm 2 Numerical scheme of spectral identification

1. Choose an observation function f and measure r time series $f(X^{(j)}(t))$

2. Choose the parameters S, Δt and obtain the snapshots z_s from the time series

3. Choose the shift parameters c and δ and construct the data matrix $\tilde{Z}$

4. Apply the DMD algorithm 1 to the data matrix $\tilde{Z}$ and obtain the DMD eigenvalues π_j and the associated Koopman eigenvalues $\tilde{\mu}_j = \frac{\log(\pi_j)}{\Delta t}$

5. Optional: remove outliers, i.e. $\tilde{\mu}_j$ with a large real or imaginary part

6. Optional: if the units are identical, use proposition 1 to obtain the exact Laplacian eigenvalues

7. Use k-means clustering to identify n_c clusters of eigenvalues $\tilde{\mu}_j$ (optional: remove the outliers thereafter)

8. Compute the convex hull of each cluster and its moments of area

9. Estimate the spectral moments of the matrix K using (1.24)

10. If the units are identical, approximate the spectral moments of L using (1.20) and (1.21); if they are non-identical, see [2]

11. If the distribution of the weight of the edges is known, compute the spectral moments of the unweighted Laplacian $\bar{L}$ using proposition 4

1.7 Our goal: improve the estimation of Koopman eigenvalues

The goal of spectral network identification is to infer global information about the topology of a network system from sparse measurements of its dynamics. These information about the network structure can be inferred from the spectral properties of its Laplacian matrix L. Since it depends on the topology of the graph, L is unknown but its spectral properties can be estimated from the spectral properties of the matrix of the collective dynamics of the network, K, through theoretical relations. As K depends on L, it is also unknown but its spectral properties can be inferred from measurements of the dynamics. This is achieved by means of the DMD algorithm. This algorithm yields the dominant Koopman eigenvalues, which have been proven to be equal to the eigenvalues of K.

The estimation of the eigenvalues of K could be improved. Indeed, the DMD algorithm is not always fully reliable and, as it is quite recent, is still being enhanced. Furthermore, this technique has been optimized with respect to the Koopman modes and not to the Koopman eigenvalues [26]. The better the spectral properties of the Laplacian matrix L are estimated, the more accurate the information we get from these about the global structure of the network. As the spectral properties of L are inferred from those of K, enhancing the estimation of the spectrum of K by improving the performances of the DMD algorithm would enhance the spectral identification network results. We will try to achieve this goal using a technique borrowed from the machine learning field, which is called cross-validation.

Our goal

Improve the estimation of the Koopman eigenvalues by enhancing the performance of DMD in the prediction of eigenvalues to obtain better approximations of the spectral properties of the network and so, more accurate information about its global structure. This will be done using machine learning methods and more specifically cross-validation.

Chapter 2

Using cross-validation to improve the DMD algorithm

2.1 Supervised learning in machine learning

The method we developed to enhance the estimation of the Koopman eigenvalues uses the cross-validation method that has been developed in the framework of supervised learning, which is one of the classes of machine learning tasks. We will first briefly introduce machine learning, define supervised learning and then focus on cross-validation.

2.1.1 Brief introduction to machine learning

Algorithms learning from experience
According to [36] and [37], the fundamental challenge addressed in machine learning is the development of algorithms automatically learning from experience. The goal of machine learning is, in other words, to create algorithms that improve at performing some tasks, based on past experience which is represented by observed data. The learning has to be as automatic as possible, i.e. the algorithms should learn as much as possible with as little as possible of human intervention, ideally none at all. In [36], it is formally defined as follows.

Definition 2. *A computer program is said to learn from an experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

If we consider the well-known application of handwriting recognition learning problem,

- the task T is the recognition and classification of handwritten words within images, i.e. the association of a series of grouped handwritten characters with a word such as apple or book
- the performance P is the percentage of words that are correctly classified
- the experience E is a database of handwritten words with a given classification

There exist numerous tasks for which machine learning methods are used, among which speech and image recognition, weather forecasting, detection of fraudulent credit-card transactions and medical diagnosis, to only list a few.

Machine learning and data mining
Machine learning has been receiving more and more interest over the past decades. One of the reasons for this is that it can be used efficiently in data mining. Data mining focuses on the use of computers to discover patterns or relationships in datasets. The amount of available data has not ceased increasing over the past decades so that it is too complicated for a human to analyze them and uncover those patterns by himself. Together with the growth of computing power, machine learning methods are thus a relevant solution to tackle the challenges of data mining. For instance, an insurance company may be collecting information about its customers related to their spending habits or life situation. Using machine learning methods may help the company to make predictions about future events in the clients' life or finding groups of similar customers.

2.1.2 Supervised learning

The cross-validation method has been developed in the context of a specific class of machine learning tasks, referred to as supervised learning ones. We will therefore present this framework (based on [37]) in particular but note that machine learning does not reduce to supervised learning only.

Definition
Supervised learning "refers to any machine learning process that learns a function from an input type to an output type, using data comprising examples that have both input and output values" according to [38]. Put otherwise, the goal is to develop algorithms that determine (or learn) a function g mapping an input value x to an output value y, based on a set of observed data, i.e. pairs of inputs and outputs (x_i, y_i) corresponding to the past experience. Note that both x and y can be vectors.

If y is continuous, the task is referred to as regression and if it is discrete, as classification. Regression is illustrated in Figure 2.1 and classification in Figure 2.2.

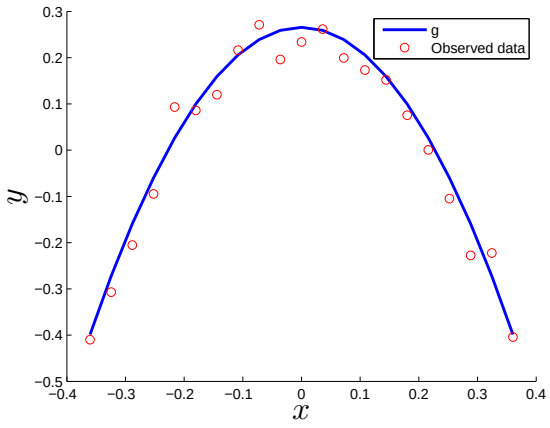


Figure 2.1: Regression: the red dots are the observed pairs (x_i, y_i) that the algorithm has used to determine the function g in blue that maps x to y.

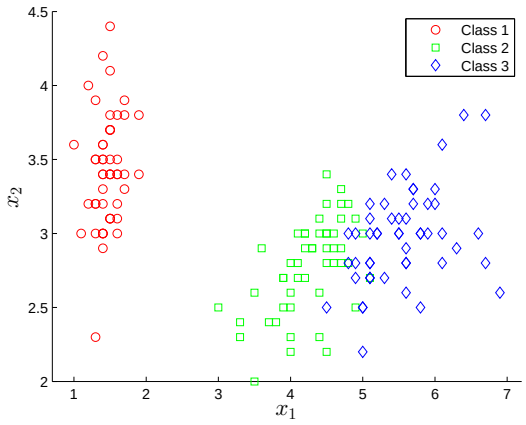


Figure 2.2: Classification: the goal is to determine a function g mapping the inputs $x = (x_1, x_2)$ to the correct discrete output $y = 1, 2, 3$, i.e. the right class (that is, a group of similar observations), based on the observed data (x_i, y_i).

An example of a regression task is determining the weight of a person from its height. The input x is the height and the (continuous) output y is the weight. An instance of a classification task is to predict whether a patient is likely to have a disease based on physiological criterion such as her blood pressure, age, sex and so on. In this case, the input x is a vector whose elements are the physiological properties and the (discrete) output y is *yes* ($y = 1$) or *no* ($y = 0$).

General scheme of supervised learning

Training set X and Y denote the matrices whose rows are respectively the N observed inputs $x_i \in \mathbb{R}^m$ and the N corresponding observed outputs $y_i \in \mathbb{R}^p$, i.e.

$$X \triangleq \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \in \mathbb{R}^{N \times m} \qquad Y \triangleq \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix} \in \mathbb{R}^{N \times p}$$

Taken together, they constitute the so-called training set

$$\mathcal{D}^{train} = (X, Y)$$

that is the set of all available data that the supervised learning method is using to learn. It is also said that the method is trained on this dataset, which explains the use of the word "training".

Training phase and prediction rule Machine learning consists in constructing a method or program that uses the training set $\mathcal{D}^{train}$ to build a function $g : \mathbb{R}^m \to \mathbb{R}^p$ that maps x to y, i.e.

$$y = g(x, w) \tag{2.1}$$

where w is a vector of tunable parameters. Learning amounts to selecting the "best" values for w, based on the training set. Learning the prediction rule g from $\mathcal{D}^{train}$ is referred to as the training phase or learning phase.

Model The method or computer program (along with the assumptions it relies upon) that performs this task, i.e. constructing g from $\mathcal{D}^{train}$, is known as a model, denoted by $\mathcal{M}$. As a reminder, if y is continuous $\mathcal{M}$ is a regression model and if y is discrete, $\mathcal{M}$ is a classification model. Depending on the training set, a same model will construct different prediction rules, i.e. determine different values for the parameters w. There may also exist several models that will all construct different functions g for a same training set.

Test phase Imagine, now, that a new set of observed data, on which the model has not been trained, is available. It is called the test set, $\mathcal{D}^{test}$. As the outputs associated with the inputs in $\mathcal{D}^{test}$ are known, these test observations can be used to assess the accuracy of the predictions made by means of the model. This is referred to as the test phase.

Returning to the instance where we would like to be able to predict the weight of a person from its height, the training set consists of a vector $X \in \mathbb{R}^N$ containing the height of N persons and a vector $Y \in \mathbb{R}^N$ whose elements are their weight. The supervised learning task is to develop a model $\mathcal{M}$ that constructs a prediction rule g expressing the relationship between the height and the weight of a person, based on the available data (X, Y). g can then be used to estimate the weight of people whose height only is known.

Example of a linear regression model
If we consider a linear regression model and a scalar output, y is a linear combination of the m components of the input x. This yields

$$y = g(x, w) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m$$

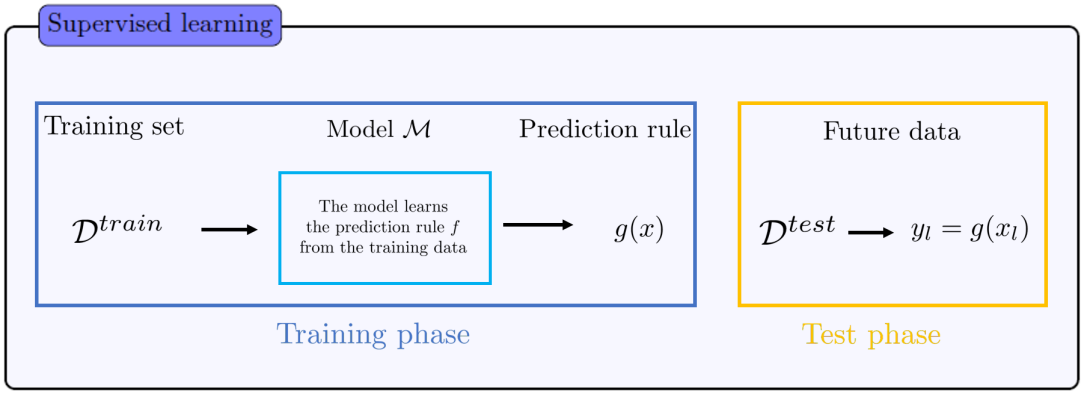
Training this model on $\mathcal{D}^{train}$ amounts to determine the values of the weights w_i such that g fits as good as possible the training data. This can be achieved by using the least squares method, i.e. by solving the optimization problem

$$\min_w \sum_{i=1}^{N} (y_i - g(x_i, w))^2 \tag{2.2}$$

In the rest of this document, we will omit the dependence of f on the parameters w to lighten the notations.

2.1.3 Summary

Supervised learning can be summarized by the following diagram



Other learning tasks
Supervised learning stands in contrast to unsupervised learning and reinforcement learning. In the case of unsupervised learning, only the observed inputs x_i are available, not the associated outputs, and the goal is to detect structure in the data, such as groups of similar observations called clusters.

Reinforcement learning tasks [39] are typically related to autonomous systems (such as robots) evolving in an environment. The agent collects information about its surroundings (by means of sensors for instance), considered as the input x_i. It then has to select an action to execute, based on that input. The agent receives afterwards a reward based on the action it chose. The goal of reinforcement learning is to construct a policy, i.e. a mapping from the state of the environment to an action, maximizing the reward received over time. Unlike supervised learning, there are no available behaviors categorized as correct or incorrect but, contrary to unsupervised learning, a reward is perceived which gives, to some extent, an indication about how relevant the selected action was with respect to the input.

Evaluation of the performance of a model
An important question that has not been discussed yet is the evaluation of the performances of a model, i.e., in the case of supervised learning, its ability to make accurate predictions. That is precisely where cross-validation comes in.

2.2 Cross-validation

2.2.1 Generalization performance

Cross-validation is a method that has been developed, among others, to estimate the performance of a supervised learning model. We will first explain how this performance is measured and then how it can be estimated using cross-validation, based on [37].

Training error
A first idea that could come to one's mind is that the prediction accuracy of a model could be analyzed by quantifying how well the model fits the training data. This measure of the model performance corresponds to the training error

$$\mathcal{E}_{\mathcal{M}}^{train} = \frac{1}{|\mathcal{D}^{train}|} \sum_{i \in \mathcal{D}^{train}} L(y_i, g_{\mathcal{M}}(x_i))$$

where $g_{\mathcal{M}}$ is the model $\mathcal{M}$ fitted to $\mathcal{D}^{train}$ and $L(y, g_{\mathcal{M}}(x))$ is a loss function, i.e. a function computing the error associated to predicting $g_{\mathcal{M}}(x)$ when the exact value is y. An example of such a function is the squared error, $L(y, g_{\mathcal{M}}(x)) = ||y - g_{\mathcal{M}}(x)||^2$.

But there is an important issue related to the use of the training error as performance estimator. Let us consider a dataset of scalar inputs x_i and scalar outputs y_i and three regression models:
$\mathcal{M}_1$ ={first order polynomials}, i.e. $g_{\mathcal{M}_1}(x) = w_0 + w_1 x$
$\mathcal{M}_2$ ={second order polynomials}, i.e. $g_{\mathcal{M}_2}(x) = w_0 + w_1 x + w_2 x^2$
$\mathcal{M}_3$ ={sixth order polynomials}, i.e. $g_{\mathcal{M}_3}(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6$

The first step is to fit them on a same dataset, as is shown in Figure 2.3 and to compute thereafter their associated training error.

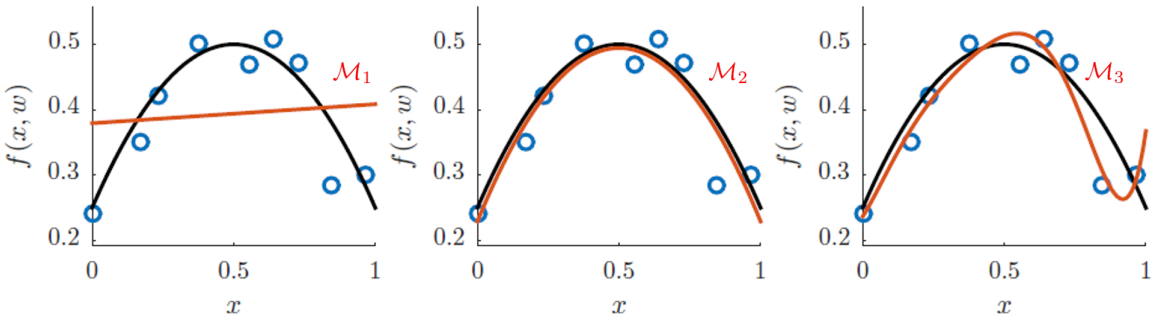


Figure 2.3: Illustration of the issue related to using the training error for evaluating the performance of a model. The blue dots are the observed data generated from the curve in black to which some noise has been added. The three red curves represent the three regression models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. (Source: [37])

$\mathcal{M}_1$ is clearly the worst model. Regarding $\mathcal{M}_3$, it is obvious that it better fits the data and is so associated with the smallest training error. However, it is not close to the function it tries to approximate, i.e. the black curve. $\mathcal{M}_3$ is said to overfit the data, i.e. it "fits the training observations too well". On the contrary, $\mathcal{M}_2$ better account for the true black curve even though it has a larger training error.

Test error

The training error is thus not a good criterion to determine the performance of a model. Furthermore, we are mostly interested in the ability of the model to accurately predict new data (that is data that were not part of the training set). Indeed, being able to make precise predictions only on the training data is of little interest as the final purpose of machine learning is to develop functions g that can generalize to inputs for which the associated output is unknown. The ability of a model to correctly predict data in the test set, or in other words how well the model generalizes to new data, is defined as the generalization performance of the model.

If $\mathcal{D}^{test}$ denotes the test set, i.e. a set of observed data that were not part of the training set, computing the test error of the model on $\mathcal{D}^{test}$ allows us to estimate how well it generalizes to new data.

$$\mathcal{E}_{\mathcal{M}}^{test} = \frac{1}{|\mathcal{D}^{test}|} \sum_{i \in \mathcal{D}^{test}} L(y_i, g_{\mathcal{M}}(x_i))$$

where $g_{\mathcal{M}}$ is the model $\mathcal{M}$ fitted to the training data $\mathcal{D}^{train}$ (not on the test data $\mathcal{D}^{test}$!). The test error of the three above mentioned models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ is illustrated in Figure 2.4.

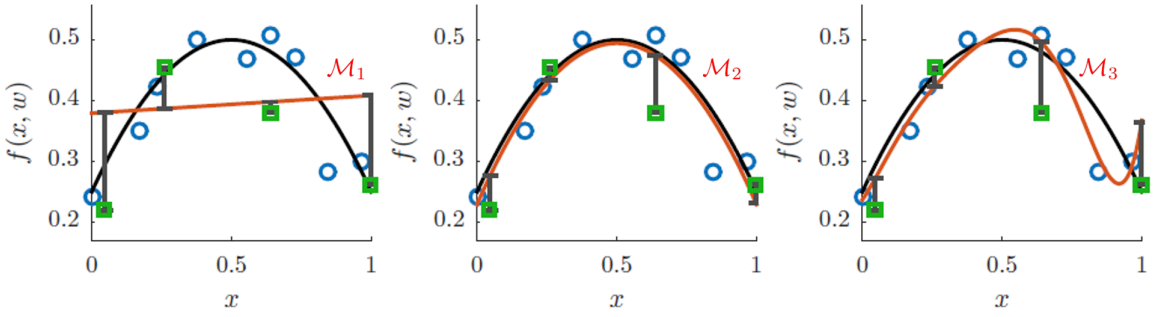


Figure 2.4: Illustration of the test error. The blue dots are the observed data generated from the curve in black to which some noise has been added. The three red curves represent the three regression models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$. The green squares are the test data. (Source: [37])

The test error allows to determine correctly which model better generalizes to new data, i.e. $\mathcal{M}_2$ as it is associated with the smallest test error.

Generalization error

However, the test error depends on the test set which implies that it is to some extent random, as $\mathcal{D}^{test}$ is characterized by some randomness. Let us consider the case in which the model is trained and then tested on a very difficult or unusual test set. The model will then perform poorly, regardless of its quality itself. In this instance, the test error is not representative of the performance of the model.

A third error is considered to get around this problem, namely the generalization error. It is an estimate of the model performance assuming that an infinite amount of test data is available. If

it is assumed that the test observations follow a distribution $p(x, y)$, the generalization error related to a model $\mathcal{M}$ is given by

$$\mathcal{E}_{\mathcal{M}}^{gen} = \mathbb{E}_{(x,y)}[L(y, g_{\mathcal{M}}(x))] = \int L(y, g_{\mathcal{M}}(x))p(x, y)dxdy$$

where $\mathbb{E}_{(x,y)}$ denotes the mathematical expectation. It is quite obvious that the generalization error is an idealized quantity as the distribution of the data is not known. But it can be estimated using the cross-validation framework.

2.2.2 Cross-validation for estimating the generalization error

Basic setup

In practice, the distribution of the data is unknown and the total amount of available observations, $\mathcal{D}(X, Y)$, is limited. The generalization error can consequently only be estimated using those observations. Yet, if the model is trained on $\mathcal{D}$, there is no test set to estimate the generalization performance of the model. $\mathcal{D}$ is thus usually divided into a training set $\mathcal{D}^{train}$ and a test set $\mathcal{D}^{test}$. $\mathcal{E}_{\mathcal{M}}^{gen}$ can be approximated using one of the three following methods.

Hold-out method

A first idea to estimate the generalization error is to use the test error as an approximation. Even if it is not the best estimate, it is still better than using the training error.

In hold-out method, the full dataset is split into a training and a test set as illustrated in Figure 2.5, such that

$$\mathcal{D} = \mathcal{D}^{train} \cup \mathcal{D}^{test}$$

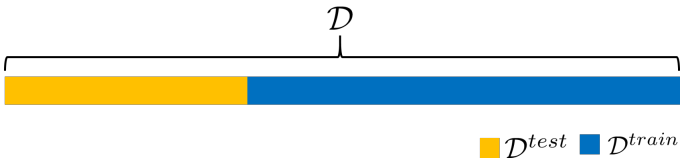


Figure 2.5: Diagram of hold-out cross-validation.

The model is trained on $\mathcal{D}^{train}$ and the test error $\mathcal{E}_{\mathcal{M}}^{test}$ is computed using $\mathcal{D}^{test}$. The generalization error is then approximated by

$$\hat{\mathcal{E}}_{\mathcal{M}}^{gen} = \mathcal{E}_{\mathcal{M}}^{test}$$

K-fold method

A drawback of the hold-out method is that each observation is either in the training set or in the test set. Yet the generalization error would be better estimated if every data point was used in the test set. This can be done by means of another cross-validation method called the K-fold method.

In K-fold cross-validation, the full dataset is divided into K subsets (folds)

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_K$$

each of them containing N/K observations, where N is the number of observations. Then, K splits into training and test sets are produced by considering $\mathcal{D}_k$ as the test set and the remaining K-1 other folds as the training set, for $k = 1, ..., K$, as illustrated in Figure 2.6

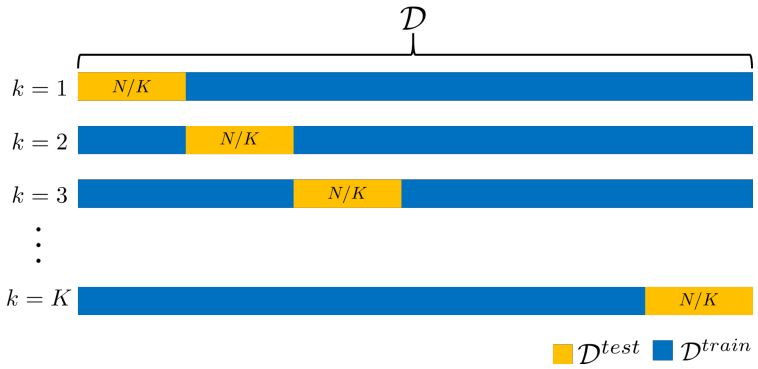


Figure 2.6: Diagram of K-fold cross-validation

The test error associated with each split $\mathcal{E}^{test}_{\mathcal{M},k}$ is computed and the generalization error is approximated by

$$\hat{\mathcal{E}}^{gen}_{\mathcal{M}} = \sum_{k=1}^{K} \frac{N^{test}_k}{N} \mathcal{E}^{test}_{\mathcal{M},k}$$

where N^{test}_k is the number of data points in the test set, i.e. N/K.
Since each observation is used once in the test set, this estimation is more accurate than the hold-out method but it requires K more training and testing steps.

Leave-one-out method
Leave-one-out cross-validation is simply K-fold cross-validation with K equal to N. Each of the N model is thus trained on the full dataset except a single observation on which it is tested, as illustrated in Figure 2.7.

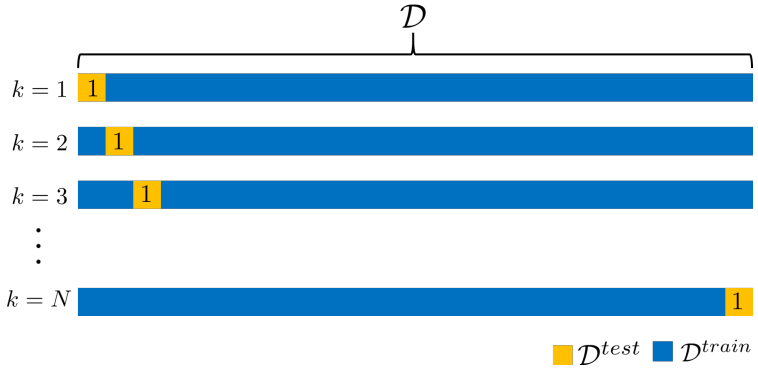


Figure 2.7: Diagram of leave-one-out cross-validation

In this case,

$$\hat{\mathcal{E}}^{gen}_{\mathcal{M}} = \frac{1}{N} \sum_{k=1}^{N} \mathcal{E}^{test}_{\mathcal{M},k}$$

It is the method estimating the most faithfully the generalization error but it requires N training and testing steps.

2.2.3 Cross-validation for model selection

Cross-validation can also be used to select the best model $\mathcal{M}_{s^*}$ among several ones, $\mathcal{M}_1$, $\mathcal{M}_2$,..., $\mathcal{M}_S$, by estimating the generalization error of each model and selecting the model with the

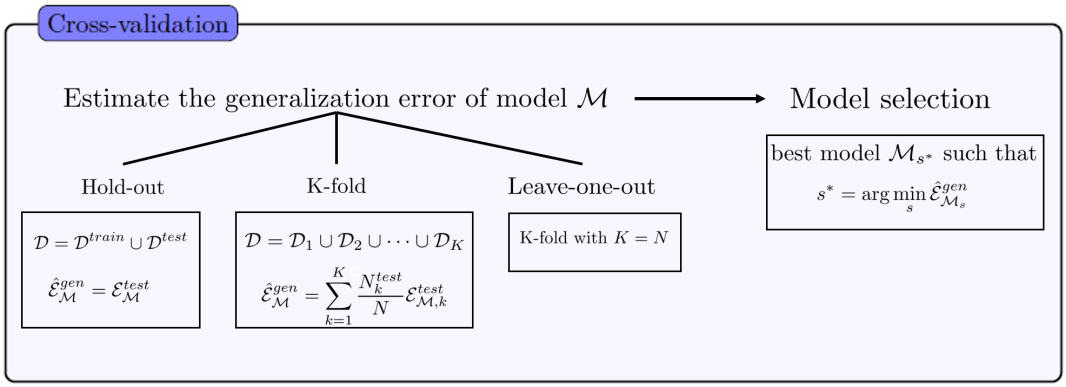
lowest cross-validation error. Put otherwise,

(1) for each model, estimate the generalization error $\hat{\mathcal{E}}^{gen}_{\mathcal{M}_s}$ using cross-validation

(2) select the best model $\mathcal{M}_{s^*}$ such that

$$s^* = \arg\min_s \hat{\mathcal{E}}^{gen}_{\mathcal{M}_s}$$

2.2.4 Summary



2.3 A parallel between the estimation of the dynamics spectrum and supervised learning

Goal of this work: reminder
The goal of this work is to develop a method enhancing the estimation of the eigenvalues of the matrix K, describing the collective dynamics of the network. Improving their approximation would allow to better estimate the spectral properties of the Laplacian matrix L and so, the global information about the topology of the network that can be inferred from those. As the eigenvalues of K have been shown to be the same as the dominant eigenvalues of the Koopman operator (see subsection 1.4.3 (The Koopman operator)) and as those are approximated through the DMD algorithm (see subsection 1.6.2 (Using DMD to approximate the Koopman eigenvalues) and in particular (1.30) and (1.31)), we will try to improve the performance of this algorithm regarding the prediction of the Koopman eigenvalues.

First, it is interesting to notice that a parallel can be drawn between the approximation of the Koopman eigenvalues by means of the DMD algorithm and supervised learning.

2.3.1 Observed data

Koopman operator framework
As exposed above, the collective dynamics of the network system is unknown but it can be measured over time. All the measurements are gathered in the data matrix $Z \in \mathbb{R}^{rp \times S+1}$

$$Z = \begin{bmatrix} f(X^{(1)}(0)) & f(X^{(1)}(\Delta t)) & f(X^{(1)}(2\Delta t)) & \dots & f(X^{(1)}(S\Delta t)) \\ f(X^{(2)}(0)) & f(X^{(2)}(\Delta t)) & f(X^{(2)}(2\Delta t)) & \dots & f(X^{(2)}(S\Delta t)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(X^{(r)}(0)) & f(X^{(r)}(\Delta t)) & f(X^{(r)}(2\Delta t)) & \dots & f(X^{(r)}(S\Delta t)) \end{bmatrix} \triangleq \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix}$$

where $f : \mathbb{R}^{nm} \to \mathbb{R}^p$ is the observation function through which the dynamics of the system is measured, for r different initial conditions. $X^{(j)}(t)$ $j = 1, ..., r$ is a trajectory of the system associated with the initial condition $X_0^{(j)}$, Δt is the fixed sampling period and $S + 1$ is the total number of measurements. The columns of Z are also referred to as snapshots of the dynamics.

Supervised learning framework
Note that the analytical expression of $X(t)$ and f can be unknown but that we know the value of the sampling interval (as it is fixed by the one taking the measures) and the values of the measurements z_s. Hence, we know that z_s has been observed at time $t = s\Delta t$ if we define the time at which the first measurement has been taken to be 0. Considering this from the point of view of supervised learning, the values of time at which the measures have been taken, $s\Delta t$ $(s = 0, 1, ..., S)$, correspond to the inputs x_i and the associated measurements z_s to the outputs y_i. Denoting by

$$T = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ \vdots \\ t_S \end{bmatrix} \triangleq \begin{bmatrix} 0 \\ \Delta t \\ 2\Delta t \\ \vdots \\ S\Delta t \end{bmatrix}$$

the available observations are

$$\mathcal{D} = (T, Z^T) \tag{2.3}$$

2.3.2 Model and prediction rule

Imagine that one would like to build a function predicting the dynamics of the network system for values of time that differ from those at which the measurements were taken. This can be achieved by using observed data, i.e. the snapshots of the dynamics, as will be exposed thereafter. In this sense, a parallel can be drawn between the approximation of the Koopman eigenvalues and supervised learning even if the DMD algorithm is not a supervised learning method.

Koopman operator framework
As a reminder, using the framework of the Koopman operator allows to expand the measurements of the dynamics as follows

$$f(X(t)) = X^* + \sum_{(j_1,...,j_{nm}) \in \mathbb{N}^{nm}} V_{(j_1,...,j_{nm})} e^{(j_1\mu_1 + ... + j_{nm}\mu_{nm})t}$$

where

- X^* is the stable fixed point towards which all the units converge
- $j_1\mu_1 + ... + j_{nm}\mu_{nm}$ are the eigenvalues of the Koopman operator associated with f
- $V_{(j_1,...,j_{nm})}$ are the corresponding Koopman modes

In the case of linear local dynamics, X^* is equal to zero and the number of modes and eigenvalues is finite if f is linear. If we use the following notation

$$\tilde{f}(\tilde{X}(t)) \triangleq \begin{bmatrix} f(X^{(1)}(t)) \\ f(X^{(2)}(t)) \\ \vdots \\ f(X^{(r)}(t)) \end{bmatrix} \quad \text{and} \quad \tilde{X}^* \triangleq \begin{bmatrix} X^{(1)^*} \\ X^{(2)^*} \\ \vdots \\ X^{(r)^*} \end{bmatrix}$$

where $X^{(j)^*}$ is the stable fixed point towards which the trajectory $X^{(j)}(t)$ converges, then

$$\tilde{f}(\tilde{X}(t)) = \tilde{X}^* + \sum_{(j_1,...,j_{nm})\in\mathbb{N}^{nm}} \tilde{V}_{(j_1,...,j_{nm})} e^{\tilde{\mu}_j t} \quad \text{with } \mu_j \triangleq (j_1\mu_1 + ... + j_{nm}\mu_{nm})$$

where $\tilde{V}_{(j_1,...,j_{nm})}$ and $\tilde{\mu}_j$ are respectively the modes and eigenvalues of the Koopman operator associated with the observation function $\tilde{f}$. As the number of Koopman modes and eigenvalues is infinite and as the DMD algorithm only yields the rp dominant ones, given that $z_s = \tilde{f}(\tilde{X}(s\Delta t))$, the measurements z_s can be approximated by

$$\hat{z}_s = \sum_{j=1}^{rp} \hat{V}_j e^{\hat{\mu}_j s\Delta t} \tag{2.4}$$

where $\hat{V}_j$ and $\hat{\mu}_j$ are respectively the Koopman modes and eigenvalues obtained through DMD. ($\tilde{X}^*$ is taken into account in the estimated modes $\hat{V}_j$).

Supervised learning framework
From the point of view of machine learning, (2.4) is the model $\mathcal{M}$ that allows to obtain a prediction function g that maps time to the value of the observation function, i.e.

$$z = g(t) = \sum_{j=1}^{rp} \hat{V}_j e^{\hat{\mu}_j t} \tag{2.5}$$

where the Koopman modes $\hat{V}_j$ and eigenvalues $\hat{\mu}_j$ correspond to the tunable parameters in (2.1). The learning, i.e. the computation of the $\hat{V}_j$ and $\hat{\mu}_j$, is done by means of the DMD algorithm from the observed data $\mathcal{D} = (T, Z^T)$. (The DMD algorithm does not require to know the values of time at which the measurements z_s have been made but those will be needed in what follows).

2.4 Our method

Why use supervised learning and cross-validation
Note that, in the framework of spectral network identification, we are not interested in the prediction function g itself but in the approximated eigenvalues $\hat{\mu}_j$ obtained through DMD, considered as the parameters of the model. And our goal is neither using cross-validation to select between several models as we know which one should be used, i.e. (2.5), nor estimating its generalization error.
The question then arises as to why consider the estimation of the eigenvalues of the dynamics μ_j in regards to supervised learning and cross-validation. The answer is exposed just below together with the main idea of the method we developed to improve the DMD algorithm.

2.4.1 Main idea behind our approach

Generating several sets of estimated eigenvalues
We are seeking to develop a method enhancing the estimation of the eigenvalues of the dynamics of the network. These are approximated by means of the DMD algorithm from measurements of the dynamics. However, this algorithm has been developed to optimize the Koopman modes and not the eigenvalues. A key idea to overcome this issue is that several sets of approximated eigenvalues $\hat{\mu}_j$, obtained through the DMD algorithm, would allow to select the best one among them and so, to potentially enhance the estimated eigenvalues of the dynamics.

In the framework developed in [2], one single set of eigenvalues $\hat{\mu}_j$ is estimated by performing the DMD algorithm on the entire set of measurements $Z = [z_0\ z_1\ ...\ z_S]$. Since, in real applications, the amount of available measurements is often limited, the same dataset Z should be used several times to obtain the above mentioned sets of eigenvalues.

The method we developed is based on K-fold cross-validation and model selection through cross-validation. In order to obtain several sets of eigenvalues $\hat{\mu}_j$, the set of measurements Z is divided into K disjoint subsets Z_k $(k = 1, ...K)$ such that

$$Z = Z_1 \cup Z_2 \cup ... \cup Z_K$$

which amounts to split the dataset $\mathcal{D}$ defined in (2.3) as follows

$$\mathcal{D} = \Big(\mathcal{D}_1 = (T_1, Z_1^T)\Big) \cup ... \cup \Big(\mathcal{D}_K = (T_K, Z_K^T)\Big)$$

Then for each value of k, a set of eigenvalues

$$\hat{\sigma}_k \triangleq \{\hat{\mu}_{j,k}\}$$

and corresponding modes $\{\hat{V}_{j,k}\}$ are obtained by performing the DMD algorithm on the set of measurements Z from which the k^{th} subset Z_k has been removed. From the cross-validation viewpoint, the model (2.5) is trained on the training set

$$\mathcal{D}^{train} = \Big(T \setminus T_k, Z^T \setminus Z_k^T\Big) \tag{2.6}$$

and values are obtained for its parameters, $\hat{\mu}_j$ and $\hat{V}_j$. The associated prediction is denoted by g_k and is given by

$$g_k(t) = \sum_{j=1}^{rp} \hat{V}_{j,k} e^{\hat{\mu}_{j,k} t}$$

Selecting the best estimated spectrum

Even if we stated above that we were not interested in the prediction function per se since our goal is not to predict the dynamics, we still need it. Indeed, selecting the best estimated spectrum requires to have a criterion enabling to compare the K estimated spectra $\hat{\sigma}_k$ and decide which one is the best. As the exact spectrum $\sigma(K)$ is unknown, we need to define a selection criterion based on the available information, i.e. $\mathcal{D}$ (2.3). The one we chose is the test error on the removed subset of measurements

$$\mathcal{D}^{test} = \Big(T_k, Z_k^T\Big) \tag{2.7}$$

and is given by

$$E_{\hat{\sigma}_k} = \frac{1}{|Z_k|} \sum_{s \in \mathcal{K}} L(z_s, g_k(s\Delta t)) = \frac{1}{|Z_k|} \sum_{s \in \mathcal{K}} ||z_s - g_k(s\Delta t)||^2 \qquad \text{where } \mathcal{K} = \{s | z_s \in Z_k\} \tag{2.8}$$

where $||.||$ denotes the 2-norm. It means that the spectrum $\hat{\sigma}_k$ associated with the smallest error $E_{\hat{\sigma}_k}$ is considered to be the best one. Defining the loss function L to be the squared error is a common choice in cross-validation. (The choice of this criterion explains why the time matrix T has to be included in the dataset $\mathcal{D}$ in (2.3)).

The choice of this selection criterion is based on the intuition that a "good" estimated spectrum should be able to obtain a more accurate prediction of the dynamics than a "bad" one. It is to some extent related to the generalization error, except that it does not estimate the generalization performance of a model, i.e. of (2.5), but rather the generalization performance of this model fitted to $\mathcal{D}^{train}$ given in (2.6), i.e. of the prediction function g_k.

2.4.2 Similarities and differences with cross-validation

Comparison with K-fold cross-validation
Even if this method has been inspired from cross-validation, it is not equivalent to one of the three above presented methods used to estimate the generalization error of a model (hold-out, K-fold and leave-one-out methods) as it is not its goal. However, we seek to estimate the generalization performance of the prediction function g_k, i.e. of the model g (2.5) fitted to the training set defined by (2.6). It can thus be to some extent related to K-fold cross-validation. Indeed, the entire dataset $\mathcal{D}$ is divided into K subsets $\mathcal{D}_k$ and, for each value of k, it is split into a training set $\mathcal{D}^{train}$ and a test set $\mathcal{D}^{test}$ the same way it is done in K-fold cross-validation. The model g is then trained on $\mathcal{D}^{train}$, yielding the estimated spectrum $\hat{\sigma}_k$ and associated modes $\{\hat{V}_{j,k}\}$ of the dynamics. The test error associated with g_k, $E_{\hat{\sigma}_k}$, is thereafter computed on $\mathcal{D}^{test}$. So far, the scheme is the same as for K-fold cross-validation. Yet, the K test errors $E_{\hat{\sigma}_k}$ are not used to estimate the generalization error of the model g but to compare the estimated spectra $\hat{\sigma}_k$ and the one associated with the smallest test error is selected as the best one.

Comparison with cross-validation for model selection
From this perspective, our approach seems to resemble to the one consisting in using cross-validation for model selection (see subsection 2.2.3 (Cross-validation for model selection)). Again, it is not exactly the same strategy as we do not have to choose one model among several ones. But we have to select an estimated spectrum among K ones, i.e. we have to choose a set of values of the parameters of the prediction function among K sets of values. The way it is achieved is similar to the scheme of model selection using cross-validation as it can be very briefly summarized by

(1) split the set of measurements and associated values of time gathered in $\mathcal{D}$ into K folds, $\mathcal{D}_k$

(2) for $k = 1, ..., K$, estimate $\hat{\sigma}_k$ and the test error $E_{\hat{\sigma}_k}$ using hold-out cross-validation with $\mathcal{D}^{train}$ and $\mathcal{D}^{test}$ as in (2.6) and (2.7).

(3) select the best spectrum $\hat{\sigma}_{k^*}$ such that

$$k^* = \arg\min_k E_{\hat{\sigma}_k}$$

2.4.3 Splitting the measurements into folds

Leave-d-out instead of K-fold cross-validation
In the rest of this work, the dataset $\mathcal{D}$ will not be split into folds the same way as in K-fold cross-validation (it does not change anything to the general scheme we exposed just above, only the splitting procedure is different). We will rather use an approach similar to leave-one-out cross-validation, called the leave-d-out cross-validation. In this method, instead of considering only one observation in the test set, d of them are used.

Note that the main ideas behind K-fold and leave-d-out methods are equivalent but both methods differ in that the parameter determining how the observations are divided into folds is not the same. In the case of K-fold, we choose the number of folds, K, which determines the number of data points in each fold. On the contrary, if leave-d-out is used, we choose the number of data points in each fold, d, which defines the number of folds. As we would like to study the influence of the size of the test set, we prefer to fix the value of d, rather than the one of K. Given the number of measurements is $S + 1$, leave-d-out method is equivalent to K-fold cross-validation with $K = (S + 1)/d$. If $S + 1$ is not a multiple of d, we chose to take $K = \lfloor (S + 1)/d \rfloor$ folds

(where $\lfloor a \rfloor$ denotes the closest, lower integer to a) and not to consider $S + 1 - \lfloor (S+1)/d \rfloor$ of the observations when splitting the dataset into K folds but still use them in the training set.

Potential issue related to random split
We first thought about randomly selecting the d observations in each fold but it appeared that it might be problematic in some cases. Indeed, the measurements are not independent from each other since, as exposed in subsection 1.6.2 (Using DMD to approximate the Koopman eigenvalues),

$$z_s = U z_{s-1}$$

with $U \triangleq U^{\Delta t}$ the Koopman operator associated with the observation function $\tilde{f}$. The DMD approach assumes the existence of a matrix $P \in \mathbb{R}^{rp \times rp}$ such that

$$z_s \approx P z_{s-1}$$

We will illustrate the issue that could potentially arise using the following example in which we consider a discrete dynamical system such that

$$x_{t+1} = A x_t$$

For the sake of simplicity, we assume that A has l distinct eigenvalues α_j such that $|\alpha_1| \geq |\alpha_2| \geq ... \geq |\alpha_l|$. This implies that their associated eigenvectors v_j are linearly independent and form a basis of $\mathbb{R}^l$. The initial condition x_0 can thus be rewritten as

$$x_0 = \sum_{j=1}^{l} c_j v_j \qquad \text{with } c_j \in \mathbb{R} \;\; j = 1, ..., l$$

As $x_t = A x_{t-1} = A^t x_0$,

$$x_t = A^t \sum_{j=1}^{l} c_j v_j = \sum_{j=1}^{l} c_j (A^t v_j) = \sum_{j=1}^{l} c_j \alpha_j^t v_j$$

Note that

$$x_t = \alpha_1^t \left(c_1 v_1 + c_2 \left(\frac{\alpha_2}{\alpha_1} \right)^t v_2 + ... + c_l \left(\frac{\alpha_l}{\alpha_1} \right)^t v_l \right)$$

This expression allows to study the asymptotic behavior of the system, i.e. when $t \to \infty$. Indeed, as α_1 is larger in absolute value than the other eigenvalues, it follows that each of the

$$\frac{\alpha_j}{\alpha_1} \quad j = 2, ..., l$$

is less than one in absolute value, which, in turns, implies that

$$\left(\frac{\alpha_j}{\alpha_1} \right)^t \to 0 \;\; \text{when } t \to \infty \qquad j = 2, ..., l$$

This means that the approximation

$$x_t \approx \alpha_1^t c_1 v_1$$

is valid for large values of t. It yields that the influence of α_1 on the dynamics is much larger than the influence of the other eigenvalues for large values of t.

If, by accident, the measurements z_s are split into folds such that the i^{th} fold contains the d last measurements, then our selection method is likely not to choose the best spectrum. Indeed, if $S+1$ and Δt are large enough, all the snapshots in the test set $\mathcal{D}^{test} = (T_i, Z_i^T)$ would be mostly influenced by the dominant eigenvalue of the dynamics which implies that $E_{\hat{\sigma}_i}$ would only evaluate the "quality" of the dominant estimated eigenvalue and not of the other $\hat{\mu}_{j,i}$. Consequently, if the dominant approximated eigenvalue allows to predict very well the last d snapshots but that the rest of the estimated eigenvalues are of "bad quality", $\hat{\sigma}_i$ might still be selected even though it is not be the best spectrum as it would perform poorly in predicting other measurements.

Splitting procedure
In order to prevent this specific case, we developed the splitting procedure presented in algorithm 3.

Algorithm 3 Splitting Procedure

0. Choose a value for d

1. Divide the data matrix Z into d subsets of $l = \lfloor \frac{S+1}{d} \rfloor$ measurements such that each subset G_i contains the measurements

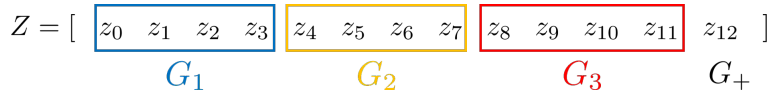
$$G_i = [z_{(i-1)\cdot l}\ z_{(i-1)\cdot l+1}\ ...\ z_{i\cdot l-1}] \qquad i = 1, ..., d$$

If $S+1$ is not a multiple of d, do not include the last $S+1-\lfloor \frac{S+1}{d} \rfloor$ observations in any of the G_i but store them in G_+.

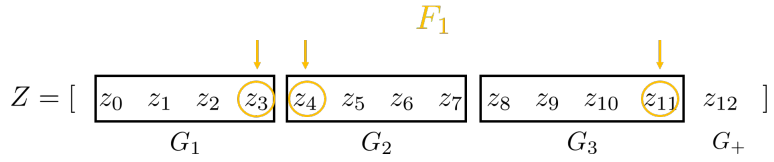
2. Construct l disjoint folds F_k containing d observations each, by randomly selecting one observation in each of the G_i and grouping them into F_k for $k = 1, 2, ..., l$. Define $F_+ \triangleq G_+$

3. Construct the datasets $\mathcal{D}_k = \left(T_k, F_k^T\right)$ $(k = 1, 2, ..., l)$ and $\mathcal{D}_+ = \left(T_+, F_+^T\right)$, where T_k contains the value of time at which the measurements in F_k have been taken.

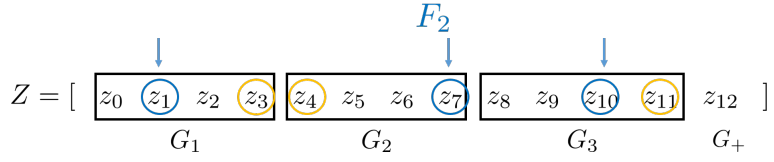
An example of this procedure is given in Figure 2.8 for $S+1 = 13$ and $d = 3$.



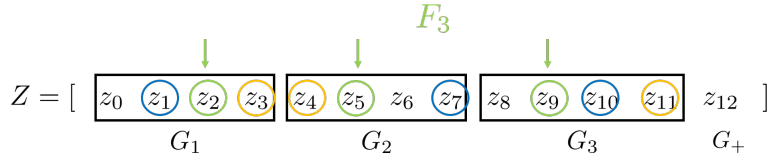
(a) 1. Dividing the measurements into $d = 3$ subsets G_i of size $l = \lfloor \frac{13}{3} \rfloor = 4$. Note that $S + 1 = 13$ is not a multiple of $d = 3$ so the last measurement z_{12} is not in any of the G_i but in G_+.



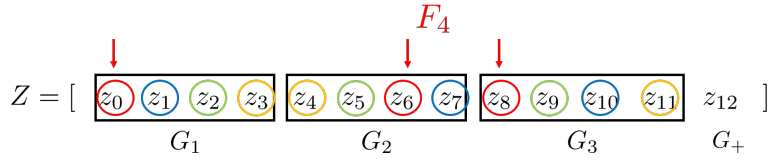
(b) 2. Randomly selecting 1 measurement in each of the G_i to form fold F_1.



(c) 2. Randomly selecting 1 measurement in each of the G_i to form fold F_2.



(d) 2. Randomly selecting 1 measurement in each of the G_i to form fold F_3.



(e) 2. Randomly selecting 1 measurement in each of the G_i to form fold F_5.

$\mathcal{D}_1 = ((t_3, z_3), (t_4, z_4), (t_{11}, z_{11}))$ $\quad$ $\mathcal{D}_3 = ((t_2, z_2), (t_5, z_5), (t_9, z_9))$

$\mathcal{D}_2 = ((t_1, z_1), (t_7, z_7), (t_{10}, z_{10}))$ $\quad$ $\mathcal{D}_4 = ((t_0, z_0), (t_6, z_6), (t_8, z_8))$

$\mathcal{D}_+ = ((t_{12}, z_{12}))$

(f) 3. Datasets $\mathcal{D}_k$ and $\mathcal{D}_+$

Figure 2.8: Illustration of the splitting procedure exposed in algorithm 3

2.4.4 Slight modification of the DMD algorithm 1

As snapshots of the dynamics are removed before performing the DMD algorithm on the measurements, step 1 in the standard DMD algorithm 1 has to be slightly adapted. Indeed, this algorithm approximates the eigenvectors and associated eigenvalues of the matrix P such that

$$z_s \approx P z_{s-1}$$

where the column vectors z_s are the snapshots of the dynamics, gathered in the data matrix Z. One of the assumptions of this DMD algorithm is that the snapshots form an ordered sequence and are separated by a constant sampling time Δt. If we remove columns from the data matrix Z and then perform the DMD algorithm on the so obtained modified matrix, this hypothesis will no longer hold, since some pairs of snapshots will be separated by $2\Delta t$.

As a reminder, step 1 in algorithm 1 consists in arranging the data matrix $Z = \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix}$ into matrices

$$X \triangleq \begin{bmatrix} z_0 & z_1 & \dots & z_{S-1} \end{bmatrix} \quad \text{and} \quad Y \triangleq \begin{bmatrix} z_1 & z_2 & \dots & z_S \end{bmatrix}$$

such that each column j of Y, $Y_{:,j}$, is related to the j^{th} column of X, $X_{:,j}$, by the relationship $Y_{:,j} = PX_{:,j}$, which is equivalent to state that they are separated by the fixed sample time Δt. To ensure that this is verified, the matrices X and Y are constructed in our method as detailed in algorithm 4.

Algorithm 4 Modification of step 1 of the standard DMD algorithm 1

Input: $Z = [z_0\ z_1\ ...\ z_S]$ the matrix of the measurements of the dynamics
$\mathcal{K} = \{s | z_s \in Z_k\}$ where Z_k is the subset of columns of Z to be removed
Output: X and Y, the matrices obtained in step 1 of the standard DMD algorithm 1

1. Arrange the data matrix $Z = \begin{bmatrix} z_0 & z_1 & \dots & z_S \end{bmatrix}$ into matrices

$$\tilde{X} \triangleq \begin{bmatrix} z_0 & z_1 & \dots & z_{S-1} \end{bmatrix} \quad \text{and} \quad \tilde{Y} \triangleq \begin{bmatrix} z_1 & z_2 & \dots & z_S \end{bmatrix}$$

for $j \in \mathcal{K}$ **do**
2. Remove the j^{th} column of $\tilde{X}$, $\tilde{X}_{:,j}$, and the j^{th} column of $\tilde{Y}$, $\tilde{Y}_{:,j}$
end for

3. Define $X \triangleq \tilde{X}$ and $Y \triangleq \tilde{Y}$

(Step 2 in algorithm 4 assumes that the numbering of the columns of $\tilde{X}$ and $\tilde{Y}$ starts at 0.) An example of algorithm 4 is shown in Figure 2.9 for $S + 1 = 13$, $d = 3$ and $\mathcal{K} = \{1,\ 6,\ 8\}$.

$$Z = \begin{bmatrix} z_0 & z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 & z_9 & z_{10} & z_{11} & z_{12} \end{bmatrix}$$

(a) Input: data matrix Z

$$\tilde{X} = \begin{bmatrix} z_0 & \cancel{z_1} & z_2 & z_3 & z_4 & z_5 & \cancel{z_6} & z_7 & \cancel{z_8} & z_9 & z_{10} & z_{11} \end{bmatrix}$$
$$\tilde{Y} = \begin{bmatrix} z_1 & \cancel{z_2} & z_3 & z_4 & z_5 & z_6 & \cancel{z_7} & z_8 & \cancel{z_9} & z_{10} & z_{11} & z_{12} \end{bmatrix}$$

(b) 1. Arranging Z into the matrices $\tilde{X}$ and $\tilde{Y}$. 2. Removing the columns z_z such that $s \in \mathcal{K} = \{1,\ 6,\ 8\}$

$$X = \begin{bmatrix} z_0 & z_2 & z_3 & z_4 & z_5 & z_7 & z_9 & z_{10} & z_{11} \end{bmatrix}$$
$$Y = \begin{bmatrix} z_1 & z_3 & z_4 & z_5 & z_6 & z_8 & z_{10} & z_{11} & z_{12} \end{bmatrix}$$

(c) 3. Matrices X and Y

Figure 2.9: Illustration of the adaptation of the standard DMD algorithm exposed in algorithm 4

To ensure that each of the snapshots in $\mathcal{K}$ is effectively removed, we do not allow the last column of Z, z_S, to be part of any of the test sets in algorithm 3. It would indeed not be discarded by algorithm 4 given that it is not part part of X. Note that one could choose to discard the snapshots z_s such that $s \in \mathcal{K}$ from matrices $\tilde{X}$ and $\tilde{Y}$ instead of removing them only from $\tilde{X}$ but this would lead to loose too much information about the dynamics of the system. Indeed, in the worst case, it would amount to eliminate $2d$ snapshots from Z.

2.4.5 Summary

The method we developed to enhance the estimation of the Koopman eigenvalues obtained through the DMD algorithm is summarized in algorithm 5.

Cross-validation to improve the estimation of the Koopman eigenvalues

Algorithm 5 Cross-validation to improve the estimation of the Koopman eigenvalues

Input: $\mathcal{D} = (T, Z^T)$ with $Z = [z_0\ z_1\ ...\ z_S]$ the matrix of the measurements of the dynamics
T the times at which these have been taken
d, the number of measurements z_s in the test set
Output: $\hat{\sigma}^*(K)$, the best estimated spectrum of the dynamics matrix K

0. Split $\mathcal{D} = (T, Z^T)$ into $l = \lfloor \frac{S+1}{d} \rfloor$ folds $\mathcal{D}_k = (T_k, Z_k^T)$ $(k = 1, .., l)$ and $\mathcal{D}_+ = (T_+, Z_+^T)$ using algorithm 3

for $k = 1...l$ **do**

1. Set $\mathcal{D}^{train} \triangleq \mathcal{D} \setminus \mathcal{D}_k$ and $\mathcal{D}^{test} \triangleq \mathcal{D}_k$

2. Perform the DMD algorithm on Z from which the snapshots in Z_k are removed, by running algorithm 4 and then steps 2-5 of algorithm 1. Obtain so the estimated Koopman modes $\{\hat{V}_{j,k}\}$ and associated eigenvalues $\hat{\sigma}_k \triangleq \{\hat{\mu}_{j,k}\}$

3. Construct the prediction rule

$$g_k(t) = \sum_{j=1}^{rp} \hat{V}_{j,k} e^{\hat{\mu}_{j,k} t}$$

and compute the prediction error

$$E_{\hat{\sigma}_k} = \frac{1}{|Z_k|} \sum_{s \in \mathcal{K}} ||z_s - g_k(s\Delta t)||^2 \qquad \text{where } \mathcal{K} = \{s | z_s \in Z_k\}$$

end for

4. Select the best spectrum $\hat{\sigma}_{k^*}$ such that

$$k^* = \arg\min_k E_{\hat{\sigma}_k}$$

$$\hat{\sigma}^*(K) \triangleq \hat{\sigma}_{k^*}$$

2.4.6 Some comments

Value of d

First of all, note that d cannot be larger than the difference between the number of columns of Z, $S+1$, and its number of rows, rp, minus 1. Indeed, as a reminder (see section 1.6 (Dynamic Mode Decomposition algorithm)), the DMD algorithm seeks to estimate the matrix P such that

$$z_{s+1} \approx P z_s \quad s = 0, 1, ..., S-1$$

or, equivalently its goal is to solve the following system

$$Y = PX \quad \text{with } X \triangleq [z_0\ z_1\ ...\ z_{S-1}] \text{ and } Y \triangleq [z_1\ z_2\ ...\ z_S] \tag{2.9}$$

with $P \in \mathbb{R}^{rp \times rp}$, and $Y, X \in \mathbb{R}^{rp \times S}$. At each iteration of algorithm 5, d snapshots are discarded and the DMD algorithm is then performed on the modified matrices X and Y obtained by means of algorithm 4. The dimensions of P do not change, which implies that the number of estimated Koopman eigenvalues and modes does not vary, it is always equal to rp. However, the dimensions of X and Y depend on the value of d, i.e. their number of rows remains constant regardless of d but their number of columns is equal to $S-d$. If the value of d is chosen such that $S - d < rp$, then the system (2.9) becomes underdetermined. The value of d should therefore always be smaller than

$$d_{\max} = S - rp \tag{2.10}$$

Why remove columns of Z and not rows

One could suggest that instead of columns, rows could be removed from the data matrix Z before performing the DMD algorithm on it. Even though it would also allow to generate several sets of estimated eigenvalues $\hat{\mu}_j$, it is not relevant in the case of the method we developed. Firstly, as exposed above, we need some "test" data points to evaluate the quality of the estimated spectrum $\hat{\sigma}_k$. This cannot be done by using rows as data points instead of columns since (2.5) allows to predict a snapshot of the dynamics for a fixed value of time and not the trajectory of the system as a function of the initial condition. And as we do not know a model that could achieve this, it would not be possible to compute the error $E_{\hat{\sigma}_k}$. Secondly, as the number of estimated eigenvalues $\hat{\mu}_j$ is equal to the number of rows of Z, removing rows would decrease the size of $\hat{\sigma}_k$. Since the number of $\hat{\mu}_j$ is typically smaller than the number of exact eigenvalues, decreasing the size of the approximated spectrum would not help to better estimate the eigenvalues of K.

Increase the number of estimated eigenvalues

It is possible to increase the number of estimated eigenvalues $\hat{\mu}_j$ by augmenting the number of rows of the data matrix Z as exposed in subsection 1.6.2 (Using DMD to approximate the Koopman eigenvalues). It should be done before applying algorithm 5.

Chapter 3

Validation

3.1 Validation of the selection criterion

3.1.1 Quality indicator of the estimated spectrum

Quality of the estimated spectrum
There is no intrinsic definition of the quality of a spectrum. In this work, an approximated spectrum, $\hat{\sigma}_k$, is loosely said to be of good quality if it is a good estimate of the spectrum $\sigma(K)$, i.e. the exact eigenvalues of the dynamics matrix K. Furthermore, it is not possible to evaluate the quality of $\hat{\sigma}_k$ computed in the algorithm 5, by comparing it to $\sigma(K)$, since the goal of the algorithm is precisely to estimate $\sigma(K)$. We therefore had to choose another means of evaluation, requiring only the available information, i.e. the snapshots z_s and the values of time at which they were taken, $s\Delta t$.

Relying on the framework of the Koopman operator related to the DMD setting, we defined the quality of the spectrum as its "ability" to make accurate predictions about the dynamics of the network. This criterion makes sense in that the spectrum we are trying to estimate is precisely the eigenvalues of the matrix K, describing the collective dynamics of the network.

Quality indicator
As a reminder, the quality indicator we chose in this work is the error $E_{\hat{\sigma}_k}$ (2.8): the lower its value, the better the estimated spectrum.

3.1.2 Relevance of the choice of the quality indicator

Evaluating the prediction performance of g_k
As $E_{\hat{\sigma}_k}$ is used to determine the best spectrum, which is the goal of algorithm 5, we would like to have some evidence that the choice of this quality indicator is relevant. This can be achieved by investigating if g (see (2.5)) correctly predicts the snapshots of the dynamics. Indeed, if it made absurd or random predictions, $E_{\hat{\sigma}_k}$ would not give any indication about the quality of the spectrum as we defined it above. Note that g has been derived from (1.18) which is exact, but g is only a finite approximation of it so we do not have any guarantee that it is a good prediction of the dynamics, even if there exists a theoretical framework allowing us to assume that it is a relevant choice.

Comparison with a naive prediction
Investigating if g predicts correctly the dynamics can be done by comparing its prediction

performance to the performance of a naive prediction method. If g gives results that are as bad as the ones obtained with the naive method, we know it does not make sense to use it. If g better performs than the naive prediction function, it indicates that our choice is reasonable.

We first apply the algorithm 5 to a data matrix Z and associated times T, and obtain the estimated spectrum $\hat{\sigma}_{k^*}$ and associated modes $\{\hat{V}_{j,k^*}\}$ as values for the parameters of g.
Let

- z_i be a snapshot of the dynamics, i.e. a column of the data matrix Z
- $\hat{z}_i$ be the prediction of z_i obtained by using g
- $\hat{z}_i^n$ be the prediction of z_i using a naive prediction function

We choose the naive prediction function

$$h^n : \mathbb{R}^{rp \times S} \to \mathbb{R}^{rp} : Z \setminus z_i \to \hat{z}_i^n \tag{3.1}$$

such that $\hat{z}_i^n$ is a vector of rp elements which as been generated from a normal distribution with mean and variance equal to the mean and variance of all the elements of the matrix $Z \setminus z_i$, i.e. the matrix Z from which the column z_i has been removed.

Mean of comparison
We define

$$\text{err}_{g_i} = ||z_i - \hat{z}_i||_2$$

as the error of predicting $\hat{z}_i$ using g when the exact snapshot is z_i, and

$$\text{err}_{h_i^n} = ||z_i - \hat{z}_i^n||_2$$

as the error of predicting $\hat{z}_i^n$ using h^n when the exact snapshot is z_i. We did not divide err_{g_i} and $\text{err}_{h_i^n}$ by $||z_i||$, which would have been to some extent equivalent to considering the relative error of the prediction, for the following reason. In the case of linear local dynamics, the system converges to zero, which implies that, if z_i is very close to zero, then both errors would become very large.

In order to compare both errors, we now define the ratio

$$r_p = \frac{\frac{1}{|\mathcal{K}|}\sum_{i \in \mathcal{K}} \text{err}_{h_i^n}}{\frac{1}{|\mathcal{K}|}\sum_{i \in \mathcal{K}} \text{err}_{g_i}} \qquad \text{where } \mathcal{K} = \{i | z_i \in Z_{k^*}\} \tag{3.2}$$

If

$$r_p > 1$$

then function g makes better predictions about the dynamics than the function h^n. We chose to use the ratio of the means of the errors and not the mean of their ratios to avoid very large ratios due to small errors err_{g_i}. We chose

$$\mathcal{K} = \{i | z_i \in Z_{k^*}\} = \{i | z_i \text{ has not been used to estimate } \hat{\sigma}_{k^*} \text{ through the DMD algorithm}\}$$

because the quality indicator $E_{\hat{\sigma}_k}$ precisely used those snapshots to evaluate the quality of $\hat{\sigma}_{k^*}$.

3.1.3 Results

Simulations settings
A matrix $Z \in \mathbb{R}^{20\times 45}$ is first generated from a network system characterized by a directed Erdős-Renyi graph structure and identical local dynamics described by 2 states at each node. The chosen observation function is the one given in (1.10), i.e.

$$f = e^l \quad \Rightarrow \quad f(X(t)) = (e^l)^T X(t) = X_l \quad \in \mathbb{R}$$

where e^l is the l^{th} unit vector, i.e. the vector whose components are all zero except the l^{th} one which is equal to 1. So only one state among all the dynamic states of the network is measured and consequently $p = 1$.
We vary the following parameters:

- n, the number of nodes in the network (note that, $K \in \mathbb{R}^{2n\times 2n}$ as there are 2 states at each node)
- d, the number of snapshots z_s removed from Z at each iteration in algorithm 5. Note that, as $Z \in \mathbb{R}^{20\times 45}$, $d_{\max} = 24$ (see (2.10))
- whether the local dynamics is linear or not

If the local dynamics is linear it is described by (1.4) with

$$A = \begin{bmatrix} -1 & -2 \\ 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

If it is nonlinear, it is given by

$$\begin{bmatrix} \dot{x}_{k,1} \\ \dot{x}_{k,2} \end{bmatrix} = \begin{bmatrix} -x_{k,1} - x_{k,1}^3 - 2x_{k,2} \\ x_{k,1} - x_{k,2} - x_{k,2}^3 \end{bmatrix} + \begin{bmatrix} \cos(x_{k,2}) \\ 1.5\cos(x_{k,2}) \end{bmatrix} u_k$$
$$y_k = x_{k,1} + x_{k,1}^2 + x_{k,2}$$

For each pair of values of (n, d), the means of the errors,

$$\langle \mathrm{err}_g \rangle \triangleq \frac{1}{|\mathcal{K}|} \sum_{i\in\mathcal{K}} \mathrm{err}_{g_i} \qquad \text{and} \qquad \langle \mathrm{err}_{h^n} \rangle \triangleq \frac{1}{|\mathcal{K}|} \sum_{i\in\mathcal{K}} \mathrm{err}_{h_i^n}$$

are computed.

This is done for each pair of values (n, d) over 100 simulations, i.e. for 100 different matrices Z, and the ratio r_p is then calculated afterwards as

$$r_p = \frac{\overline{\mathrm{err}}_{h^n}}{\overline{\mathrm{err}}_g} \triangleq \frac{\frac{1}{100}\sum_{j=1}^{100} \langle \mathrm{err}_{h^n} \rangle_j}{\frac{1}{100}\sum_{j=1}^{100} \langle \mathrm{err}_g \rangle_j}$$

Numerical results

We considered to be estimated spectra $\sigma(K)$ of large and small size. Large (small) size spectra correspond to networks with a high (low) number of nodes. In what follows, we chose $n = 100$ for large networks and $n = 20$ for small ones, made d vary and obtained the following values for $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$. The value of r_p is not displayed as it is obvious that it is always larger than 1 when looking at the results.

Results obtained for a linear local dynamics are displayed in Figure 3.1 and Table 3.1 for $n = 20$, and in Figure 3.2 and Table 3.2 for $n = 100$. Results corresponding to a nonlinear local dynamics are exposed in Figure 3.3 and Table 3.3 for $n = 20$ and in Figure 3.4 and Table 3.4 for $n = 100$.

Linear local dynamics, $n = 20$

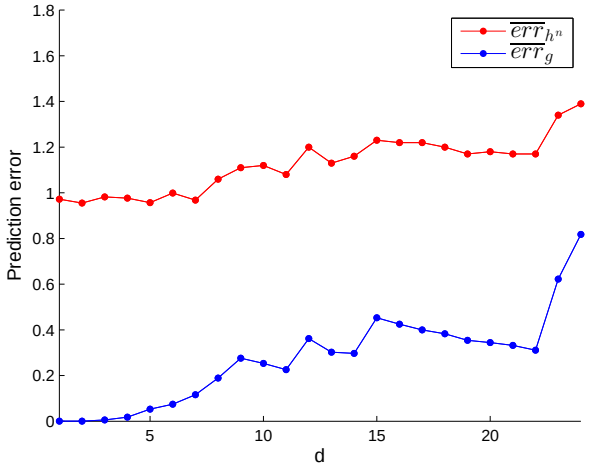


Figure 3.1: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 20$, linear local dynamics and d varying.

d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$		d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$
1	$5.63\ 10^{-8}$	$9.72\ 10^{-1}$		13	$3.02\ 10^{-1}$	1.13
2	$2.36\ 10^{-4}$	$9.55\ 10^{-1}$		14	$2.97\ 10^{-1}$	1.16
3	$5.84\ 10^{-3}$	$9.82\ 10^{-1}$		15	$4.53\ 10^{-1}$	1.23
4	$1.76\ 10^{-2}$	$9.77\ 10^{-1}$		16	$4.25\ 10^{-1}$	1.22
5	$5.27\ 10^{-2}$	$9.57\ 10^{-1}$		17	$4\ 10^{-1}$	1.22
6	$7.43\ 10^{-2}$	$9.99\ 10^{-1}$		18	$3.83\ 10^{-1}$	1.2
7	$1.16\ 10^{-1}$	$9.68\ 10^{-1}$		19	$3.54\ 10^{-1}$	1.17
8	$1.89\ 10^{-1}$	1.06		20	$3.44\ 10^{-1}$	1.18
9	$2.76\ 10^{-1}$	1.11		21	$3.32\ 10^{-1}$	1.17
10	$2.53\ 10^{-1}$	1.12		22	$3.11\ 10^{-1}$	1.17
11	$2.26\ 10^{-1}$	1.08		23	$6.22\ 10^{-1}$	1.34
12	$3.62\ 10^{-1}$	1.2		24	$8.18\ 10^{-1}$	1.39

Table 3.1: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 20$, linear local dynamics, and d varying.

Linear local dynamics, $n = 100$

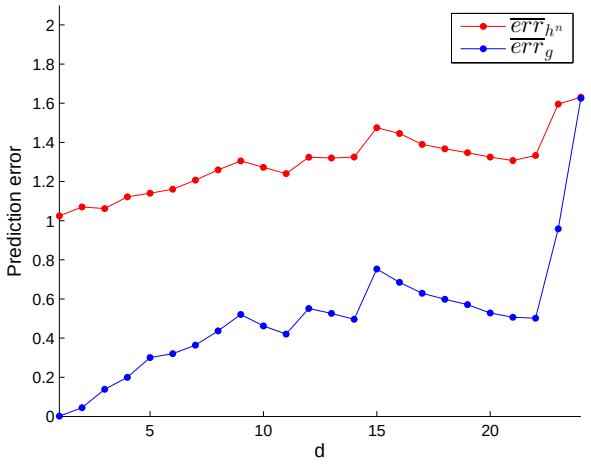


Figure 3.2: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 100$, linear local dynamics and d varying.

d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$		d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$
1	$1.71\ 10^{-3}$	1.02		13	$5.26\ 10^{-1}$	1.32
2	$4.48\ 10^{-2}$	1.07		14	$4.96\ 10^{-1}$	1.32
3	$1.39\ 10^{-1}$	1.06		15	$7.53\ 10^{-1}$	1.48
4	$2\ 10^{-1}$	1.12		16	$6.85\ 10^{-1}$	1.45
5	$3.01\ 10^{-1}$	1.14		17	$6.3\ 10^{-1}$	1.39
6	$3.2\ 10^{-1}$	1.16		18	$5.99\ 10^{-1}$	1.37
7	$3.64\ 10^{-1}$	1.21		19	$5.71\ 10^{-1}$	1.35
8	$4.37\ 10^{-1}$	1.26		20	$5.28\ 10^{-1}$	1.33
9	$5.21\ 10^{-1}$	1.31		21	$5.07\ 10^{-1}$	1.31
10	$4.62\ 10^{-1}$	1.27		22	$5.02\ 10^{-1}$	1.33
11	$4.21\ 10^{-1}$	1.24		23	$9.59\ 10^{-1}$	1.6
12	$5.52\ 10^{-1}$	1.32		24	1.63	1.63

Table 3.2: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 100$, linear local dynamics and d varying.

Nonlinear local dynamics, $n = 20$

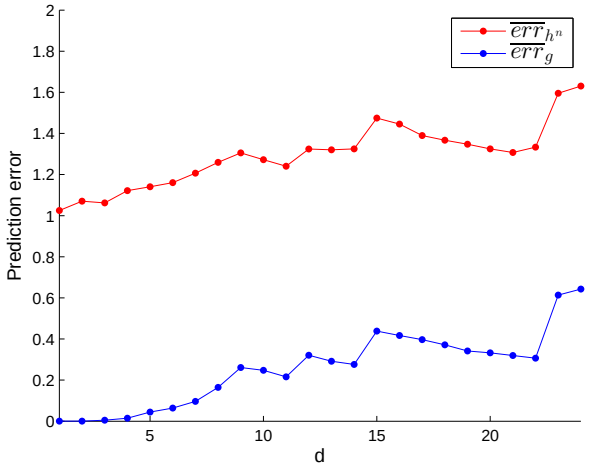


Figure 3.3: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 20$, nonlinear local dynamics and d varying.

d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$		d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$
1	$5.68\ 10^{-8}$	1.02		13	$2.912\ 10^{-1}$	1.32
2	$2.25\ 10^{-4}$	1.07		14	$2.76\ 10^{-1}$	1.32
3	$4.68\ 10^{-3}$	1.06		15	$4.39\ 10^{-1}$	1.48
4	$1.49\ 10^{-2}$	1.12		16	$4.17\ 10^{-1}$	1.45
5	$4.48\ 10^{-2}$	1.14		17	$3.97\ 10^{-1}$	1.39
6	$6.4\ 10^{-2}$	1.16		18	$3.71\ 10^{-1}$	1.37
7	$9.63\ 10^{-1}$	1.21		19	$3.42\ 10^{-1}$	1.35
8	$1.65\ 10^{-1}$	1.26		20	$3.32\ 10^{-1}$	1.33
9	$2.61\ 10^{-1}$	1.31		21	$3.19\ 10^{-1}$	1.31
10	$2.48\ 10^{-1}$	1.27		22	$3.07\ 10^{-1}$	1.33
11	$2.16\ 10^{-1}$	1.24		23	$6.13\ 10^{-1}$	1.6
12	$3.21\ 10^{-1}$	1.32		24	$6.43\ 10^{-1}$	1.63

Table 3.3: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 20$, nonlinear local dynamics, and d varying.

Nonlinear local dynamics, $n = 100$

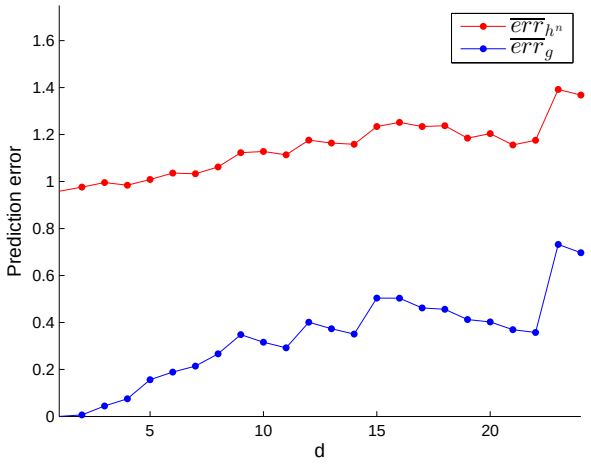


Figure 3.4: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 100$, nonlinear local dynamics and d varying.

d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$		d	$\overline{\text{err}}_g$	$\overline{\text{err}}_{h^n}$
1	$3.93\ 10^{-5}$	0.96		13	$3.74\ 10^{-1}$	1.16
2	$6.57\ 10^{-3}$	0.98		14	$3.51\ 10^{-1}$	1.16
3	$4.46\ 10^{-2}$	0.99		15	$5.04\ 10^{-1}$	1.23
4	$7.51\ 10^{-2}$	0.98		16	$5.04\ 10^{-1}$	1.25
5	$1.57\ 10^{-1}$	1.01		17	$4.62\ 10^{-1}$	1.23
6	$1.89\ 10^{-1}$	1.04		18	$4.56\ 10^{-1}$	1.24
7	$2.14\ 10^{-1}$	1.03		19	$4.13\ 10^{-1}$	1.18
8	$2.67\ 10^{-1}$	1.06		20	$4.03\ 10^{-1}$	1.2
9	$3.48\ 10^{-1}$	1.12		21	$3.7\ 10^{-1}$	1.16
10	$3.16\ 10^{-1}$	1.13		22	$3.57\ 10^{-1}$	1.18
11	$2.92\ 10^{-1}$	1.11		23	$7.32\ 10^{-1}$	1.39
12	$4.01\ 10^{-1}$	1.18		24	$6.97\ 10^{-1}$	1.37

Table 3.4: Values of $\overline{\text{err}}_{h^n}$ and $\overline{\text{err}}_g$ for $n = 100$, nonlinear local dynamics and d varying.

Discussion

First note that the results are qualitatively similar for small and large spectra and for linear and nonlinear dynamics. Secondly, observe that the ratio r_p is smaller than 1 for large and small spectra, for linear and nonlinear local dynamics and for all values of d. This indicates that the function g (2.5) better predicts the network dynamics than the naive prediction function h^n (3.1) and thus that choosing the error $E_{\hat{\sigma}_k}$ as quality indicator of the estimated spectrum is a reasonable choice.

Notice that $\overline{\text{err}}_g$ is very small for low values of d in any case which means that g allows to predict the network dynamics with a high accuracy. $\overline{\text{err}}_g$ grows with the increase of the value of d. This is due to the fact that the higher d, the higher the proportion of information about the dynamics that is not taken into account for the estimation of the Koopman modes and eigenvalues of the dynamics. As the prediction function g depends upon these, it is less accurate if the modes and eigenvalues capture the dynamics less well. The fast growth of $\overline{\text{err}}_g$ for values of d close to $d_{\max}$ (usually for $d = d_{\max}$, $d_{\max} - 1$), that is especially noticeable in Figure 3.2 and Table 3.2, can be explained by the following. If $d = d_{\max}$, the system $Y = PX$ (2.9) bears as many unknowns as equations and admits thus a single solution depending on all the data so it might be ill-conditioned (see subsection 2.4.6 (Some comments)).

The growth of $\overline{\text{err}}_{h^n}$ with the increase of d is, at first sight, surprising as the prediction function h^n is not influenced at all by d (see (3.1)). It might be due to the fact that $\overline{\text{err}}_{h^n}$ depends on d as it is equal to the mean of $\langle \text{err}_{h^n} \rangle$ over several simulations and that $\langle \text{err}_{h^n} \rangle$ is influenced by d. Indeed,

$$\langle \text{err}_{h^n} \rangle \triangleq \frac{1}{|\mathcal{K}|} \sum_{i \in \mathcal{K}} \text{err}_{h_i^n} \tag{3.3}$$

and $|\mathcal{K}| = d$. As d increases, the number of terms in the sum in (3.3) grows and, consequently,

$\langle \text{err}_{h^n} \rangle$ would be larger for a higher value of d, although this effect is to some extent balanced by dividing the sum by d. This might also influence the increase of $\overline{\text{err}}_g$ but it is not the only factor, as exposed above.

3.2 Does cross-validation improve the DMD algorithm?

We will now study whether using cross-validation as is done in algorithm 5 allows to improve the estimation of the eigenvalues of the dynamics matrix K through the DMD algorithm or not.

To this end, we compare the estimated spectra computed with and without cross-validation, both obtained from the same data matrix Z. We denote them respectively by

$$\hat{\sigma}_c(K) \qquad \text{and} \qquad \hat{\sigma}_{nc}(K)$$

where c stands for *cross-validation* and nc for *no cross-validation*.

3.2.1 Comparing the estimated spectra

No known mean of comparison
Now, our goal is to determine which of both estimated spectra $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$ is the best approximation of the exact set of eigenvalues $\sigma(K)$. This will be done by comparing $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$ to $\sigma(K)$. However, there exists no commonly accepted measure of similarity or distance between two spectra. Some have been proposed, such as the *spectral distance* in [40] but it requires that both spectra contain the same number of eigenvalues. This is however not the case here given that

$$|\hat{\sigma}_c(K)| = |\hat{\sigma}_{nc}(K)| = rp$$

that is the number of rows of the data matrix Z which is typically smaller than

$$|\sigma(K)| = nm$$

i.e. the total number of states in the network system composed of n dynamical systems with each m states.

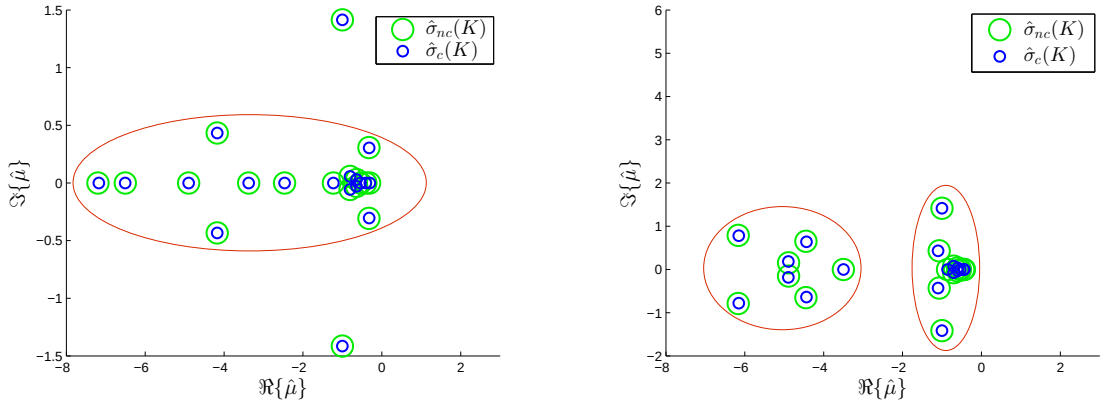
Using the spectral moments
We therefore had to come up with our own mean of comparison. We chose to use the spectral moments and to define the best spectrum among $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$ as the one enabling to estimate the best the spectral moments associated to $\sigma(K)$. This choice was made for two reasons. Firstly, it allows to compare spectra of different sizes. Secondly, the first two spectral moments are the quantities we aim at measuring in the case of spectral identification for large graphs so the accuracy of their estimation is of particular interest.

The i^{th} exact spectral moment of the dynamics matrix K is given by (1.19). $\hat{\mathcal{M}}_c^{(i)}$ and $\hat{\mathcal{M}}_{nc}^{(i)}$ are respectively the i^{th} spectral moments associated to $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$. They are computed as follows

$$\hat{\mathcal{M}}_c^{(i)} = \frac{1}{|\hat{\sigma}_c(K)|} \sum_{\hat{\mu}_{c_j} \in \hat{\sigma}_c(K)} \hat{\mu}_{c_j}^i \tag{3.4}$$

$$\hat{\mathcal{M}}_{nc}^{(i)} = \frac{1}{|\hat{\sigma}_{nc}(K)|} \sum_{\hat{\mu}_{nc_j} \in \hat{\sigma}_{nc}(K)} \hat{\mu}_{nc_j}^i \tag{3.5}$$

We first thought about using (3.4) and (3.5) only when the number of eigenvalues to be estimated, i.e. nm, was small and using the same heuristics as for spectral identification of large graphs (see subsection 1.5.1 (Dynamical systems with identical units)) in the case of high values of nm. However, we have observed through simulations that the number of clusters of eigenvalues is not always equal to the number of states at each unit, as is assumed in this heuristics. This is illustrated in Figure 3.5.



(a) One single cluster

(b) Two clusters

Figure 3.5: Issue related to using the clusters of estimated eigenvalues to approximate the spectral moments. The simulations parameters have the same values for both simulations ($n = 80$, $Z \in \mathbb{R}^{20\times 45}$, $d = 1$ and linear local dynamics). Although each unit is described by 2 dynamical states, estimated eigenvalues in (a) form only one cluster, while two clusters are observed in (b) (after removing the outliers).

The clusters are determined using k-means clustering. This method requires to fix the number of clusters before applying it to the data but it is not possible to determine the right number of clusters, based on $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$, in an automatic way. Always setting the number of clusters equal to the number of states at each unit would thus induce, in the estimation of the spectral moments, errors that are not due to the use of cross-validation and this would to some extent bias the results. The second reason for which we chose not to use this heuristic method is because it is characterized by some randomness. This partially obscures the effects of using cross-validation since it introduces randomness in the computation of the prediction error of the spectral moments.
Approximating the first two spectral moments using (3.4) and (3.5) regardless of the network size implies that their estimate might be less precise in some instances for large graphs. Yet, this drawback seems to have a less important impact than the ones exposed just above when evaluating whether using cross-validation effectively enhances the approximation of the dynamics eigenvalues or not.

Mean of comparison
In order to evaluate how accurately each estimated spectrum allows to approximate the spectral moments of the matrix K, we define

$$\text{err}_c^{(i)} = \left| \frac{\mathcal{M}^{(i)} - \hat{\mathcal{M}}_c^{(i)}}{\mathcal{M}^{(i)}} \right| \tag{3.6}$$

and

$$\text{err}_{nc}^{(i)} = \left| \frac{\mathcal{M}^{(i)} - \hat{\mathcal{M}}_{nc}^{(i)}}{\mathcal{M}^{(i)}} \right| \tag{3.7}$$

which are the absolute values of the relative prediction errors of the spectral moments of K.

Adopting the same approach as in subsection 3.1.2 (Relevance of the choice of the quality indicator), we define

$$r_{\mathcal{M}^{(i)}} = \frac{\text{err}_{nc}^{(i)}}{\text{err}_{c}^{(i)}}$$

Then, if

$$r_{\mathcal{M}^{(i)}} > 1$$

algorithm 5 (i.e. using cross-validation) provides better results than simply performing the DMD algorithm on the data matrix Z. We chose to study only the first two spectral moments as the other ones can hardly be obtained.

Simulations settings

The simulations settings are the same as in subsection 3.1.3 (Results). For each pair of values of (n, d), the errors $\text{err}_{c}^{(i)}$ and $\text{err}_{nc}^{(i)}$ $(i = 1, 2)$ are computed over 100 simulations, i.e. for 100 different matrices Z. The ratios $r_{\mathcal{M}^{(i)}}$ are then calculated afterwards as

$$r_{\mathcal{M}^{(i)}} = \frac{\langle \text{err}_{nc}^{(i)} \rangle}{\langle \text{err}_{c}^{(i)} \rangle} \triangleq \frac{\frac{1}{100} \sum_{j=1}^{100} \text{err}_{nc_j}^{(i)}}{\frac{1}{100} \sum_{j=1}^{100} \text{err}_{c_j}^{(i)}}$$

We chose to use the ratio of the means instead of the mean of the ratios for the same reason as above, i.e. to avoid large ratios due to very small errors $\text{err}_{c}^{(i)}$.

3.2.2 Results for linear local dynamics

n=10

We first considered spectra to be estimated of small size, i.e. $n = 10$. In this way, the number of estimated eigenvalues $|\hat{\sigma}_c(K)| = |\hat{\sigma}_{nc}(K)| = rp = 20$ (since $Z \in \mathbb{R}^{20\times 45}$) is equal to the number of exact eigenvalues $|\sigma(K)| = nm = 20$. This choice was made to better illustrate visually some of the results discussed thereafter. The values obtained for $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.6 and Figure 3.7 displays the values of $\langle \text{err}_{c}^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$.

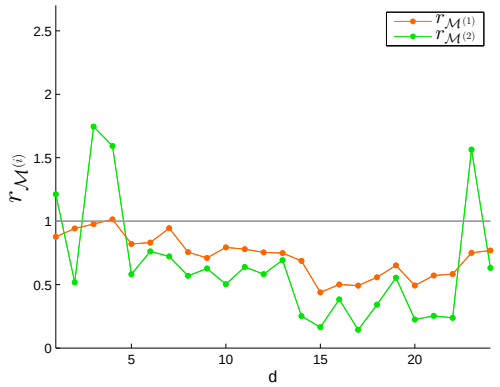
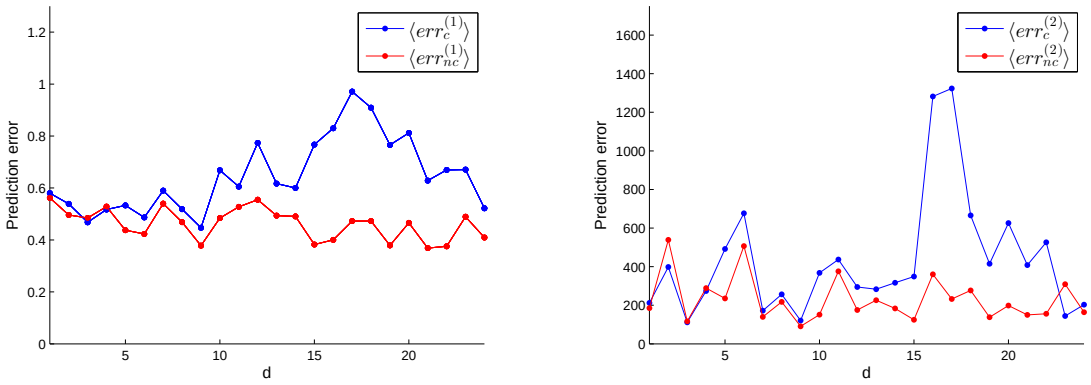


Figure 3.6: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 10$, d varying and a local linear dynamics



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.7: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 10$, d varying and a local linear dynamics

No performance enhancement When looking at Figure 3.6, it seems that using cross-validation does not allow to enhance the performance of the DMD algorithm with respect to the estimation of $\sigma(K)$. For $d \leq 5$, $r_{\mathcal{M}^{(1)}}$ is close to 1 but for larger values, it is clearly lower. $r_{\mathcal{M}^{(2)}}$ "oscillates" around 1 for $d \leq 5$ but is also smaller for larger values of d. Its high value for $d = 23$ is probably due to the fact that the DMD algorithm is not always fully reliable because it has not been observed in other simulations with the same parameter values. The same observations can also be made when analyzing the evolution of the mean prediction errors in Figure 3.7. In both cases, $\langle \text{err}_c^{(i)} \rangle$ oscillates around $\langle \text{err}_{nc}^{(i)} \rangle$ for $d \leq 5$ but is clearly larger for $d \geq 5$.

Consequently, for small values of d, using cross-validation sometimes slightly enhances the estimation of the dynamics eigenvalues but not always. This is due to the fact that, in case of linear dynamics, the DMD algorithm performance is already good, as is shown in [2] and is illustrated in Figure 3.8. In some instances, the estimated spectrum obtained with cross-validation is a little better than the one computed without resorting on it. In other cases, $\hat{\sigma}_c(K)$ is worse than $\hat{\sigma}_c(K)$, even if it is not a poor estimate, like in Figure 3.8.

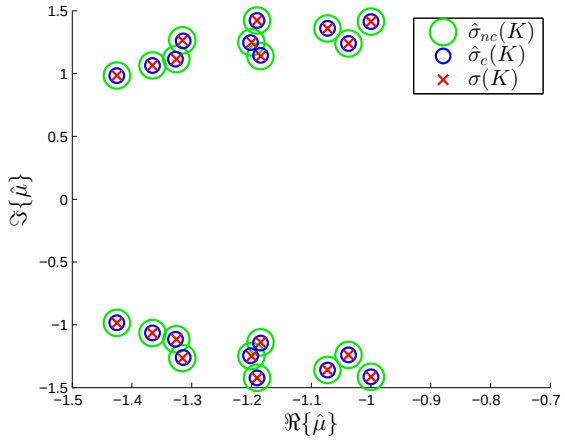


Figure 3.8: $d = 2$
$r_{\mathcal{M}^{(1)}} = 0.54$ and $r_{\mathcal{M}^{(2)}} = 0.36$

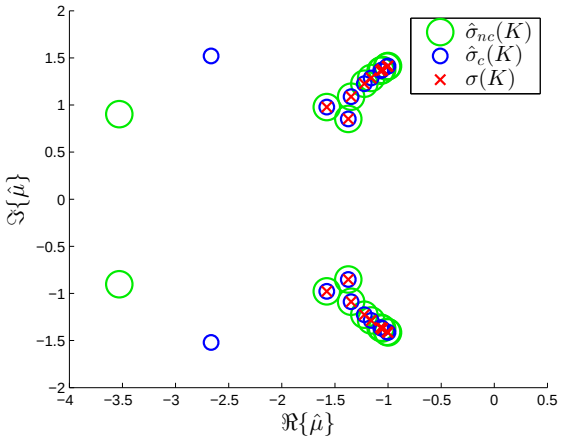


Figure 3.9: $d = 2$
$r_{\mathcal{M}^{(1)}} = 1.54$ and $r_{\mathcal{M}^{(2)}} = 2.23$

It seems, from simulations, that for a same $\sigma(K)$, there is a low variability between the spectra estimated with and without cross-validation. Indeed, we have observed that each time $\hat{\sigma}_{nc}(K)$ contains outliers, so does $\hat{\sigma}_c(K)$, as illustrated in Figure 3.9. This implies that algorithm 5 has to "choose" between spectra $\hat{\sigma}_k$ that are all bad ones. However, using cross-validation sometimes allows to select a spectrum containing "better" outliers, such as in Figure 3.9.

Low variability between the estimated spectra and the good performance of the DMD algorithm in case of linear dynamics seem both to account for ratios values close to 1 for small values of d. For larger values, using cross-validation clearly does not improve the DMD algorithm performance. In addition to the two above mentioned factors, it is due to the already explained fact that the more snapshots of the dynamics are available to obtain its spectral properties, the better these are estimated. No snapshot is discarded to obtain $\hat{\sigma}_{nc}(K)$ regardless of d, while the number of snapshots removed to compute $\hat{\sigma}_c(K)$ is precisely equal to d.

Large values of $\langle \mathbf{err}_c^{(2)} \rangle$ and $\langle \mathbf{err}_{nc}^{(2)} \rangle$ Let us now take a closer look at the abnormally high values of $\langle \mathrm{err}_c^{(2)} \rangle$ and $\langle \mathrm{err}_{nc}^{(2)} \rangle$ in Figure 3.7. High values for $\mathrm{err}_c^{(2)}$ and $\mathrm{err}_{nc}^{(2)}$ seem to be correlated with the presence of outliers in the estimated spectra and might be explained by the following. As a reminder, the first two estimated spectral moments can be written as (see subsection 1.5.1 (Dynamical systems with identical units) and more precisely (1.22) and (1.23))

$$\hat{\mathcal{M}}^{(1)}(K) = \frac{1}{rp}\left(\sum_{j=1}^{rp} \Re\{\hat{\mu}_j\} + i \underbrace{\sum_{j=1}^{rp} \Im\{\hat{\mu}_j\}}_{0}\right) \tag{3.8}$$

and

$$\hat{\mathcal{M}}^{(2)}(K) = \frac{1}{rp}\left(\sum_{j=1}^{rp} (\Re\{\hat{\mu}_j\})^2 + \sum_{j=1}^{rp} (i\Im\{\hat{\mu}_j\})^2 + \underbrace{\sum_{j=1}^{rp} 2i\Re\{\hat{\mu}_j\}\Im\{\hat{\mu}_j\}}_{0}\right) \tag{3.9}$$

The second and third term in respectively (3.8) and (3.9) are equal to zero only if the eigenvalues are either purely real or have a complex conjugate. First note that outliers in $\hat{\sigma}_c(K)$ and $\hat{\sigma}_{nc}(K)$ induce larger errors $\mathrm{err}_c^{(i)}$ and $\mathrm{err}_{nc}^{(i)}$ $(i = 1, 2)$, i.e. larger prediction error on the spectral moments. The prediction errors on the second spectral moment are larger than for the first one since the real and imaginary parts of the eigenvalues are squared in (3.9) and not in (3.8), and that their imaginary part does not impact $\hat{\mathcal{M}}^{(1)}$. This is illustrated in Figure 3.10.

The complex eigenvalues in $\sigma(K)$ always have a complex conjugate but it may happen that estimated spectra contain complex eigenvalues that do not. If so, the above mentioned terms are not equal to zero, which might induce a higher prediction error that is even larger if those eigenvalues are outliers. In that case, the error on the second spectral moment is again larger than the one on the first moment. Indeed, the non-zero term in (3.9) is twice the product of the imaginary and the real part of those eigenvalue while it is only their imaginary part in (3.8). In addition, it has been observed that the exact value of the second moment is often very small ($\mathcal{M}^{(2)} \sim 10^{-2}$ or 10^{-3}), which makes the prediction error even larger (see Figure 3.10). Note that, if some complex eigenvalues do not have a complex conjugate, the first two estimated spectral moments are not real and, as such, not good approximations of the exact ones. As we

decided to take the absolute value of the relative errors on those (see (3.6) and (3.7)), it is not apparent but one should be aware of it.

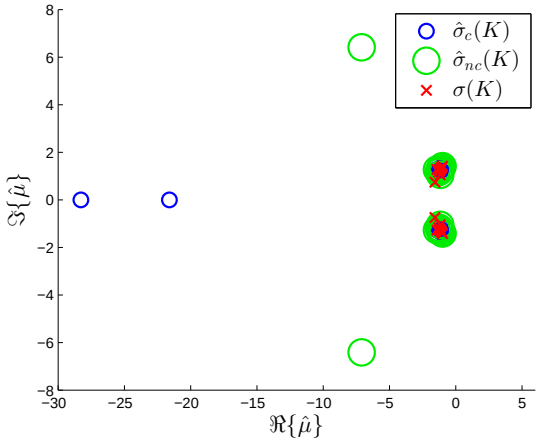


Figure 3.10: $d = 20$

$\text{err}_{nc}^{(2)} = 46.66$ and $\text{err}_{c}^{(2)} = 3.95\ 10^3$

$\text{err}_{nc}^{(1)} = 0.47$ and $\text{err}_{c}^{(1)} = 1.93$

$\mathcal{M}^{(1)} = -1.21$ and $\mathcal{M}^{(2)} = -0.01$

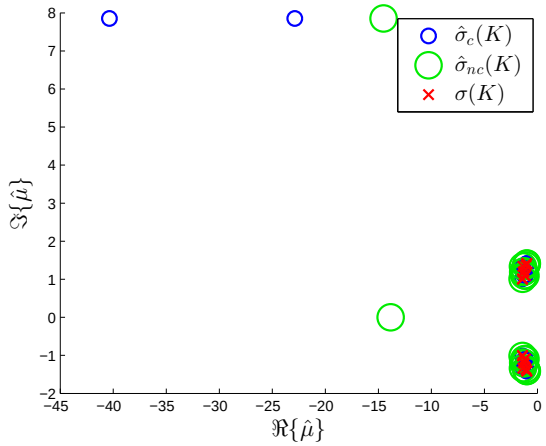


Figure 3.11: $d = 15$

$\text{err}_{nc}^{(2)} = 793$ and $\text{err}_{c}^{(2)} = 3.18\ 10^3$

$\text{err}_{nc}^{(1)} = 1.09$ and $\text{err}_{c}^{(1)} = 2.55$

All this could explain the behavior of $r_{\mathcal{M}^{(2)}}$ that might look slightly erratic, and its high variance. $\text{err}_{c}^{(2)}$ and $\text{err}_{nc}^{(2)}$ are more sensitive to outliers and, as such, depending on whether some are present in the estimated spectra or not and on their value, both errors can take very different values from simulation to simulation. As $\langle \text{err}_{c}^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$ are computed as the arithmetic mean of respectively $\text{err}_{c}^{(2)}$ and $\text{err}_{nc}^{(2)}$ over 100 simulations, a single simulation with large outliers is enough to have a very high value of $\langle \text{err}_{c}^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$. Even though, it makes sense to make statistical tests by computing means since the outcome of the DMD algorithm varies, sometimes widely, from one simulation to another.

n=100

We then studied spectra to be estimated of large size, i.e. $n = 100$. The values obtained for $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.12 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.13.

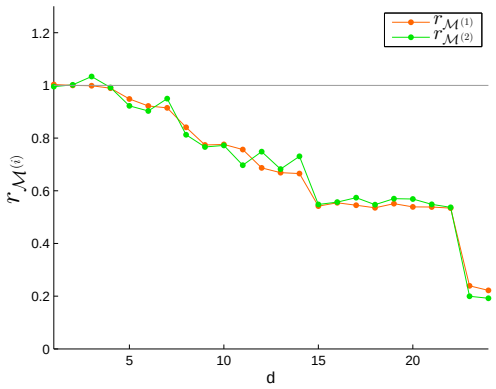
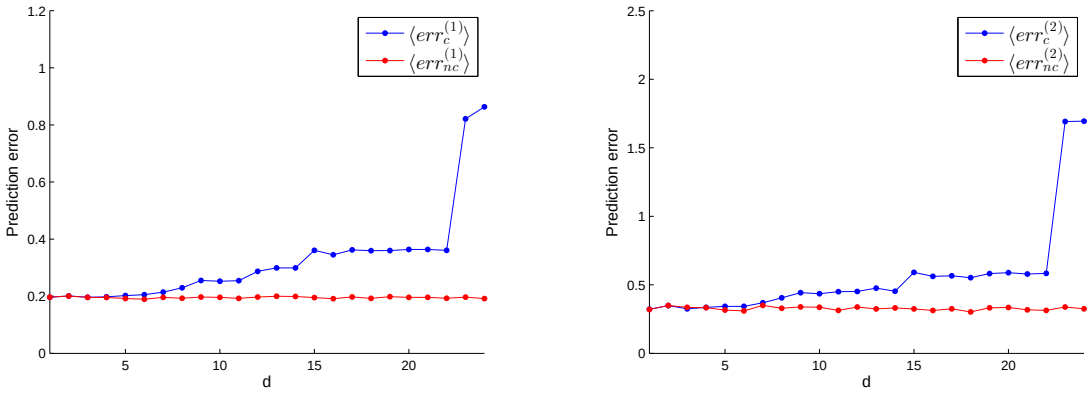


Figure 3.12: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 100$, d varying and a local linear dynamics



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$

(b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.13: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 100$, d varying and a local linear dynamics

No performance enhancement Regarding $r_{\mathcal{M}^{(i)}}$, the same trends as for $n = 10$ are observed in Figure 3.12 but they are clearer for $n = 100$. Both ratios are very close to 1 for $d \leq 4$. They oscillate around 1 for the same reason as exposed above: the DMD algorithm is already high-performance in case of linear dynamics so using cross-validation gives results very similar to the ones obtained without making use of it: sometimes $\hat{\sigma}_c(K)$ is a little better than $\hat{\sigma}_{nc}(K)$ and in other instances, of a slightly lower quality. Furthermore, the same low variability between the spectra estimated without and with cross-validations is observed through simulations (see Figure 3.14 for example). $r_{\mathcal{M}^{(i)}}$ then decrease with d growing given that the more snapshots are removed to compute $\hat{\sigma}_c(K)$, the less information about the dynamics is available and the lower the DMD algorithm performance is. This can also be deduced by looking at the evolution of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ in Figure 3.13: both errors have values very close to each other for $d \leq 4$ and, for $d \geq 4$, $\langle \text{err}_{nc}^{(i)} \rangle$ remains almost constant, while $\langle \text{err}_c^{(i)} \rangle$ increases. The sharp decrease of $r_{\mathcal{M}^{(i)}}$ for d close to $d_{\max}$ is due to the fact that, when $d = d_{\max}$, the system $Y = PX$ solved by

the DMD algorithm bears as many unknowns as equations and admits consequently one single solution, depending on all the data, so it might be ill-conditioned.

Lower values of $\langle \mathbf{err}_c^{(i)} \rangle$ and $\langle \mathbf{err}_{nc}^{(i)} \rangle$ Another significant difference with the case $n = 10$ is that $\langle \mathrm{err}_c^{(2)} \rangle$ and $\langle \mathrm{err}_{nc}^{(2)} \rangle$ take values that are substantially smaller. This could be explained by two observations we made by looking at simulations outcome. The first one is that outliers in the estimated spectra seem to occur less often than for $n = 10$ and that, when they do, there are less of them. Furthermore, the outliers have a smaller real part for $n = 100$ (it is usually larger than -15) than for $n = 10$ (it is sometimes -40 as in Figure 3.10). Their imaginary part however have approximately the same value, i.e. larger than -8. And this can be seen even for large values of d, such as in Figure 3.15. The second observation that might account for the low prediction errors with respect to $\mathcal{M}^{(2)}$ is that its true value is much larger than for $n = 10$. Usually, $\mathcal{M}^{(2)} \sim 10$, as in Figures 3.14 and 3.15. It might be explained by the fact that in this simulation setting, the imaginary part of the exact eigenvalues have approximately the same value as for $n = 10$ but that their real part is larger (compare Figures 3.8 and 3.14 for instance). This has a bigger impact on $\mathcal{M}^{(2)}$ because the eigenvalues real parts are squared in (3.9) and not in (3.8).

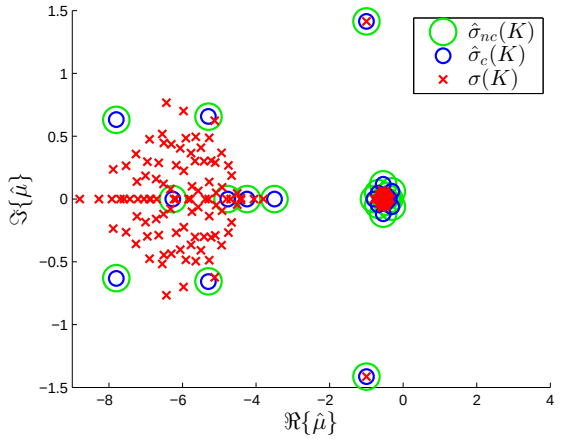


Figure 3.14: $d = 2$

$\mathrm{err}_{nc}^{(2)} = 0.28$ and $\mathrm{err}_c^{(2)} = 0.28$
$\mathrm{err}_{nc}^{(1)} = 0.2$ and $\mathrm{err}_c^{(1)} = 0.2$
$\mathcal{M}^{(1)} = -3.25$ and $\mathcal{M}^{(2)} = 18.6$

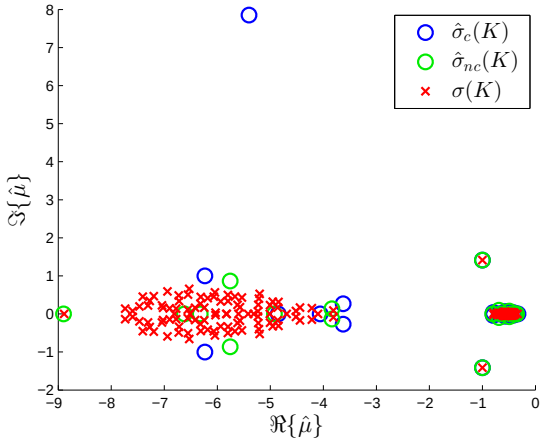


Figure 3.15: $d = 22$

$\mathrm{err}_{nc}^{(2)} = 0.24$ and $\mathrm{err}_c^{(2)} = 0.74$
$\mathrm{err}_{nc}^{(1)} = 0.18$ and $\mathrm{err}_c^{(1)} = 0.37$
$\mathcal{M}^{(1)} = -3.25$ and $\mathcal{M}^{(2)} = 18.61$

We would expect $\langle \mathrm{err}_c^{(1)} \rangle$ and $\langle \mathrm{err}_{nc}^{(1)} \rangle$ to be higher for $n = 100$ than for $n = 10$. Indeed, in the latter case, the number of estimated eigenvalues is equal to the number of exact ones while, for $n = 100$, 20 estimated eigenvalues are used to approximate a spectrum consisting of 200 eigenvalues. We could reasonably assume that $\hat{\sigma}_{nc}(K)$ and $\hat{\sigma}_c(K)$ would therefore estimate less well the first two spectral moments related to $\sigma(K)$ since those are computed as the weighted sum of the (squared) eigenvalues. But it is not what is observed when comparing Figures 3.7a and 3.7a. This is probably because outliers apparently occur less often for $n = 100$ than for $n = 10$ and that their real part is smaller.

Summary
In conclusion, we do not recommend using our method, summarized in algorithm 5, for the purpose of improving the estimation of the dynamics eigenvalues through the DMD algorithm, in instances in which the local dynamics is linear. Using cross-validation for large values of d clearly worsens the approximation of $\sigma(K)$. Regarding small values of d, algorithm 5 does not systematically improve its estimation and, when it does, it only barely enhances the results.

3.2.3 Results for nonlinear local dynamics

Using cross-validation in case of linear dynamics does not allow to improve the estimation of the dynamics spectrum, partly because of the high performance of DMD algorithm and of the low variability between the estimated spectra obtained through it. Considering nonlinear local dynamics might bear better results. Indeed, it is shown in [2] that the DMD algorithm performance is lower when dealing with nonlinear dynamics and it may also be that the variability between the estimated spectra is higher in this instance.

n=10
For $n = 10$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.16 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.17.

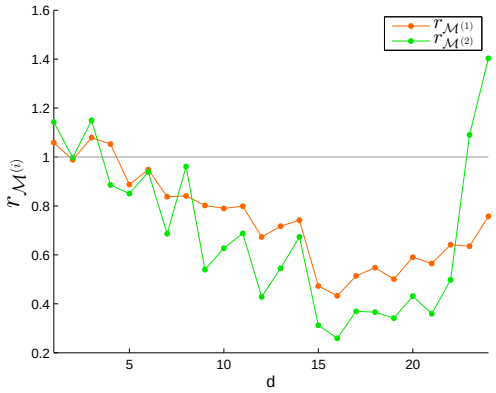
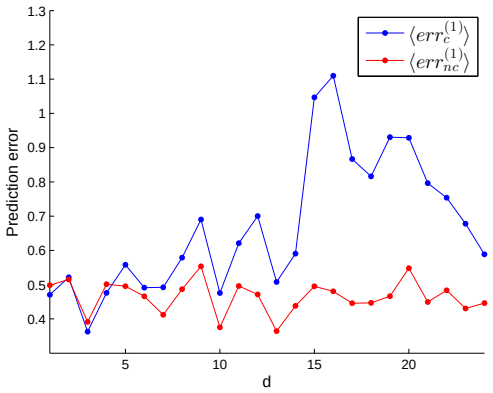
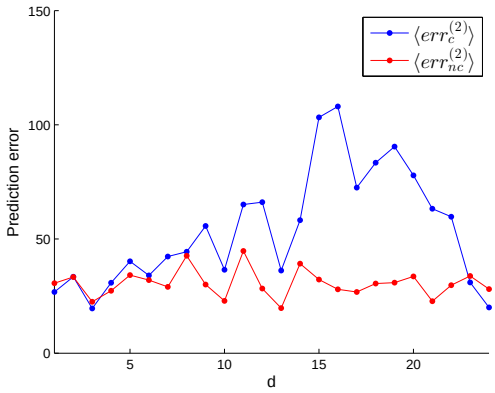


Figure 3.16: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 10$, d varying and a nonlocal linear dynamics



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$



(b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.17: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 10$, d varying and a nonlocal linear dynamics

Slight enhancement for small d The evolution of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ in Figure 3.16 is similar to their behavior in case of linear local dynamics in Figure 3.6. They, though, do not oscillate around 1 for small values of d but are slightly larger. This might be explained by two factors, the first one being the lower accuracy of the DMD algorithm outcome in case of nonlinear dynamics. This can be observed in Figure 3.18, where $\hat{\sigma}_c(K)$ better estimates two exact eigenvalues (indicated by the arrows) than $\hat{\sigma}_{nc}(K)$. Secondly, it seems from simulations that the variability between the spectra estimated with and without cross-validation is slightly higher than for linear dynamics. In Figure 3.19, $\hat{\sigma}_{nc}(K)$ contains two outliers (indicated by arrows) that are not present in $\hat{\sigma}_c(K)$. This specific instance does not happen very often and the outliers in question are not so large but still, it does not occur in case of linear dynamics. Using cross-validation with small d ($d \leq 3$ here) might thus slightly enhance the DMD algorithm performance with respect to the estimation of $\sigma(K)$. However, it is not valid for larger values of d: the negative impact of removing columns seems to take over the positive influence of using cross-validation.

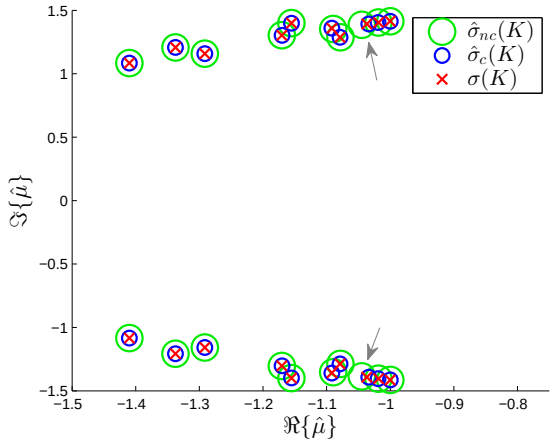


Figure 3.18: $d = 1$

$r_{\mathcal{M}^{(1)}} = 2.14$ and $r_{\mathcal{M}^{(2)}} = 6.78$

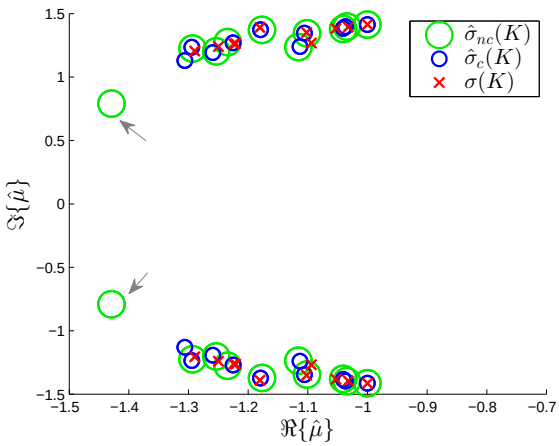


Figure 3.19: $d = 1$

$r_{\mathcal{M}^{(1)}} = 2.04$ and $r_{\mathcal{M}^{(2)}} = 2.31$

The increase of $r_{\mathcal{M}^{(i)}}$ for large values of d and the fact that $r_{\mathcal{M}^{(2)}} > 1$ for $d = 23$ and 24 is surprising but is observed for most of the simulations. It seems to be related to simulations where both $\hat{\sigma}_{nc}(K)$ and $\hat{\sigma}_c(K)$ contain outliers but where those in $\hat{\sigma}_{nc}(K)$ are larger.

Values of $\langle \mathbf{err}_c^{(2)} \rangle$ and $\langle \mathbf{err}_{nc}^{(2)} \rangle$ The high values obtained for $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$ can be explained by the same factors as in the case of linear local dynamics and $n = 10$. The fact that they take smaller values than in Figure 3.7b could be because the outliers observed through simulations seem to have a smaller real part (larger than -15) than for linear dynamics (around -40 in some instances), while their imaginary part is approximately the same in both cases.

n=100

For $n = 100$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.20 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.21.

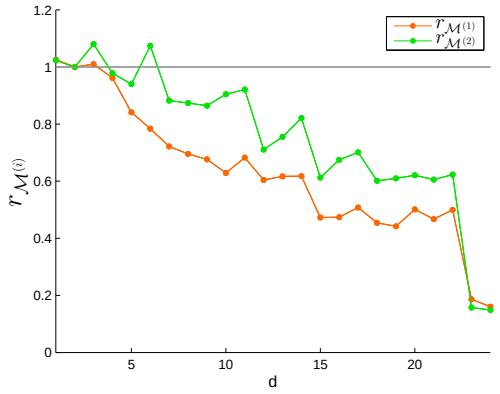
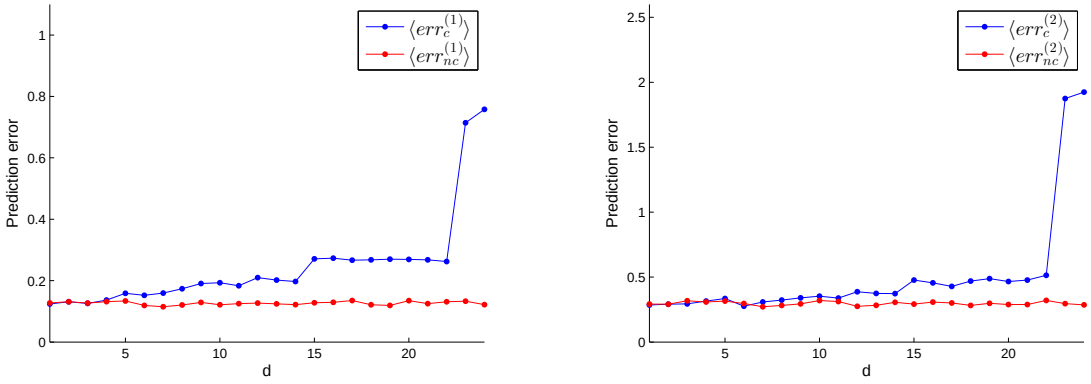


Figure 3.20: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 100$, d varying and a nonlocal linear dynamics



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.21: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 100$, d varying and a nonlocal linear dynamics

Those results are very similar to the ones obtained in case of linear local dynamics. However, as for $n = 10$ in case of nonlinear dynamics, $r_{\mathcal{M}^{(i)}}$ is larger than 1 for $d \leq 3$. Note that the values of $r_{\mathcal{M}^{(i)}}$ for $d \leq 3$ is smaller than for $n = 10$ (see Figure 3.16). So, it seems that using cross-validation with small d might slightly enhance the DMD algorithm performance with

respect to the estimation of the eigenvalues of the dynamics of large networks but that it is not the case anymore for larger values of d and the performance is lower than for spectra to be estimated of small size.

The behavior of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ is very similar to the case of linear dynamics (see Figure 3.13) and the same analysis applies.

Summary
In case of network systems described by a nonlinear local dynamics, it seems that using algorithm 5 enhances a little the estimation of $\sigma(K)$ through the DMD algorithm but only when very few columns are removed at each iteration. This might stem from the fact that there seems to be a higher variability between the estimated spectra and that the DMD algorithm has a lower performance when dealing with nonlinear dynamics. However, as soon as d increases, the negative impact of removing a too high number of snapshots takes over.

3.3 Removing rows in addition to columns for cross-validation

3.3.1 Motivation

Increasing the variability between the spectra
Even if using cross-validation as is done in algorithm 5 enhances the DMD performance in case of nonlinear dynamics, the estimation of $\sigma(K)$ is only slightly improved, as can be deduced from the values of $r_{\mathcal{M}^{(i)}}$ that are very close to 1 (and this only for small values of d). The two main factors accounting for this small enhancement seem to be the already good performance of the DMD algorithm and the low variability between the estimated spectra obtained with and without cross-validation. Lowering the performance of the DMD algorithm does not make any sense but we could try to increase the variability between the spectra.

For this purpose, we decided to remove rows in addition to columns from the data matrix Z at each iteration of algorithm 5. Indeed, since the number of estimated eigenvalues is equal to the number of rows of Z, removing some of them might allow to obtain more different spectra. We cannot, however, discard too many rows as decreasing too much the size of $\hat{\sigma}_c(K)$ would surely worsen the estimation of the spectral properties of the dynamics since less eigenvalues would be estimated. But, if the number of removed rows is such that there are still enough estimated eigenvalues and that the variability between the estimated spectra is high enough, it might be possible to improve the DMD algorithm performance by using cross-validation.

Implementation
At each iteration of the *for* loop of algorithm 5, steps $1-3$ are performed on the the matrices Z and $\bar{Z}$, the matrix Z form which q rows have been discarded. The q removed rows are randomly chosen and vary from one iteration to another. The best estimated spectra is still the one associated with the lowest prediction error $E_{\hat{\sigma}_k}$, regardless of its size.

We chose to set $q = 2$ because the data matrices Z we use for the simulations only have 20 rows but more rows could be removed when a higher number of observations is available. The simulation settings are the same as in section 3.2.1 (Simulations settings).

3.3.2 Results for linear local dynamics

n=10

For $n = 10$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.22 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.23.

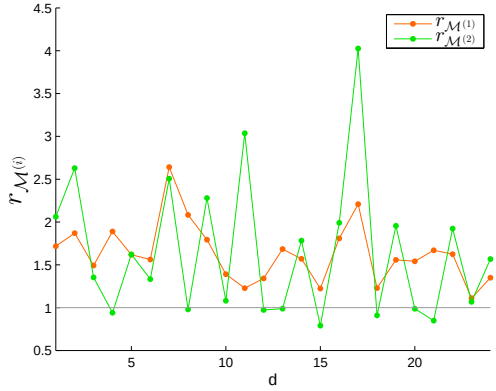
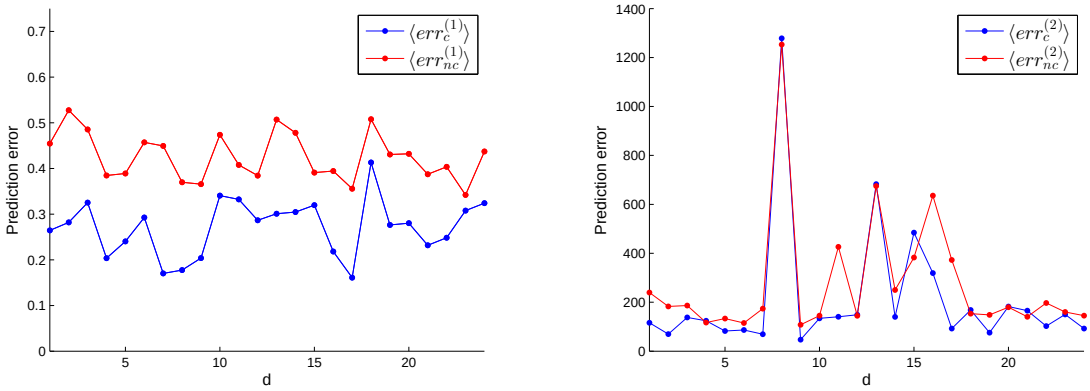


Figure 3.22: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 10$, d varying, a linear local dynamics, $q = 2$



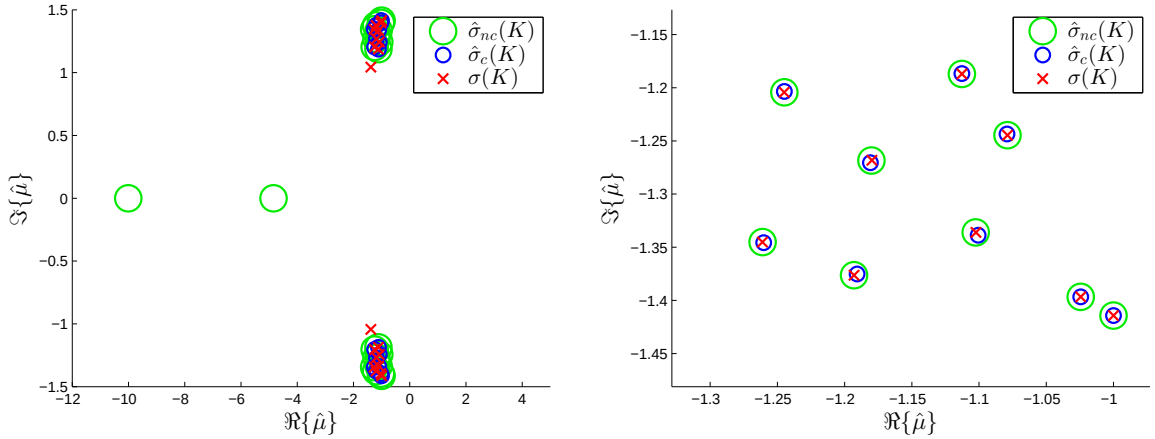
(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.23: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 10$, d varying and a linear local dynamics, $q = 2$

Significant improvement Removing rows apparently enables for algorithm 5 to choose between more different spectra and sometimes to remove outliers among the estimated eigenvalues. Indeed, because of the low variability between estimated spectra of equal size, when $\hat{\sigma}_{nc}(K)$ contains outliers, so does $\hat{\sigma}_c$. But, allowing for an estimated spectrum to be of smaller size (18 here as $q = 2$) sometimes enables to select a spectrum $\hat{\sigma}_c(K)$ in which outliers are not present even if there are some in $\hat{\sigma}_{nc}(K)$, such as in Figure 3.24.

The high values of $r_{\mathcal{M}^{(i)}}$ might seem surprising at first sight but could be explained by the following. As was exposed above, in case $n = 10$, the estimated spectra can contain outliers whose real part can be very large, -45 in some cases. If using cross-validation and discarding rows allows to remove those, then $\hat{\sigma}_c(K)$ is a much better estimation of $\sigma(K)$ than $\hat{\sigma}_{nc}(K)$, which explains the high values for $r_{\mathcal{M}^{(i)}}$. This is observed even for large values of d as is illustrated

on Figure 3.24. In this case, it seems that the positive impact of removing outliers from the estimated spectrum (see Figure 3.24a) takes over the negative influence of loosing information about the network dynamics (see Figure 3.24b).



(a) Entire spectra

(b) Zoom on one of the clusters of exact eigenvalues

Figure 3.24: $d = 20$, $r_{\mathcal{M}^{(1)}} = 23.87$, $\mathcal{M}^{(2)} = 48.24$, $|\hat{\sigma}_c(K)| = 18$

The number of outliers that can be removed depends on the number of rows that are discarded. In this specific simulations setting, no more than two outliers can be removed since $q = 2$. This is illustrated in Figure 3.25, where $\hat{\sigma}_{nc}(K)$ contains 4 outliers so 2 outliers are present in $\hat{\sigma}_c(K)$. If q was set to a higher values, those two outliers would probably disappear too.
The question then arises as why to still continue consider $\hat{\sigma}_c(K)$ obtained from Z, i.e. with no removed rows and why not to systematically choose $\hat{\sigma}_c(K)$ obtained by removing q rows, i.e. from matrix $\bar{Z}$. It is indeed observed that, in most cases, $|\hat{\sigma}_c(K)| = 18$ but not always. When $\hat{\sigma}_{nc}(K)$ does not contain any outliers, $|\hat{\sigma}_c(K)|$ is equal to 20 or 18, depending on which one gives the lowest approximation error $E_{\hat{\sigma}_k}$ (see Figure 3.26 in which $|\hat{\sigma}_c(K)| = 20$).

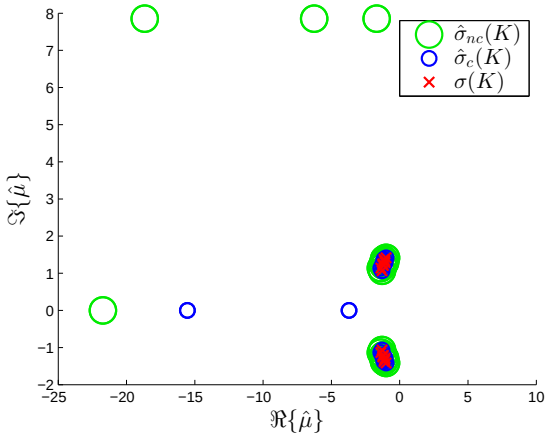


Figure 3.25: $d = 2$, $r_{\mathcal{M}^{(1)}} = 2.67$, $\mathcal{M}^{(2)} = 2.8$, $|\hat{\sigma}_c(K)| = 18$

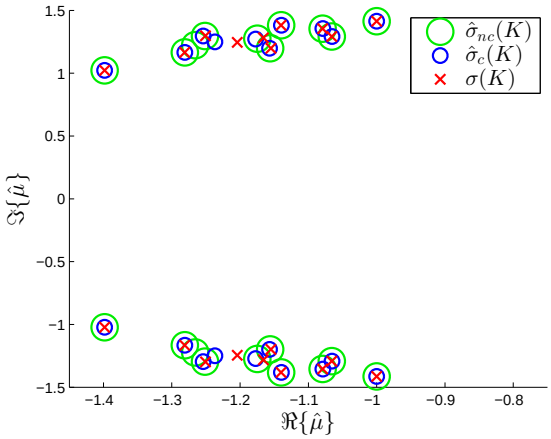


Figure 3.26: $d = 1$, $r_{\mathcal{M}^{(1)}} = 1.5$, $\mathcal{M}^{(2)} = 1.93$, $|\hat{\sigma}_c(K)| = 20$

Values of $\langle \mathbf{err}_c^{(i)} \rangle$ and $\langle \mathbf{err}_{nc}^{(i)} \rangle$ The values of the mean prediction errors on the first two spectral moments displayed in Figure 3.23 is similar to the case where no row is removed (see

Figure 3.7). High values for $\langle \text{err}_c^{(2)} \rangle$ are probably due to simulations where the number of outliers in $\hat{\sigma}_{nc}(K)$ is higher than $q = 2$ and can consequently not all be removed in $\hat{\sigma}_c(K)$. Note, though, that $\langle \text{err}_c^{(i)} \rangle$ is (almost) always lower than $\langle \text{err}_{nc}^{(i)} \rangle$, which is not observed when no row is removed in addition to columns for cross-validation.

n=100

For $n = 100$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.27 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.28.

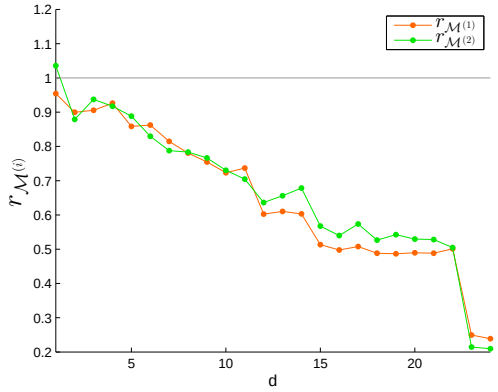
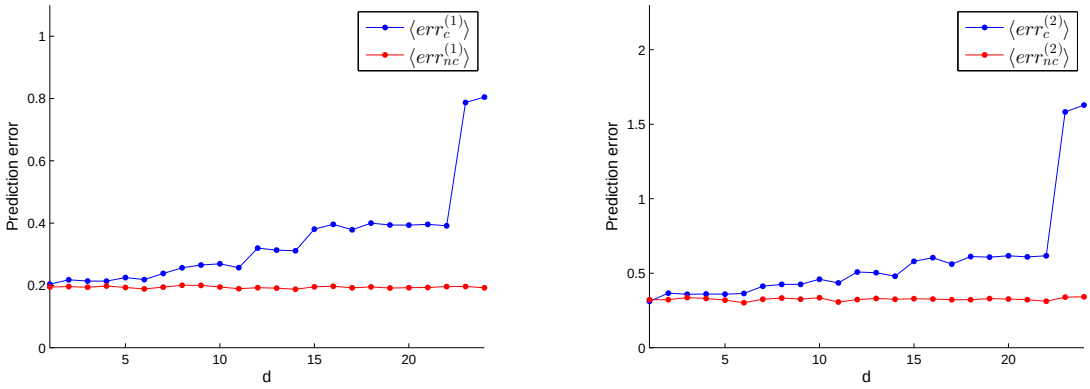


Figure 3.27: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 100$, d varying, a linear local dynamics, $q = 2$



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.28: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 100$, d varying and a linear local dynamics, $q = 2$

No improvement In case of spectra to be estimated of large size, removing rows in addition to columns for cross-validation does unfortunately not seem to improve the estimation of $\sigma(K)$. The behavior of $r_{\mathcal{M}^{(i)}}$ does not vary whether rows are removed or not, as can be deduced by comparing Figures 3.27 and 3.12. These observations could be explained by the following. Discarding rows essentially enables to get rid of outliers in the estimated spectrum. But, as has been noted above, less outliers seem to be present in $\hat{\sigma}_{nc}(K)$ for $n = 100$ and they tend to have a smaller real part than for $n = 10$. As a result of this, removing them has a less significant

impact for large values of n and it occurs less often. Furthermore, it has been observed that $|\hat{\sigma}_c(K)| = 20$ more often than for $n = 10$, which implies that, when there are not outliers to be discarded, decreasing the size of $\hat{\sigma}_c(K)$ worsens the estimation of $\sigma(K)$.
The values and behavior of the mean prediction errors in Figure 3.28 are similar to the case where no row is removed and the same analysis applies.

3.3.3 Results for nonlinear local dynamics

n=10

For $n = 10$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.29 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.30.

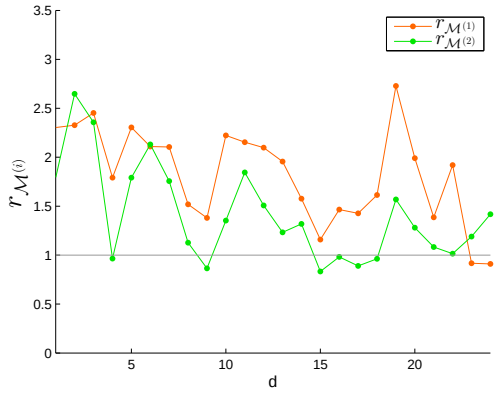
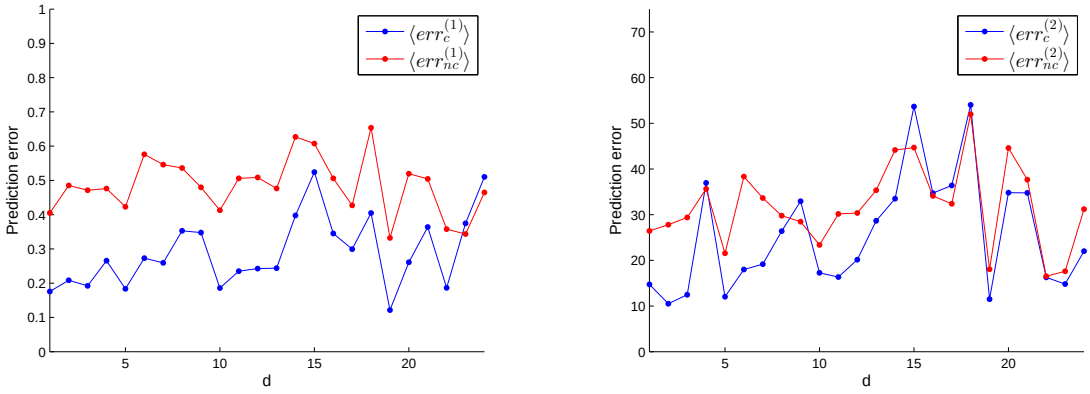


Figure 3.29: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 10$, d varying, a nonlocal linear dynamics, $q = 2$



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.30: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 10$, d varying and a nonlinear local dynamics, $q = 2$

These results are similar to the ones obtained in case of linear dynamics and the same analysis applies. Note that the values of $r_{\mathcal{M}^{(i)}}$ are a little higher in case of nonlinear local dynamics for small values of d. It might be due to the same factors as those accounting for ratios larger than one when no row is removed, that is a lower performance of the DMD algorithm and a higher variability between the estimated spectra, compared to the linear case.

n=100

For $n = 100$, the values of $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ are shown in Figure 3.31 and the values of $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ are displayed in Figure 3.32.

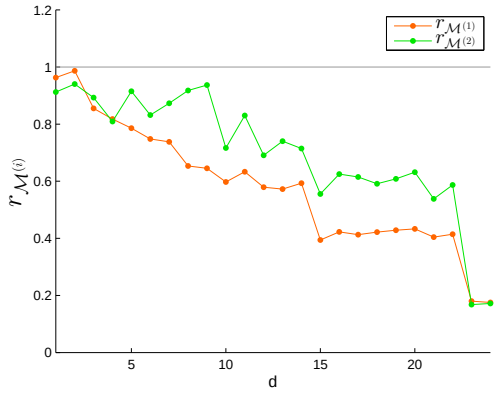
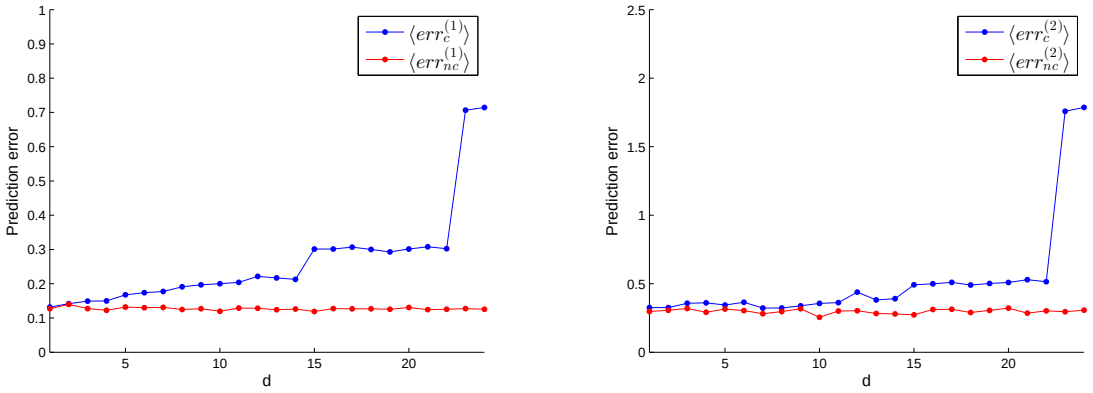


Figure 3.31: $r_{\mathcal{M}^{(1)}}$ and $r_{\mathcal{M}^{(2)}}$ for $n = 100$, d varying, a nonlocal linear dynamics, $q = 2$



(a) $\langle \text{err}_c^{(1)} \rangle$ and $\langle \text{err}_{nc}^{(1)} \rangle$ (b) $\langle \text{err}_c^{(2)} \rangle$ and $\langle \text{err}_{nc}^{(2)} \rangle$

Figure 3.32: $\langle \text{err}_c^{(i)} \rangle$ and $\langle \text{err}_{nc}^{(i)} \rangle$ $(i = 1, 2)$ for $n = 100$, d varying and a nonlinear local dynamics, $q = 2$

These results are again very similar to those obtained in case of linear local dynamics and the same value of n, as can be accounted by the same factors. It is, however, surprising that $r_{\mathcal{M}^{(i)}}$ is smaller than 1 for $d \leq 3$ since it is larger than 1 in case no row is removed (see Figure 3.20). One possible explanation is the following. Note that, in Figure 3.20, the ratios are very close to 1 so, depending on the simulations used to obtain the results, the ratios are a little smaller or slightly larger than 1. Though, it is certainly not the only factor explaining this surprising observation and it would be worthwhile to further investigate it.

Summary

It seems thus from numerical simulations that removing rows in addition to columns from the data matrix Z for cross-validation allows to significantly enhance the DMD algorithm performance but only in case of small values of n. In this setting, it indeed enables to get rid of

outliers among the estimated eigenvalues. For larger values of n, it seems to be best not to use cross-validation at all.

3.4 Does cross-validation allow to enhance the performance of spectral network identification?

3.4.1 Goal of this work: reminder

The primary purpose of this work is to improve the estimation of the global topological properties of the network. In order to achieve this goal, we enhanced the approximation of the dynamics spectrum that is related to the spectral properties of the network, from which information about its structure are inferred. The spectrum of the dynamics can be obtained by means of the DMD algorithm and we showed in the previous two sections of this chapter that its performance can be increased by using cross-validation, and more specifically our method 5 modified such that rows are removed in addition to columns from the data matrix Z. To conclude this work, we will consequently briefly investigate whether our (modified) method also allows or not to enhance the estimation of the topological properties of the network.

3.4.2 Numerical evidence

We consider settings in which it has been shown that algorithm 5 significantly enhances the estimation of the dynamics spectrum, i.e. for $n = 10$ and $q = 2$. We set $d = 10$ and look at the results for linear and nonlinear local dynamics (the simulations settings are the same as in the other results sections). Tables 3.5 and 3.6 display the exact values of the first two spectral moments of the network Laplacian matrix, L, and of the Laplacian matrix of the corresponding unweighted network, $\bar{L}$. It also breaks down their estimates, obtained with and without cross-validation. Table 3.5 gathers the results obtained for a linear local dynamics while Table 3.6 corresponds to a nonlinear dynamics.

	$\mathcal{M}^{(1)}(L)$	$\mathcal{M}^{(2)}(L)$	$d_{mean} = \mathcal{M}^{(1)}(\bar{L})$	$\mathrm{M}^{(2)}(\bar{L})$
Cross-validation	0.122	0.023	2.433	8.388
No cross-validation	0.178	0.041	3.563	15.191
Exact value	0.12	0.025	2.6	10

Table 3.5: Exact and estimated spectral properties of the network for $n = 10$, $d = 10$, $q = 2$ and a linear local dynamics

	$\mathcal{M}^{(1)}(L)$	$\mathcal{M}^{(2)}(L)$	$d_{mean} = \mathcal{M}^{(1)}(\bar{L})$	$\mathrm{M}^{(2)}(\bar{L})$
Cross-validation	0.124	0.015	2.482	5.328
No cross-validation	1.404	1.898	28.078	749.981
Exact value	0.142	0.026	2.6	8.8

Table 3.6: Exact and estimated spectral properties of the network for $n = 10$, $d = 10$, $q = 2$ and a nonlinear local dynamics

These results seem to indicate that using cross-validation as is done in algorithm 5 in combination with removing rows from the data matrix Z enhances the estimation of the spectral properties of the network. Though, this assumption is based on only one simulation for both types of local dynamics so it would be necessary to study the impacts of our method on spectral identification more deeply and thoroughly.

Conclusion

Network systems occur in a wide range of engineering and scientific fields. Since our knowledge about them is incomplete most of the time, spectral network identification is a very promising research topic as it allows to infer global information about their structure from measurements of the dynamics of a few nodes. But this framework is also very recent and is still being developed and improved. The purpose of this work was to enhance the performance of the Dynamic Mode Decomposition algorithm with respect to the estimation of the dynamics spectrum, with the final goal of obtaining a better approximation of the network topological properties. The method we propose is based on the cross-validation technique and aims at generating several estimated spectra from a same set of snapshots of the dynamics. This is achieved by iteratively applying the DMD algorithm to that snapshots set from which several snapshots are removed. The best spectrum is then the one associated with the lowest prediction error regarding the snapshots that have been discarded.

We first exposed numerical evidence that this choice for the selection criterion, i.e. the lowest prediction error, was relevant. We then studied whether using cross-validation as is done in our method effectively allows to improve the estimation of the dynamics spectrum through the DMD algorithm or not. To this end, we compared the exact dynamics spectrum to the ones estimated with and without cross-validation. Since there exists no known mean to compare spectra, we defined the best spectrum as the one enabling to estimate the best the exact first two spectral moments of the dynamics. In case of linear local dynamics, our method does not improve the approximation of the dynamics eigenvalues, this being probably due to the already good performance of the DMD algorithm in this setting and the low variability between the estimated spectra obtained with and without cross-validation. This second factor is mainly observed through the fact that, when the spectrum obtained without cross-validation contains outliers, so does the spectrum estimated by using cross-validation. Note also that, the more snapshots are discarded, the worse is the estimation of the dynamics spectrum since less information is available about the network dynamics. In case of nonlinear local dynamics, the performance of the DMD algorithm is a little lower and a greater variability between the estimated spectra is observed. This accounts for the fact that using cross-validation seems to improve the approximation of the dynamics spectrum. However, barely a slight enhancement is obtained and only when a very few snapshots are removed. We therefore sought to increase the variability between the spectra estimated with and without cross-validation in such a way that outliers among the estimated eigenvalues could be removed. For this purpose, we modified our method by discarding rows in addition to snapshots from the matrix gathering the dynamics snapshots. Significant improvements are observed but only for spectra to be estimated of small size. We finally exposed numerical results indicating that applying our modified method for small to be estimated spectra enhances the estimation of the network topological properties.

Perspectives In this work, we succeeded to enhance the estimation of the dynamics spectrum through the DMD algorithm (at least in some cases) but there is still a wide range of further research perspectives. Firstly, it would be interesting to vary the simulations settings we chose to obtain the numerical results in the third chapter. Studying more deeply the influence of the value of the parameters n, m, S, r, p should be done. It could also be worth of interest to consider different local dynamics than those used in this work and networks characterized by nonidentical local dynamics. Additionally, using other graphs such as grids, stars or chains could be examined. Note also that Erdős-Renyi graphs are not the best suited to represent real-world networks, unlike the Albert-Barabasi model. Secondly, further investigations about the impact of our method on the estimation of the topological properties of the network, which is the final goal of spectral network identification, should be conducted. And finally, the selection criterion that is used to choose the best spectrum in our method does not only evaluate the quality of the Koopman spectrum but also of the associated Koopman modes, which we are not interested in here. As such, it might not be the best selection criterion. However, as modes can be used for clustering [21], our method could be extended to this framework.

Bibliography

[1] M. Porter and J. Gleeson, *Dynamical systems on networks, a tutorial*, vol. 4 of *Frontiers in applied dynamical systems: reviews and tutorials*, ch. 3 *Examples of dynamical systems*. Springer International Publishing, 2016.

[2] A. Mauroy and J. Hendrickx, "Spectral identification of networks using sparse measurements," *SIAM*, vol. 16(1), pp. 479–513, 2017.

[3] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–442, 1998.

[4] J. Cortès, "Networked systems," in *Encyclopedia of Systems and Control*, pp. 849–853, Springer London, 2015.

[5] M. Gevers, "Identification of dynamical newtorks," 2017. Seminar given at the Mathematical Engineering Department at Ecole Polytechnique de Louvain.

[6] P. van den Hof, A. Dankers, P. Heuberger, and X. Bombois, "Identification of dynamic models in complex networks with prediction error methods : basic methods for consistent module estimates," *Automatica*, vol. 49(2013), no. 10, pp. 2994–3006, 2013.

[7] M. Gevers and A. Bazanella, "Identification in dynamic networks: Identifiability and experiment design issues," in *Proceedings of the 54th IEE Conference on Decision and Control*, 2015.

[8] A. Bazanella, M. Gevers, J. Hendrickx, and A. Parraga, "Identifiability of dynamical networks: which nodes need be measured?." submittted to IEEE Conference on Decision and Control, Merlbourne, Australia, December 2017.

[9] M. Timme and J. Casadiego, "Revealing networks from dynamics: an introduction," *Journal of Physics A: Mathematical and Theoretical*, vol. 47, no. 34, pp. 343001–343036, 2014.

[10] D. Napoletani and T. D. Sauer, "Reconstructing the topology of sparsely connected dynamical networks," *Physical Review E, 77*, p. 26103, 2008.

[11] M. Timme, "Revealing network connectivity from response dynamics," *Physical Review Letters*, p. 224101, 2007.

[12] Z. Levnajić and A. Pikovsky, "Untangling complex dynamical systems via derivative-variable correlations," *Scientific Reports*, vol. 4, 2014.

[13] S. G. Shandilya and M. Timme, "Inferring network topology from complex dynamics," *New Journal of Physics*, p. 13004, 2011.

[14] D. Yu, M. Righero, and L. Kocarev, "Estimating topology of networks," *Physical Review Letters*, vol. 97, p. 188701, 2006.

[15] D. Yu and U. Parlitz, "Driving a network to steady states reveals its cooperative architecture," *Europhysics Letters*, vol. 81, p. 48007, 2008.

[16] T. He, X. Lu, X. Wu, J. Lu, and W. X. Zheng, "Optimization-based structure identification of dynamical networks," *Physica A: Statistical Mechanics and its Applications*, vol. 392, pp. 1038–1049, 2013.

[17] W.-X. Wang, R. Yang, Y.-C. Lai, V. Kovanis, and M. A. F. Harrison, "Time-series–based prediction of complex oscillator networks via compressive sensing," *Europhysics Letters*, vol. 94, p. 48006, 2011.

[18] J. Ren, W.-X. Wang, B. Li, and Y.-C. Lai, "Noise bridges dynamical correlation and topology in coupled oscillator networks," *Physical Review Letters*, vol. 104, p. 58701, 2010.

[19] E. Erhardt, S. Rachakonda, E. Bedrick, E. Allen, T. Adali, and V. Calhoun, "Comparison of multi-subject ica methods for analysis of fmri data," *Human brain mapping*, vol. 32, pp. 2075–2095, 2011.

[20] E. Schneidman, M. J. Berry, R. Segev, and W. Bialek, "Weak pairwise correlations imply strongly correlated network states in a neural population," *Nature*, vol. 440, pp. 1007–1012, 2006.

[21] A. Mauroy and J. Hendrickx, "Spectral identification of networks with inputs." submittted to 56th IEEE Conference on Decision and Control, 2017.

[22] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, pp. 1031–1036, 2013.

[23] T.-M.-D. Tran and A. Y. Kibangou, "Distributed estimation of laplacian eigenvalues via constrained consensus optimization problems," *Systems and Control Letters*, vol. 80, pp. 56–62, 2015.

[24] T. Sahai, A. Speranzon, and A. Banaszuk, "Hearing the clusters of a graph: A distributed algorithm," *Automatica*, vol. 48, pp. 15–24, 2012.

[25] C. Kreutz and J. Timmer, "Observable," in *Encyclopedia of Systems Biology*, pp. 1557–1558, Springer New York, 2013.

[26] I. Mezić, "Analysis of fluid flows via spectral properties of the koopman operator," *Annual Review of Fluid Mechanics*, vol. 45, pp. 357–378, 2013.

[27] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos*, vol. 22, p. 047510, 2012.

[28] "Clustering," in *Encyclopedia of Machine Learning and Data Mining*, Springer US, 2017.

[29] C. M. Bishop, *Pattern recognition and machine learning*, ch. 9 *Mixture models and EM*. Springer, 2006.

[30] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[31] C. W. Rowley, I. Mezić, S. Bagheri, and P. Schlatter, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, pp. 115–127, 2009.

[32] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied Numerical Mathematics*, vol. 43, p. 9–44, 2002.

[33] J. Tu, C. Rowley, D. Luchtenburg, S. Brunton, and J. Kutz, "On dynamic mode decomposition: theory and applications," *Journal of Computation Dynamics*, vol. 1, pp. 391–421, 2014.

[34] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the koopman operator: extending dynamic mode decomposition," *Journal of Nonlinear Science*, 2015.

[35] K. K. Chen, J. H. Tu, and C. W. Rowley, "Variants of dynamic mode decomposition: boundary conditions, koopman, and fourier analyses," *Journal of Nonlinear Science*, vol. 22, p. 887–915, 2012.

[36] T. M. Mitchell, *Machine learning*, ch. 1 *Introduction*. McGraw-Hill Education, 1997.

[37] T. Herlau, M. N. Schmidt, and M. Morup, "Introduction to machine learning and data mining." Lecture notes of the course of the same name given at DTU (Technical University of Denmark), 2016.

[38] C. Sammut and G. I. Webb, "Supervised learning," in *Encyclopedia of Machine Learning and Data Mining*, Springer US, 2017.

[39] P. Stone, "Reinforcement learning," in *Encyclopedia of Machine Learning and Data Mining*, Springer US, 2017.

[40] S. de Lange, M. de Reus, and M. van den Heuvel, "The laplacian spectrum of neural networks," *frontiers in computational neuroscience*, 2014.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**