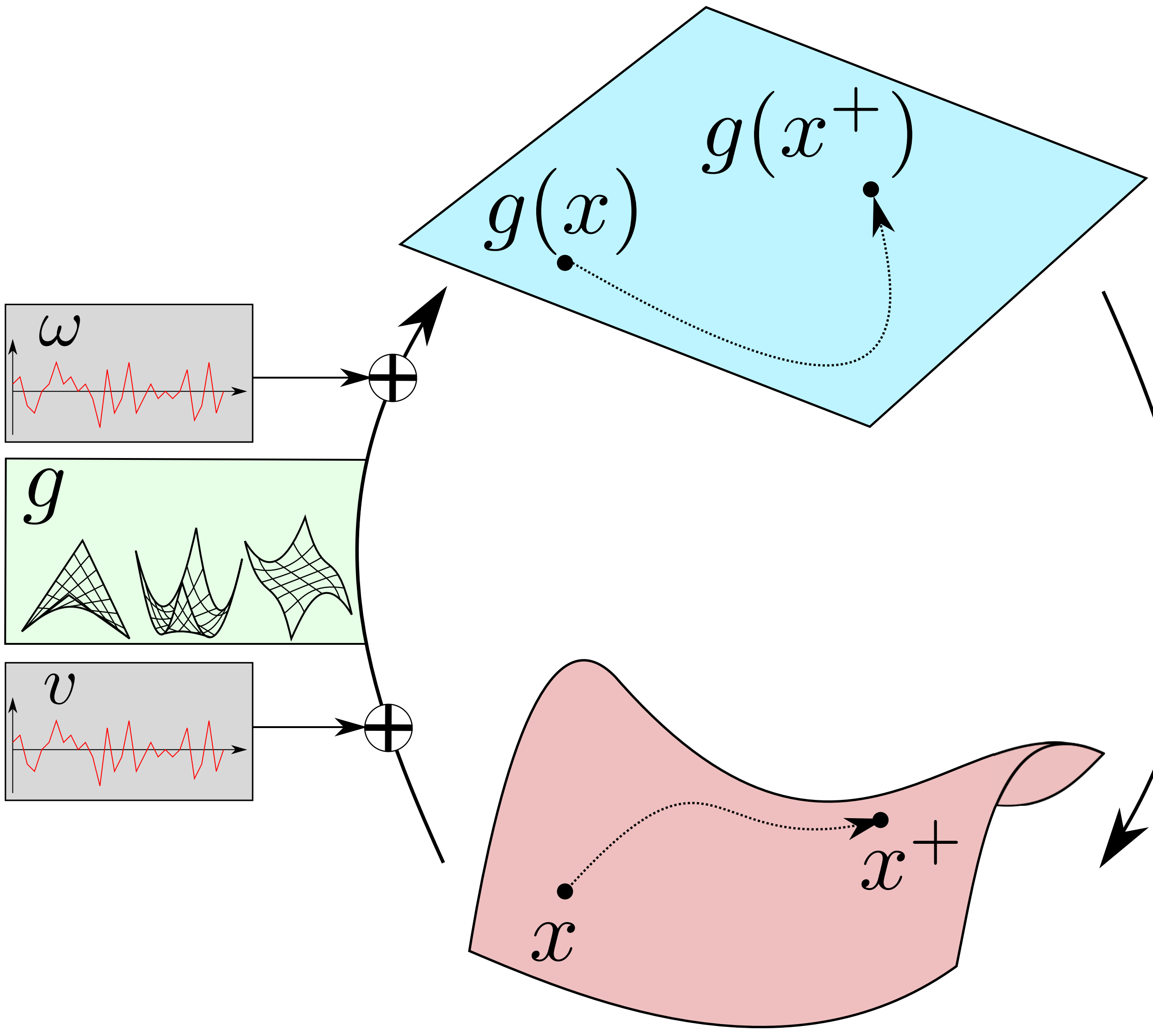
Koopman Subspace Identification in the Presence of Measurement Noise

Alessandro Borghi

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

Koopman Subspace Identification in the Presence of Measurement Noise

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Alessandro Borghi

September 19, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Koopman Subspace Identification in the Presence of Measurement Noise

by

Alessandro Borghi

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 19, 2021

Supervisor(s):

Dr.-Ing. Sander Wahls

Reader(s):

Dr. Manuel Mazo Espinosa

N/A

N/A

Abstract

The ability to compute models that correctly predict the trajectories of a nonlinear system can become a significant challenge in systems and control. The introduction of Koopman operator theory helped to deal with this challenge. The Koopman operator is a composition operator that globally describes a nonlinear system in an infinite-dimensional linear framework. To implement this theory, the usual approach is to approximate the Koopman operator through data-driven methods. These algorithms use measurements of the nonlinear system to compute the approximated operator.

Generally, noise can be present in real-world scenarios. Noisy measurements can have a considerable deteriorating effect on the data-driven approximation of Koopman operators. The approximation of this operator in presence of noisy training data is a necessary step for its implementation to a wider spectrum of real-world applications. Many robust numerical methods were designed to solve this issue. Koopman subspace identification (KSI) is a promising approach. As the name suggests, this algorithm employs subspace identification modeling to compute the matrix approximation of the Koopman operator. In this work, we test KSI against other state-of-the-art techniques. Additionally, we improve its performance in predicting the state trajectories of the nonlinear system in the presence of noisy measurements. To this end, we propose a reducing-order routine that computes the most robust model against measurement noise. Furthermore, a randomized singular value decomposition is adopted to reduce computational times. The improved KSI is then compared against the other state-of-the-art algorithms in the presence of noisy data sets. We will show that the upgraded KSI outperforms most of the other techniques.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Dr.-Ing. Sander Wahls for letting me choose the topic I was interested in, for his assistance during this thesis, and the pleasant discussions during our weekly meetings.

I want to thank my parents for their support. They gave me the strength to get through these hard times, listened to my issues, and helped me make the right decisions. *Grazie mamma, grazie papi, vi voglio bene.*

Thanks also to my grandparents, my cousins, uncle and aunt for helping me see the bigger picture and for making me talk about something that was not my thesis.

To conclude I would like to thank my friends for helping me to not feel lonely during the Covid lockdown. For their support in making hard decisions and during difficult times. Even if this Covid period was tough for everyone, we helped each other to lighten it up. Thank you.

Delft, University of Technology
September 19, 2021

Alessandro Borghi

“Noise? Only God knows it”
— *Francesca Andretta*

Chapter 1

Introduction

The abundance of nonlinear systems in the real world has attracted a lot of attention from the scientific community. The analysis and computation of such systems is, still our days, a major goal in research. Many of the adopted solutions are based on linear control theory which approximates the local dynamics of a system to the linearization around a hyperbolic equilibrium point (see [2, Section 8.2]). Additionally, as stated in [3, Section 1], this local approximation may not describe the entire dynamics in the state space. This disadvantage might prevent the application of linear control techniques for certain nonlinear systems, depending on the needs of the user.

With the increased abundance of data in the last decade, new techniques were developed to analyze and model the properties and dynamics of real systems. Many of these techniques come from the field of machine learning [4]. Among these approaches, the Koopman operator framework started to gain attention for its ability to globally describe a nonlinear system in a linear infinite-dimensional framework. First proposed by Bernard Koopman [5] in 1931 and then extended by Koopman and Von Neumann [6], the Koopman operator initially did not attract much interest from the scientific community. This disinterest was due to the impossibility of implementing it numerically because of the low computational power present at the time. Most of the attention given to the topic in the last decade is due to the pioneering work of Mezić et al. [7, 8]. They introduced the concept of Koopman operator theory to modern engineering problems, such as model reduction and big data applications for dynamical systems.

At the core of the Koopman operator theory lies the concept of observables. The observables are functions of the states of the underlying system. They play a crucial role as they *"lift"* the dynamics of the states into an infinite-dimensional space. If the observables are correctly chosen, then their evolution is linear. The computation of the trajectory evolution in this observables space is usually carried out through approximations. This made the computation of an approximated version of the Koopman operator a pivotal goal. Many numerical methods were developed to achieve this. The most famous one is the dynamic mode decomposition (DMD), defined by Schmid [9], and connected to the Koopman operator theory by Rowley et al. [10]. The DMD has been improved in many different ways. A few examples are the extended dynamic mode decomposition (EDMD) by Williams et al. [1] and the dynamic mode decomposition with control (DMDc) by Proctor et al. [11]. The field of subspace identification modeling also helped the design of algorithms for the computation of the approximated Koopman operator. The most important examples are the subspace dynamic mode decomposition (subDMD) of Takeishi et al. [12] and the subspace-based method of Lian et al. [13]. These techniques rely on the assumption that the lifted dynamics of the nonlinear system lie in a linear space. In this thesis, we will make a reformulation of the algorithm of [13] following the book of Verhaegen et al. [14]. We refer to this algorithm as the Koopman subspace identification (KSI) method. Additionally, we improve KSI with two modifications. First, we introduce a reduced-order technique to compensate for possible noise in the training data set. Second, we adopt the randomized singular value decomposition (rSVD) to reduce computational times without excessively affecting performance.

One of the standard issues in machine learning is the presence of noise in the training data [15]. Noisy data can drastically deteriorate the predictive performance of the model. This issue is present also in the computation of the approximated Koopman operator. In particular, the presence of noisy training data can negatively impact DMD [16]. To model a system with noisy measurements, the development of an algorithm robust to noise is needed. Numerical methods like total dynamic mode decomposition (tDMD) of Hemati et al. [17] and subDMD [12] were designed to overcome noise-corrupted training data. In this thesis, we benchmark the performances of some of the algorithms above mentioned with noisy data sets. In particular, we want to analyze the effect of two types of noise: Additive noise in the observables measurements and additive noise in the measurements of the states of the underlying system. To the best of our knowledge, this is the first work that realizes a benchmark framework to test the listed algorithms in the presence of the additive noise mentioned above.

This thesis is organized as follows: In Chapter 2 we introduce the theory behind the Koopman operator framework. Chapter 3 lists and describes all the state-of-the-art numerical methods that we chose to evaluate. More in detail: tDMD, EDMD along with a noise-free version of it named NFEDMD, subDMD, and KSI are explained. In Chapter 4 the evaluation results of the above-mentioned algorithms are showed and discussed. The improvements on KSI designed during this thesis are proposed and described in Chapter 5, along with an examination of their performance. Finally, in Chapter 6 the conclusions based on the showed results are presented.

Chapter 2

The Koopman Operator Framework

The theory behind dynamical systems was first defined by Henry Poincaré through his geometric viewpoint [7], which originated from his work on the three-body problem in celestial mechanics. Poincaré's geometric viewpoint, or qualitative approach, focuses on the study of the overall behavior of a system's dynamics in the state space. The dynamic behaviour is ruled by ordinary differential equations (ODEs). This was a leap forward from the classic computation of single ODE trajectories from specific initial conditions (see Section 1 of [18]). In this work we will use the following notation to define systems states trajectories:

$$\dot{x}(t) = f(x(t)),\ t \in \mathbb{R}^+, \quad \text{for continuous time, and} \tag{2-1}$$

$$x[k+1] = T(x[k]),\ k \in \mathbb{N}_0, \quad \text{for discrete time,} \tag{2-2}$$

where x belongs to the state space $\mathcal{M}$ of the system[1], $\dot{x} = \frac{dx}{dt}$, and $f : \mathcal{M} \to \mathcal{M}$ as well as $T : \mathcal{M} \to \mathcal{M}$ are possibly nonlinear transformations in the state space. The dependence on the time variable might be omitted during the discussions in this thesis, i.e. $x = x(t)$ or $x = x[k]$ depending on the context. Assuming that a solution for the system in Eq. (2-1) exists, then the flow map F^t is defined as the map that, given the initial condition $x_0 = x(0)$, results the value of x at time t:

$$F^t(x_0) = x_0 + \int_{s=0}^{t} f(x(s))ds.$$

Regarding complex systems, including nonlinear ones, the work of Budisic et al. [7] states: "However, the geometric viewpoint is ill-suited to many of the situations that are of interest in real systems." (Section 1, Page 2). A few examples that the paper makes to emphasize this issue are: the difficulty in computing the model of a system in presence of noise, uncertainty, or in the case of a large number of states. Additionally, Section 5 of Jones [18] explains that traditional modeling methodologies need to be updated to face the complexity of today's systems. Finding a solution to this issue has attracted a lot of attention from many research areas. The large amount of data available today gave the possibility to develop new modeling

[1]The state space of the system can be, for example, a manifold or an Euclidean space.

techniques. In particular, in the last decade, much attention has been given to employ the Koopman operator framework in the analysis of nonlinear systems from a theoretical and practical perspective.

This chapter is taken from Chapter 2 of the author's literature survey titled *"Koopman Operator Theory: a new way to analyze nonlinear systems"*. In the following sections, we will go through the basics of the Koopman operator along with its spectral analysis, mode decomposition and the fundamentals on how to approximate it.

2-1 Koopman Operator Theory

The Koopman operator $\mathcal{K}$ is a composition operator, where the function composition is between the *observables* $g : \mathcal{M} \to \mathbb{C}$ and the flow map $F^{\tau}(x(t))$, or $T(x[k])$ in the discrete case. The operator $\mathcal{K}$ describes the "lifted" evolution of the states, i.e. g, in a Hilbert space $\mathscr{H}$ [19], or, more general, an infinite-dimensional space of observables. More in detail, the dynamics of the underlying system, such as the one in Eq. (2-1) or Eq. (2-2), are lifted through the use of observables $g(x)$, i.e functions of the states. The dynamics of $g(x)$ are then described through the Koopman operator. For our case, we will focus on nonlinear dynamics in the state space.

Definition 2-1.1 (Koopman Operator in continuous time)**.** Let us consider a Hilbert space $\mathscr{H}$ of observables $g : \mathcal{M} \to \mathbb{C}$ where $\mathcal{M}$ is the state space of the system in Eq. (2-1). The Koopman operator $\mathcal{K}^{\tau} : \mathscr{H} \to \mathscr{H}$, with $\tau \in \mathbb{R}^+$ and is associated with the nonlinear transformation $f : \mathcal{M} \to \mathcal{M}$, is defined as

$$\mathcal{K}^{\tau} g(x(t)) = g \circ F^{\tau}(x(t)) \qquad \forall g \in \mathscr{H}, t \in \mathbb{R}^+$$

where $\circ$ is the function composition, i.e. $f \circ h(x) = f(h(x))$.

Definition 2-1.2 (Koopman Operator in discrete time)**.** Let us consider a Hilbert space $\mathscr{H}$ of observables $g : \mathcal{M} \to \mathbb{C}$ where $\mathcal{M}$ is the state space of the system in Eq. (2-2). The Koopman operator $\mathcal{K} : \mathscr{H} \to \mathscr{H}$, associated with the nonlinear transformation $T : \mathcal{M} \to \mathcal{M}$, is defined as

$$\mathcal{K} g(x[k]) = g \circ T(x[k]) \quad \forall g \in \mathscr{H},$$

with $k \in \mathbb{N}_0$.

These definitions were taken from Definition 1.2 and Definition 1.1, respectively, of [20]. A good representation of the Koopman operator framework is present in Fig. 2-1, where the observables $g(x)$ used to lift the dynamics are chosen to form a suitable basis for the observables space. On the other side, to return to the state space, the Koopman mode decomposition (KMD) (λ, ψ, v) is adopted. The KMD elements will be the topic of the next section.

The linearity of the Koopman operator is one of the main strengths of this approach. As a matter of fact, describing the behaviour of a nonlinear system in a "lifted" linear space, gives us the ability to apply already known linear control and identification techniques. For proving the linearity of $\mathcal{K}$, we can use as an example Eq. (2-2) and take the following steps:

$$\begin{aligned}\mathcal{K}[g_1 + g_2](x[k]) &= [g_1 + g_2](T(x[k])) = [g_1 + g_2](x[k+1]) = g_1(x[k+1]) + g_2(x[k+1]) = \\ &= \mathcal{K} g_1(x[k]) + \mathcal{K} g_2(x[k]),\end{aligned}$$

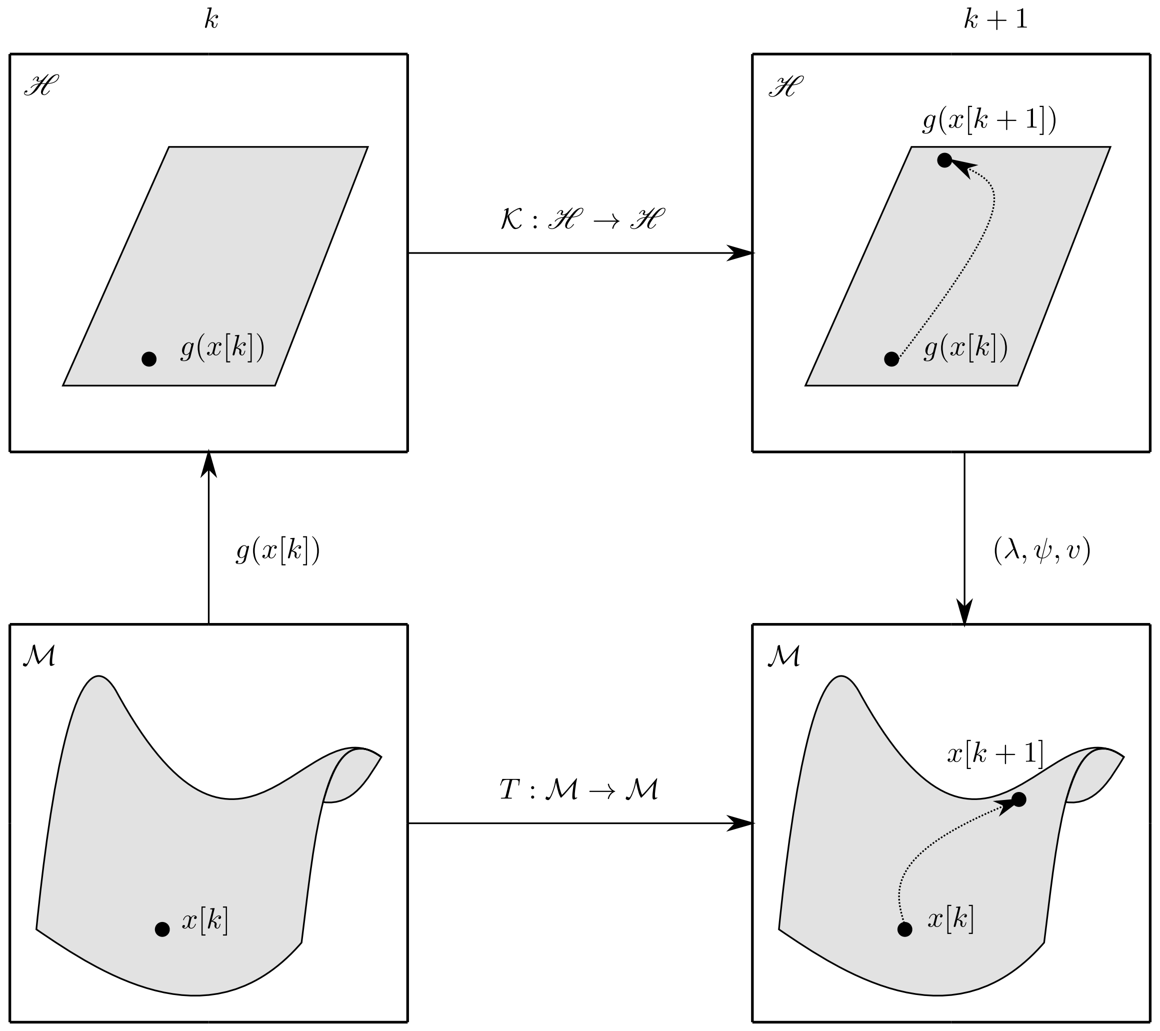


Figure 2-1: Scheme of the Koopman operator framework based on Figure 1 in Williams et al. [1]. In the figure, it is possible to see the lifting of the nonlinear dynamics of the state space $\mathcal{M}$ into a linear infinite-dimensional framework in the space $\mathscr{H}$ through the use of observables g. As the computation of the next step through the nonlinear mapping T is generally expensive and complex, the linear operator $\mathcal{K}$ can be used to evolve the observables. To come back to the state space, the Koopman mode decomposition (λ, ψ, v) is needed

Considering the continuous time-invariant system in Eq. (2-1), we have that the Koopman operator $\mathcal{K}^\tau$ forms an operator *semigroup* for $\tau \geq 0$. The following definition can be found in Page 6 of [20] and in Slide 9 of [21].

Definition 2-1.3 (Koopman Semigroup)**.** A Koopman operator semigroup in a Hilbert space $\mathscr{H}$ is a family of operators $\mathcal{K}^\tau : \mathscr{H} \to \mathscr{H}$ such that:

(i) $\mathcal{K}^{\tau_1+\tau_2} g = \mathcal{K}^{\tau_1}\mathcal{K}^{\tau_2} g$,

(ii) $\mathcal{K}^0 = I$.

Any semigroup $\mathcal{K}^\tau$ can be described through its *infinitesimal generator* $L_{\mathcal{K}}$.

Definition 2-1.4 (Infinitesimal Generator)**.** An infinitesimal generator $L_{\mathcal{K}}$ of a Koopman semigroup $\mathcal{K}^\tau$ is the operator given by the limit

$$L_{\mathcal{K}} g = \lim_{\tau \to 0} \frac{\mathcal{K}^\tau g - g}{\tau}.$$

In other words, the infinitesimal generator can be seen as a derivative in the space of observables. This definition will be useful for the next section in which the spectral properties of the Koopman operator will be discussed.

2-2 Koopman Spectral Analysis

A strong point of the linear Koopman operator framework is its ability to describe the global behaviour of a nonlinear system through the spectral decomposition of the linear operator $\mathcal{K}$. What follows are the definitions of the Koopman eigenvalues and eigenfunctions for continuous and discrete time systems. The following definitions rely on Definition 1.4 and Definition 1.3, respectively, of [20].

Definition 2-2.1 (Koopman eigenfunctions and eigenvalues in continuous time)**.** An eigenfunction $\psi(x) : \mathcal{M} \to \mathbb{C}$ of the Koopman semigroup $\mathcal{K}^\tau$, $\tau \geq 0$, associated with $f : \mathcal{M} \to \mathcal{M}$ is an observable $\psi \in \mathscr{H} \backslash \{0\}$ that satisfies the eigenvalue equation

$$\mathcal{K}^\tau \psi(x) = \psi \circ f(x) = e^{\lambda_c \tau} \psi(x),$$

where $\lambda_c \in \mathbb{C}$ is the corresponding eigenvalue.

Considering the continuous time-invariant system in Eq. (2-1), Eq. 7 in Section A of Mauroy et al. [22] shows that the Koopman eigenfunctions and eigenvalues can be computed through the following partial differential equation (PDE):

$$L_{\mathcal{K}} \psi(x) = f(x) \nabla \psi(x) = \lambda_c \psi(x). \tag{2-3}$$

Note that from Eq. (2-3) we have that $L_{\mathcal{K}} = f(x)\nabla$.

Definition 2-2.2 (Koopman eigenfunctions and eigenvalues in discrete time)**.** An eigenfunction $\psi(x) : \mathcal{M} \to \mathbb{C}$ of the Koopman operator $\mathcal{K}$ associated with $T : \mathcal{M} \to \mathcal{M}$ is an observable $\psi \in \mathscr{H} \backslash \{0\}$ that satisfies the eigenvalue equation

$$\mathcal{K} \psi(x) = \psi \circ T(x) = \lambda \psi(x),$$

where $\lambda \in \mathbb{C}$ is the corresponding eigenvalue.

Furthermore, Let us assume that $\psi_1 \in \mathscr{H}$ and $\psi_2 \in \mathscr{H}$ are two eigenfunctions of the same Koopman operator $\mathcal{K}$, such that $\mathcal{K}\psi_1(x) = \lambda_1 \psi_1(x)$ and $\mathcal{K}\psi_2(x) = \lambda_2 \psi_2(x)$. We also assume that their product is $\psi_1 \psi_2 \in \mathscr{H}$. Then $\psi_1 \psi_2$ is an additional eigenfunction of $\mathcal{K}$ with eigenvalue $\lambda_1 \lambda_2$:

$$\mathcal{K}(\psi_1 \psi_2)(x) = \psi_1(T(x))\psi_2(T(x)) = \mathcal{K}\psi_1(x)\mathcal{K}\psi_2(x) = \lambda_1 \lambda_2 \psi_1 \psi_2.$$

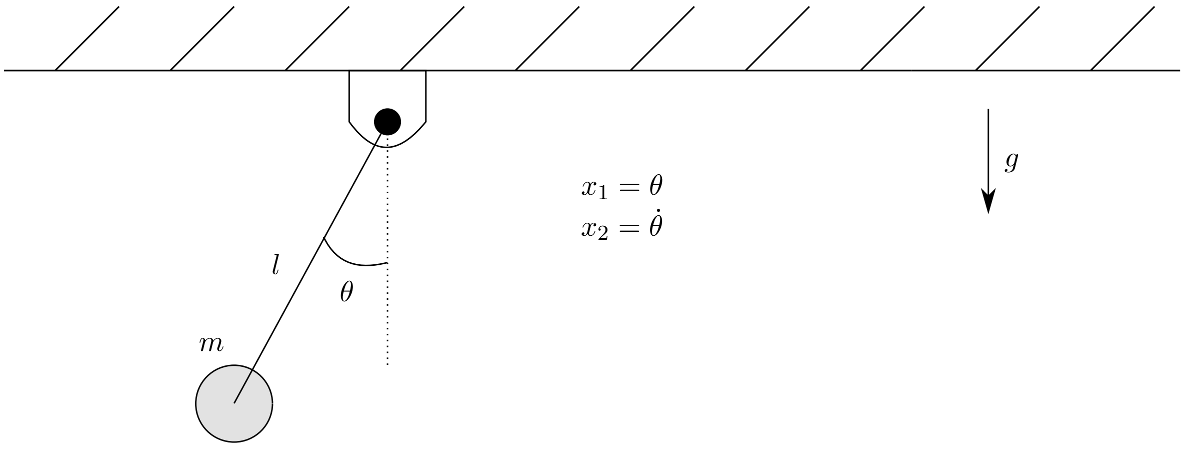


Figure 2-2: Scheme of a pendulum

Note that, as Arbabi states in Section 4 of [23], the spectrum of the Koopman operator can be of three types: point, continuous and mixed spectrum. For what regards this thesis, we will assume that the spectrum present in the considered cases has negligible continuous and mixed spectrum. [24]

The following is a good example to gain a better understanding of the importance of Koopman eigenfunctions and their ability to describe nonlinear systems.

Example 1. This example has been inspired from Example 12 in [4]. Consider the pendulum in Fig. 2-2 were the dynamics will be described by a nonlinear differential equation

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -\frac{g}{l}\sin(x_1), \end{aligned} \tag{2-4}$$

where g is the acceleration of gravity, l is the length of the pendulum wire, x_1 is the angle of the pendulum, and x_2 is its angular velocity.

Let us define the mechanical energy of the system in Eq. (2-4) as the sum of the kinetic ($\frac{1}{2}ml^2x_2^2$) and potential ($mg\cos(x_1)$) energies

$$E = \frac{1}{2}ml^2x_2^2 - mg\cos(x_1),$$

where m is the mass at the end of the pendulum. If this system is considered with no friction and no external forces, other than its weight, then its mechanical energy E will be conserved. Taking into account that E is a function of the states x_1 and x_2, we can assume that the mechanical energy is an eigenfunction of the system with eigenvalue λ (see Definition 2-2.1). Assuming the energy is conserved, i.e. $\dot{E} = 0$, then the dynamics of the energy are invariant to the ones of the pendulum. The invariancy of the eigenfunction mirrors the dynamics of the system and, as a matter of fact, the states of the pendulum (angle and angular velocity) stay in the same set for $t \to \infty$ as long as $\dot{E} = 0$.

$\triangle$

2-2-1 Koopman Modes

The eigenfunctions ψ_j of the Koopman operator can form a basis of the Hilbert space $\mathscr{H}$ of observables if the continuous and mixed parts of the spectrum are negligible. In that case, it is possible to define all the observables $g \in \mathscr{H}$ as a linear combination of ψ_j:

$$g(x) = \sum_{j=1}^{\infty} \psi_j(x) v_j = \sum_{j=1}^{\infty} \psi_j(x) \langle \psi_j, g \rangle, \tag{2-5}$$

where $v_j \in \mathbb{C}$ is the j-th *Koopman mode* associated with ψ_j and $\langle \cdot, \cdot \rangle$ is the inner product[2]. The Koopman modes v_j, as shown in Eq. (2-5), are coefficients of projections of g in the eigenfunctions ψ (see Section 2 of Georgescu et al. [26]).

Considering the discrete dynamics in Eq. (2-2), then the dynamics of g can be defined through the spectral properties in Definition 2-2.2 of the Koopman operator $\mathcal{K}$

$$\begin{aligned} \mathcal{K} g(x) = g \circ T(x) = \sum_{j=1}^{\infty} \psi_j(T(x)) v_j = \sum_{j=1}^{\infty} \mathcal{K} \psi_j(x) v_j = \\ = \sum_{j=1}^{\infty} \lambda_j \psi_j(x) v_j. \end{aligned} \tag{2-6}$$

The sequence of triplets $\{(\lambda_j, \psi_j, v_j)\}_{j=0}^{\infty}$ is also referred to as Koopman mode decomposition (KMD) in [25]. The evolution of the observables described through KMD is a focal point in data-driven analysis [27]. This is because the Koopman modes give an insight on the underlying dynamics of the observables for different eigenvalues. Connections between KMD and the dynamic mode decomposition algorithm were also established by Rowley et al. [28] (discussed in Section 3-1). This led to a wide number of applications of the Koopman modes, from modeling the energy of a building [26] to financial trading strategies [29].

2-3 Approximating the Koopman Operator: Towards Numerical Implementation

As already mentioned earlier, the Koopman operator evolves observables from an infinite-dimensional space. The usual way to make this kind of computation is through an approximation of the Koopman operator. We will define the matrix approximation of $\mathcal{K}$ as $\mathbf{K} : \mathbb{C}^p \to \mathbb{C}^p$, where p is the dimension of the basis of a subspace of $\mathscr{H}$. To do so, the choice of the basis of the observables space is crucial. As a matter of fact, to be able to approximate $\mathcal{K}$, the chosen basis must span a subspace of $\mathscr{H}$ called *Koopman-invariant subspace*. The following definition has been taken from Section 1.4.1 [20].

Definition 2-3.1 (Koopman-invariant subspace)**.** Let $\mathscr{A}_{\mathscr{H}} = \text{span}\{g_1, g_2, \ldots, g_p\}$, with p finite, be a finite dimensional linear subspace of a Hilbert space $\mathscr{H}$. Having

$$g \in \mathscr{A}_{\mathscr{H}} \tag{2-7}$$

[2]Note that to be able to describe the Koopman modes through projection (the inner product), we assumed that the underlying dynamical system in the state space is conservative [25]

then $\mathscr{A}_{\mathscr{H}}$ is Koopman-invariant if $\forall g$ we have

$$\Pi \mathcal{K} g \in \mathscr{A}_{\mathscr{H}}, \tag{2-8}$$

where $\Pi : \mathscr{H} \to \mathscr{A}_{\mathscr{H}}$ is a projection operator.

The Koopman operator restricted to $\mathscr{A}_{\mathscr{H}}$ can be represented exactly by a finite-dimensional matrix. We will now review the proof of this statement by following Section 1.4.1.1 of [20]. Let us consider a Koopman operator $\mathcal{K}$ acting on the subspace $\mathscr{A}_{\mathscr{H}}$ spanned by the basis of observables $\mathbf{g}^T : \mathbb{C}^p \to \mathscr{A}_{\mathscr{H}}$ defined as

$$\mathbf{g}^T = \begin{bmatrix} g_1 & g_2 & \dots & g_p \end{bmatrix}.$$

Then the operator in Eq. (2-8) is defined as

$$\Pi \mathcal{K} : \mathscr{A}_{\mathscr{H}} \to \mathscr{A}_{\mathscr{H}},$$

where the projection operator $\Pi : \mathscr{H} \to \mathscr{A}_{\mathscr{H}}$ is equal to

$$\Pi = \mathbf{g}^T \Gamma. \tag{2-9}$$

In Eq. (2-9) the operator $\Gamma : \mathscr{H} \to \mathbb{C}^p$ describes g on the basis $\mathbf{g}^T$ such that

$$\Gamma g = \begin{bmatrix} \langle g, g_1 \rangle \\ \langle g, g_2 \rangle \\ \vdots \\ \langle g, g_p \rangle \end{bmatrix}.$$

Taking into account that $\Gamma \mathbf{g}^T = I$, with I being an identity matrix, we have that $\Gamma g = I \Gamma g = \Gamma \Pi g$. Knowing the relation between Γ and g, we can state the following:

$$\Pi \mathcal{K} g = \Pi \mathcal{K} \Pi g = \mathbf{g}^T \Gamma \mathcal{K} \mathbf{g}^T \Gamma g = \mathbf{g}^T \mathbf{K} \Gamma g$$

where $\mathbf{K} = \Gamma \mathcal{K} \mathbf{g}^T$ is the matrix approximation of the Koopman operator in $\mathscr{A}_{\mathscr{H}}$.

The main problem of constructing a Koopman-invariant subspace for approximating $\mathcal{K}$ is finding a suitable basis. Note that, as stated in [25], any set of Koopman eigenfunctions ψ will span a Koopman-invariant subspace. Methods have been implemented to compute ψ so to find a suitable basis and, therefore, $\mathbf{K}$. Some examples of these methodologies are in Mauroy et al. [22] and Korda et al. [30]. Generally, computing the exact eigenfunctions from data is practically challenging. As discussed in Section 2.2 of [25], in a real case scenario it is more probable that we will find an approximated version of the eigenfunctions of $\mathcal{K}$.

Following the discussion in the work of Brunton et al. [31], from a control perspective, the inclusion of the states as observables, i.e. $g_i(x) = x_i$, in the basis for $\mathscr{A}_{\mathscr{H}}$ can simplify the computations and help to understand the dynamics of the system in $\mathcal{M}$. The main issue with this approach is that the approximation of the Koopman operator $\mathbf{K}$ would not be able to represent a system with multiple equilibrium points, restricting the field of applications (see also Section 3.3 of [25]).

Chapter 3

Numerical Methods for Approximating Koopman Operators

In the past decade, the availability of large amounts of data gave the possibility to researchers to design techniques and algorithms that were able to extract fundamental information on the properties of dynamical systems. Much attention has been given to the creation of machine learning algorithms to analyze dynamical systems through Koopman operator theory [4]. As introduced in Chapter 2, the Poincaré geometric viewpoint needs an analytical model to be able to analyze the system at hand. With the complexity present in the systems of today, it can be quite difficult to gain such a model through traditional techniques (see Section 5 of [18]). The Koopman operator framework can be a helpful modeling technique when large amounts of data are available. This is because the Koopman operator is a linear operator and it can be approximated[1] through fully data-driven algorithms. The design and implementation of these numerical approaches gives us the ability to compute the dynamics in the nonlinear state space from a linear space of observables. The linear description of nonlinear systems opens the doors to a wide range of applications. A couple of examples are the model predictive control (MPC) technique developed by Korda et al. [19] and the analysis of power systems stability by Susuki et al. [32].

In this chapter, we will discuss the state-of-the-art data-driven numerical methods that we chose to implement and test for the approximation of the Koopman operator. As most of these applications make use of discrete-time systems, from now on, if it is not explicitly declared the contrary, our focus will be towards the discrete formulation of the Koopman operator framework, i.e. $\mathcal{K}g(x[k]) = g \circ T(x[k])$. Some sections of this chapter (specifically dynamic mode decomposition and extended dynamic mode decomposition) were taken from the literature survey of the author.

[1]This is true assuming that the basis for the Koopman-invariant subspace is correctly chosen and that the discrete spectrum dominates the dynamics in the space of observables.

3-1 Dynamic Mode Decomposition (DMD)

The most famous and widely used method to compute $\mathbf{K}$, specifically its modes, is the dynamic mode decomposition (DMD) algorithm. It was first developed by Schmid [9] in the fluid dynamics community. DMD was designed to retrieve the dynamics of the system from snapshots of high dimensional data in a fully data-driven and equation-free way. As introduced in [9, Section 2.1], the DMD was developed with two main variants: the Arnoldi-approach-based method, and the singular value decomposition (SVD) based method. In this work, we will focus on the SVD based approach. Much of the following definitions and steps were taken from Kutz et al. [33] in Chapter 1 and 3.

Considering the system in Eq. (2-2), the DMD tries to approximate it such that

$$x[k+1] \approx \mathbf{K}x[k], \tag{3-1}$$

with $x[k] \in \mathbb{R}^l$ and $\mathbf{K} \in \mathbb{R}^{l\times l}$. This method relies on taking snapshots of the states of the system[2] which are defined as follows:

$$\mathbb{X} = \begin{bmatrix} | & | & & | \\ x[0] & x[1] & \dots & x[m-1] \\ | & | & & | \end{bmatrix}, \quad \mathbb{X}' = \begin{bmatrix} | & | & & | \\ x[1] & x[2] & \dots & x[m] \\ | & | & & | \end{bmatrix}, \tag{3-2}$$

creating two $l \times m$ matrices, where $x[j] \in \mathbb{R}^l$, $j = 0, \dots, m$, is the state at instant j. Now Eq. (3-1) can be rewritten through the snapshot matrices such that

$$\mathbb{X}' \approx \mathbf{K}\mathbb{X}.$$

The approximately equal symbol $\approx$ is used as the system in Eq. (2-1) or Eq. (2-2) is most likely to be nonlinear, while we are describing it in a linear framework. The main objective now is to find $\mathbf{K}$ so to approximate the nonlinear dynamics. To do so the following minimization problem is solved:

$$\min_{\mathbf{K}} \|\mathbb{X}' - \mathbf{K}\mathbb{X}\|_F, \tag{3-3}$$

where $\|\cdot\|_F$ is the Frobenius norm[3]. The minimization in Eq. (3-3) is easily solvable with the following equation:

$$\mathbf{K} = \mathbb{X}'\mathbb{X}^{\dagger}, \tag{3-4}$$

where † denotes the Moore-Penrose pseudoinverse. From this matrix the DMD extracts the so called *DMD modes* φ_i of $\mathbf{K}$, which are its eigenvectors. Each φ_i corresponds to a particular eigenvalue λ_i of $\mathbf{K}$.

In many applications, the direct use of Eq. (3-4) can be computationally expensive for the high amount of data to process. This is solved by analyzing a rank reduced representation of

[2]To note that the DMD assumes that the states are known and measurable. The inclusion of observables has been done in the EDMD which will be discussed in the next section.

[3]The definition of the Frobenius norm for a $n \times m$ matrix A is the following:

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2} = \sqrt{Tr(AA^H)},$$

where A^H is the Hermitian of A.

$\mathbf{K}$, which, from now on, will be defined as $\tilde{\mathbf{K}}$. To do so, we first define the concept of singular value decomposition (SVD) and proper orthogonal decomposition (POD).

What follows is the singular value decomposition theorem taken from Theorem 2.6 of Verhaegen et al. [14]:

Theorem 3-1.1. Any matrix $\mathbb{X} \in \mathbb{R}^{l\times m}$ can be decomposed as

$$\mathbb{X} = U\Sigma V^T,$$

where $U \in \mathbb{R}^{l\times l}$ and $V \in \mathbb{R}^{m\times m}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{l\times m}$ has its only nonzero elements along the diagonal. These elements ς_i are ordered such that

$$\varsigma_1 \geq \varsigma_2 \geq \cdots \geq \varsigma_q > \varsigma_{q+1} = \cdots = \varsigma_b = 0,$$

where $q = \text{rank}(\mathbb{X})$ and $b = \min(l, m)$.

The elements ς_i are called the *singular values* of $\mathbb{X}$, and the columns of U and V are respectively the *left and right singular vectors* of $\mathbb{X}$.

Following Section 3 of Chattarjee [34], the SVD-based POD describes the dynamics of the snapshot measurements in $\mathbb{X}$ with the following decomposition

$$\mathbb{X}^T = QU^T \tag{3-5}$$

where $\mathbb{X} = U\Sigma V^T$ and $Q = V\Sigma$. Each row of U^T is a *proper orthogonal mode* and Eq. (3-5) is called the POD of $\mathbb{X}$.

Let us now return to the computation of matrix $\tilde{\mathbf{K}}$. We first compute a reduced-order SVD of the snapshot matrix

$$\mathbb{X} \approx U_r\Sigma_r V_r^T,$$

where $U_r \in \mathbb{C}^{l\times r}$, $\Sigma_r \in \mathbb{C}^{r\times r}$, $V_r \in \mathbb{C}^{m\times r}$ and $r \leq l$ is the parameter dictating the chosen degree of $\tilde{\mathbf{K}}$. The rank reduced matrix $\tilde{\mathbf{K}}$ is a $r \times r$ projection of $\mathbf{K}$ onto its POD (see Section 1.3 of [33])). In other words

$$\tilde{\mathbf{K}} = U_r^T\mathbf{K}U_r, \tag{3-6}$$

In other words, the columns of U_r are POD modes of $\mathbb{X}$ and, if used as coordinate change for $\mathbf{K}$, results in the approximate $\tilde{\mathbf{K}}$. Now that we have $\tilde{\mathbf{K}}$, its eigendecomposition results from

$$\tilde{\mathbf{K}}w_i = \lambda_i w_i,$$

where w_i is an eigenvector of $\tilde{\mathbf{K}}$ and λ_i is its corresponding eigenvalue.

The DMD mode corresponding to λ_i, which are specifically called *exact DMD modes* in our case (see also Tu et al. [35]), are computed as follows:

$$\varphi_i = \frac{1}{\lambda_i}\mathbb{X}'V_r\Sigma_r^{-1}w_i.$$

Along with the exact DMD modes, there are also the *projected DMD modes*: $\varphi_i = Uw_i$, but we focus on the former as it has been proven in [35, Theorem 1] that these are the exact eigenvectors of $\mathbf{K}$. Below we will also connect the Koopman modes with the DMD modes.

The following algorithm summarizes the DMD steps described above based on [33] and [35].

```
Algorithm 1: DMD algorithm
Input: $\mathbb{X}$, $\mathbb{X}'$
Output: DMD modes $\Phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_r \end{bmatrix}$
1 Construct the snapshot matrices $\mathbb{X}$ and $\mathbb{X}'$;
2 Compute a reduced order SVD for $\mathbb{X} \approx U_r \Sigma_r V_r^T$;
3 Set $\tilde{\mathbf{K}} = U_r^T \mathbf{K} U_r = U_r^T \mathbb{X}' V_r \Sigma_r^{-1}$ which has the same non zero eigenvalues of $\mathbf{K}$;
4 Compute the spectrum of $\tilde{\mathbf{K}}$ by solving $\tilde{\mathbf{K}} w_i = \lambda_i w_i$;
5 Set $\varphi_i = \frac{1}{\lambda_i} \mathbb{X}' V_r \Sigma_r^{-1} w_i$ for $i = 1, \dots, r$.
```

Assuming that a solution for Eq. (3-3) exists, it is possible to approximately reconstruct the state dynamics by writing

$$x[k+1] = T(x[k]) \approx \sum_{j=1}^{r} \varphi_j \lambda_j^k b_j = \Phi \Lambda^k \mathbf{a}, \tag{3-7}$$

where Λ is a diagonal matrix with the eigenvalues λ_j of $\mathbf{K}$ as entries and $\mathbf{a} = \Phi^\dagger x[0]$ are the initial coefficients values. Let us now assume that the snapshot matrices $\mathbb{X}$ and $\mathbb{X}'$ are linearly consistent. Additionally, let us assume that the direct measurements of the states, i.e. $g(x) = x$, construct a suitable basis for a Koopman-invariant subspace. If these two assumptions are true we have that the DMD modes ϕ_i and eigenvalues λ_i computed in Algorithm 1 are the same as $\mathcal{K}$ (see Section 4.1 of Tu et al. [35]). A deeper study can be seen in Rowley et al. [28] and an overview can be seen in Chapter 3 of Kutz et al. [36]. The main issue of this approach lies in the fixed choice of observables $g(x) = x$ and on the assumption that the direct state measurements are linearly consistent. Indeed, this restricts the applicability of the algorithm for real case scenarios as it is probable that the observables do not build a suitable basis for a Koopman-invariant subspace (see Section 2-3). For the reason above mentioned, the extended dynamic mode decomposition (EDMD) was designed. EDMD includes a library of nonlinear observables functions of x to have a higher probability of approximating the right Koopman operator of $T(x[k])$. The implementation of this algorithm will be discussed in the next section.

3-1-1 Total Dynamic Mode Decomposition (tDMD)

Introduced by Hemati et al. [17] the total dynamic mode decomposition (tDMD) improves the performances of DMD in the presence of measurement noise in the states snapshots $\mathbb{X}$, $\mathbb{X}' \in \mathbb{R}^{l \times m}$. The tDMD algorithm solves a total least squares problem that takes into account the presence of noise in both the snapshots matrices. Let us rewrite the least squares problem of Eq. (3-3) for the overconstrained case, i.e. $l < m$, as follows [17, Section 2.2]

$$\min_{\mathbf{K}, \Delta\mathbb{X}'} \|\Delta\mathbb{X}'\|_F, \quad \text{subject to } \mathbb{X}' + \Delta\mathbb{X}' = \mathbf{K}\mathbb{X}, \tag{3-8}$$

which considers a residual term $\Delta\mathbb{X}$ only in the $\mathbb{X}'$ side of the equation. This leads to an asymmetrical treatment of the noise in the measurements (see Section 2.2 of [17]). To solve this issue, the following total least squares formulation of Eq. (3-8) is given

$$\min_{\mathbf{K}, \Delta\mathbb{X}', \Delta\mathbb{X}} \left\| \begin{bmatrix} \Delta\mathbb{X}' \\ \Delta\mathbb{X} \end{bmatrix} \right\|_F, \quad \text{subject to } \mathbb{X}' + \Delta\mathbb{X}' = \mathbf{K}(\mathbb{X} + \Delta\mathbb{X}). \tag{3-9}$$

The framework of Eq. (3-9) can be also generalized to snapshot matrices of observables. Let us have an observables library $\mathbf{g}(x) \in \mathbb{R}^n$ defined as

$$\mathbf{g}(x[k]) = \begin{bmatrix} g_1(x[k]) \\ g_2(x[k]) \\ \vdots \\ g_n(x[k]) \end{bmatrix},$$

with $g_j : \mathbb{R}^l \to \mathbb{R}$, $j = 1, 2, \ldots, n$. The observables snapshot matrices are arranged as follows

$$\mathbf{g}(\mathbb{X}) = \begin{bmatrix} | & | & & | \\ \mathbf{g}(x[0]) & \mathbf{g}(x[1]) & \ldots & \mathbf{g}(x[m-1]) \\ | & | & & | \end{bmatrix}, \quad \mathbf{g}(\mathbb{X}') = \begin{bmatrix} | & | & & | \\ \mathbf{g}(x[1]) & \mathbf{g}(x[2]) & \ldots & \mathbf{g}(x[m]) \\ | & | & & | \end{bmatrix},$$

with $\mathbf{g}(\mathbb{X})$,$\mathbf{g}(\mathbb{X}') \in \mathbb{R}^{n \times m}$ and $n < m$. Assuming the dynamics of the observables are approximately linear, the total least square problem of Eq. (3-9) can then be formulated as

$$\min_{\mathbf{K}, \Delta\mathbf{g}(\mathbb{X}'), \Delta\mathbf{g}(\mathbb{X})} \left\| \begin{bmatrix} \Delta\mathbf{g}(\mathbb{X}') \\ \Delta\mathbf{g}(\mathbb{X}) \end{bmatrix} \right\|_F, \quad \text{subject to } \mathbf{g}(\mathbb{X}') + \Delta\mathbf{g}(\mathbb{X}') = \mathbf{K}(\mathbf{g}(\mathbb{X}) + \Delta\mathbf{g}(\mathbb{X})). \tag{3-10}$$

The solution of Eq. (3-10) is based on constructing an augmented snapshot matrix (see [17, Section 2.3])

$$\mathbb{Z} := \begin{bmatrix} \mathbf{g}(\mathbb{X}) \\ \mathbf{g}(\mathbb{X}') \end{bmatrix},$$

and project the snapshot matrices $\mathbf{g}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X}')$ into a subspace of $\mathbb{Z}$ called $\mathbb{Z}_r$, where $r \leq n$ is the parameter indicating the best rank approximation. In this way we have a projection $\mathbb{Z}$ that depends on both $\mathbf{g}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X}')$, treating the two data matrices symmetrically in the total least squares framework of Eq. (3-9). The subspace of dimension $r \leq n$ is computed in a data-driven way through POD. In particular, the parameter r is chosen by looking for spectral gaps in the POD analysis (see Section 2.4 of [17]). The computation of r is fundamental in case the problem in Eq. (3-9) is underconstrained, i.e. $n > m$. As a matter of fact, the use of a subspace $\mathbb{Z}_r$ can make Eq. (3-9) overconstrained again such that $r < m < n$.

What results from the projection of the data matrices in $\mathbb{Z}_r$ is the following

$$\mathbf{g}(\mathbb{X}')\mathcal{P}_{\mathbb{Z}_r} = \mathbf{K}\mathbf{g}(\mathbb{X})\mathcal{P}_{\mathbb{Z}_r} \tag{3-11}$$

where $\mathcal{P}_{\mathbb{Z}_r}$ is the projection operator into the range of $\mathbb{Z}_r$. This projection operator can be computed as $\mathcal{P}_{\mathbb{Z}_r} = QQ^T$ where Q has the first r right-singular vectors of $\mathbb{Z}$ as columns. In order to solve Eq. (3-11), one can simply compute $\mathbf{K} = \mathbf{g}(\mathbb{X}')\mathcal{P}_{\mathbb{Z}_r}(\mathbf{g}(\mathbb{X})\mathcal{P}_{\mathbb{Z}_r})^\dagger$ or observe the steps of the next algorithm to determine the spectral properties of the approximated Koopman operator.

In what follows we introduce the algorithm of tDMD in [17]. We will make use of the MATLAB notation that is: having a matrix A then $A(1:i, 1:j)$ is a matrix that has the first i rows and j columns of A.

```
Algorithm 2: tDMD algorithm
Input: $\mathbf{g}(\mathbb{X})$, $\mathbf{g}(\mathbb{X}')$
Output: tDMD modes $\Phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_r \end{bmatrix}$
1 Construct the augmented snapshot matrix $\mathbb{Z} = [\mathbf{g}(\mathbb{X})^T \ \mathbf{g}(\mathbb{X}')^T]^T$;
2 Compute an SVD for $\mathbb{Z} = \tilde{U}\tilde{\Sigma}\tilde{V}^H$ and set $Q = \tilde{V}(:, 1:r)$;
3 Compute the projections of the snapshot matrices as $\hat{\mathbf{g}}(\mathbb{X}) = \mathbf{g}(\mathbb{X})Q$, $\hat{\mathbf{g}}(\mathbb{X}') = \mathbf{g}(\mathbb{X}')Q$;
4 Compute the SVD of $\hat{\mathbf{g}}(\mathbb{X}) = U\Sigma V^T$;
5 Compute the matrix $\hat{\mathbf{K}} = U^T\hat{\mathbf{g}}(\mathbb{X}')V\Sigma^{-1}$ from which is possible to have the DMD
   eigenvalues $\lambda_i$ and modes $w_i$ as $\hat{\mathbf{K}}w_i = \lambda_i w_i$.
```

If we want to restrict the tDMD to solely the states snapshot matrices, we simply set $\mathbf{g}(\mathbb{X}) = \mathbb{X}$ and $\mathbf{g}(\mathbb{X}') = \mathbb{X}'$.

3-2 Extended Dynamic Mode Decomposition (EDMD)

The extended dynamic mode decomposition (EDMD) was first designed by Williams et al. in [1] to specifically approximate Koopman operators. The connection between the Koopman operator framework and EDMD was already established in the work of Rowley et al. [28, Section 3.2]. The EDMD can be seen as a generalization of the DMD algorithm. As a matter of fact, the DMD computes regression with direct measurements of the states as basis, i.e. $g(x) = x$ with $x \in \mathbb{R}^l$, while EDMD computes it on a library of observables $\mathbf{g}$ chosen by the user. These chosen observables are generally nonlinear functions of the states [25]. It has to be pointed out that the choice and number of functions in the EDMD is fundamental to compute the approximated Koopman operator intrinsic to the system under study. As stated in [1, Section 2.2] the bigger the library of observables, the higher the chances to well approximate the Koopman eigenfunctions ψ. Unfortunately, the increase in the number of functions in $\mathbf{g}$, increases also the complexity of the algorithm.

The procedure of the overall algorithm is similar to DMD, the only main difference is that the snapshots are made through the observables chosen by the user. As we introduced for tDMD in Section 3-1-1, the observables snapshot matrices are defined as

$$\mathbf{g}(\mathbb{X}) = \begin{bmatrix} | & | & & | \\ \mathbf{g}(x[0]) & \mathbf{g}(x[1]) & \dots & \mathbf{g}(x[m-1]) \\ | & | & & | \end{bmatrix}, \quad \mathbf{g}(\mathbb{X}') = \begin{bmatrix} | & | & & | \\ \mathbf{g}(x[1]) & \mathbf{g}(x[2]) & \dots & \mathbf{g}(x[m]) \\ | & | & & | \end{bmatrix},$$

where

$$\mathbf{g}(x[k]) = \begin{bmatrix} g_1(x[k]) \\ g_2(x[k]) \\ \vdots \\ g_n(x[k]) \end{bmatrix}, \tag{3-12}$$

is the library of observables, with $g_j : \mathbb{R}^l \to \mathbb{R}$, $j = 1, 2, \dots, n$. Taking into account that $\mathbf{g}$ generally does not span a Koopman-invariant subspace [1, Section 2.2.1], the Koopman framework can be rewritten as

$$\mathbf{K}g(x[k]) = \mathbf{K}\mathbf{g}(x[k]) + e[k],$$

where, again, $g(x[k]) : \mathbb{R}^l \to \mathbb{R}$, $g \in \mathscr{A}_{\mathscr{H}}$ with $\mathscr{A}_{\mathscr{H}}$ being a Koopman-invariant subspace of the Hilbert space $\mathscr{H}$, $\mathbf{K} \in \mathbb{R}^{n\times n}$ and $e[k] \in \mathscr{H}$ is the residual term. As with the DMD, linear regression can be used in order to compute $\mathbf{K}$ using the snapshots $\mathbf{g}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X}')$:

$$\min_{\mathbf{K}} \|\mathbf{g}(\mathbb{X}') - \mathbf{K}\mathbf{g}(\mathbb{X})\|_F. \tag{3-13}$$

The nomenclature and passages are taken from [25] and Chapter 10 of [33]. The solution for Eq. (3-13) is

$$\mathbf{K} = \mathbf{g}(\mathbb{X}')\mathbf{g}(\mathbb{X})^\dagger, \tag{3-14}$$

where † is the Moore-Penrose pseudoinverse. The computation of Eq. (3-14) let us approximate the Koopman operator with a finite dimensional matrix $\mathbf{K}$. Now, the approximated triplet of the Koopman mode decomposition, i.e. (λ_j, ψ_j, v_j), can be computed. The approximated eigenfunctions ψ_j are computed through the left eigenvectors $\xi_j^T \in \mathbb{R}^{1\times n}$ of $\mathbf{K}$, i.e. $\xi_j^T\mathbf{K} = \lambda_j\xi_j^T$, such that

$$\psi_j(x[k]) \approx \xi_j^T\mathbf{g}(x[k]), \tag{3-15}$$

where ξ_j is related to the eigenvalue λ_j (see Section 2-2). The Koopman modes are then computed by

$$\varphi_j = \mathbf{b}^T w_j,$$

where $w_j \in \mathbb{R}^{n\times 1}$ is the right eigenvector of $\mathbf{K}$ and $\mathbf{b}$ is a vector of weights. The parameter $\mathbf{b} \in \mathbb{R}^{n\times l}$ is defined such that

$$\mathbf{g}^x(x[k]) = \mathbf{b}^T\mathbf{g}(x[k]),$$

where $\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_l \end{bmatrix}$ and

$$\mathbf{g}^x(x) = \begin{bmatrix} g_1^x(x[k]) \\ g_2^x(x[k]) \\ \vdots \\ g_l^x(x[k]) \end{bmatrix} = \begin{bmatrix} e_1^T x[k] \\ e_2^T x[k] \\ \vdots \\ e_l^T x[k] \end{bmatrix},$$

where e_j is the j-th unit vector, and $e_j^T x[k] = x_j[k]$ is a full j-th state observable (see Eq. 15 in Williams et al [1]). In other words, the matrix $\mathbf{b}$ contains the weights needed to retrieve the states of the systems from the chosen library of observables. If the measurements of the full state observables and library of observables are available, then $\mathbf{b}^T = \mathbf{g}^x(x[k])\mathbf{g}(x[k])^\dagger$.

The following algorithm summarizes the EDMD procedure:

Algorithm 3: EDMD algorithm

```
Input: $\mathbb{X}, \mathbb{X}'$
Output: EDMD modes $\Phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_r \end{bmatrix}$
1 Define a library of observables $\mathbf{g}(x)$ as basis;
2 Construct the snapshot matrices $\mathbf{g}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X}')$;
3 Compute the approximated Koopman operator: $\mathbf{K} = \mathbf{g}(\mathbb{X}')\mathbf{g}(\mathbb{X})^\dagger$;
4 Compute the Koopman eigenfunctions: $\psi_j(x[k]) \approx \xi_j^T\mathbf{g}(x[k]), \forall k \in \mathbb{N}_0$, where
    $\xi_j^T\mathbf{K} = \lambda_j\xi_j^T$;
5 Compute the modes: $\varphi_j = \mathbf{b}^T w_j$ where $\mathbf{K}w_j = \lambda_j w_j$ for $j = 1, \dots, r$, and
    $\mathbf{g}^x(x[k]) = \mathbf{b}^T\mathbf{g}(x[k])$.
```

The choice of the library of observables $\mathbf{g}$ should take into account the properties of the system. However, generally, the properties of the system are unknown. To solve this issue, there are a few standard sets of functions dictionaries that can be used. Some examples of these predefined functions are monomials, Hermite polynomials, Fourier basis, Legendre polynomials, and radial basis functions. The problem of choosing a suitable dictionary for specific problems is still an important issue. In this regard, many techniques have been developed, a few of which are from the machine learning field. A summary of these solutions can be found in Section 3.3.2 of Parmar et al. [37].

3-3 Noise-Free Extended Dynamic Mode Decomposition (NFEDMD)

In Chapter 1 we mentioned that this thesis will evaluate and compare state-of-the-art algorithms in the presence of noise. To have a reference during the benchmarking experiments of Chapter 4 we decided to implement also a noise-free version of the EDMD (NFEDMD). Let us have the following noisy observable measurement:

$$\mathbf{g}_\gamma[k] = \mathbf{g}(x[k]) + \gamma[k], \quad \mathbf{g}(x[k]), \gamma[k] \in \mathbb{R}^n, \tag{3-16}$$

where γ is additive noise. In terms of snapshot matrices, this becomes

$$\mathbf{G}_\gamma = \mathbf{g}(\mathbb{X}) + \Gamma, \; \mathbf{G}'_\gamma = \mathbf{g}(\mathbb{X}') + \Gamma',$$

where

$$\Gamma = \begin{bmatrix} | & | & & | \\ \gamma[0] & \gamma[1] & \dots & \gamma[m-1] \\ | & | & & | \end{bmatrix}, \Gamma' = \begin{bmatrix} | & | & & | \\ \gamma[1] & \gamma[2] & \dots & \gamma[m] \\ | & | & & | \end{bmatrix}$$

In this case, the normal EDMD would use snapshots of $\mathbf{G}_\gamma$. This means that the training data used to compute the matrix approximation of the Koopman operator will be noisy, leading to

$$\mathbf{K} = \mathbf{G}'_\gamma \mathbf{G}_\gamma^\dagger.$$

This can result into wrong approximations of the Koopman operator and degrade the performance of the model. NFEDMD, instead, has knowledge of the noise-free observables snapshot matrices $\mathbf{g}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X}')$ making it possible to compute $\mathbf{K}$ as follows:

$$\mathbf{K} = \mathbf{g}(\mathbb{X}')\mathbf{g}(\mathbb{X})^\dagger.$$

We will use NFEDMD to compare the other algorithms, which will compute $\mathbf{K}$ with noisy observables, against a noise-free version of EDMD. This will give us insights on how robust the other numerical methods are against noise.

3-4 Subspace Dynamic Mode Decomposition (subDMD)

Designed by Takeishi et al. [12], subspace dynamic mode decomposition (subDMD) exploits the advantages of dynamic mode decomposition and subspace identification to model random dynamical systems in the presence of noise in the observables.

Let us consider the following random dynamical system

$$x[k+1] = T(x[k], v[k]),$$

where $x[k] \in \mathbb{R}^l$ are the states and $v[k] \in V$ is the process noise. The process noise lies in a probability space defined as (V, S_V, μ_V), where V is the sample space, i.e. the set of all possible outcomes, S_V is the event space, i.e. the set of the outcomes in V we want to consider, and $\mu_V : S_V \to [0,1]$ is the probability function, i.e. a function assigning a probability to each event in S_V. In addition, $v[k]$ is independent from the states and is independent and identically distributed (i.i.d.). If these criteria on the process noise are met, then we are able to apply just DMD or EDMD in order to compute the approximated Koopman operator (see Assumption 2 in [12]).

Now let us add measurement noise to the observables as follows

$$\mathbf{g}_\alpha(x[k], \alpha[k]) = \mathbf{g}(x[k]) + \alpha[k],$$

where $\mathbf{g}_\alpha : \mathbb{R}^l \times W \to \mathbb{R}^n$ and $\alpha : W \to \mathbb{R}$ is a random variable lying in the probability space (W, S_W, μ_W). Additionally, let us assume that α is white noise, i.e. a noise signal that has zero mean, finite variance and is temporally uncorrelated [38]. The observables measurement noise drastically degrades the performances of DMD or EDMD algorithms in computing $\mathbf{K}$. This is showed in the experiments discussed in Chapter 4.

In order to define the necessary assumptions to compute $\mathbf{K}$, we introduce two matrices: R and G. Matrix R is defined as the covariance matrix between the measurement noise α and the linearization error e in the Koopman operator framework, i.e. $\mathbf{g}(x[k+1]) = \mathbf{K}\mathbf{g}(x[k]) + e[k]$. In other words:

$$\mathbb{E}_{V,W}[e[i]\alpha[j]^T] = R\delta(i-j),$$

where

$$\delta(i-j) = \begin{cases} 1, & \text{for } i-j=0 \\ 0, & \text{for } i-j \neq 0. \end{cases}$$

Matrix G, instead, is defined as

$$G = \mathbb{E}_{\mathbb{R}^l}[G_{k,k}],$$

where $G_{k,k} = \mathbb{E}_\Omega[\mathbf{g}(x[k])\mathbf{g}(x[k])^T]$ and $G_{i,j} = \mathbb{E}_\Omega[\mathbf{g}(x[i])\mathbf{g}(x[j])^T]$.

Now that the nomenclature is fully defined, we can introduce the concept of subspace DMD. Let us define a matrix of lifted snapshots

$$\mathbf{g}_\alpha(\mathbb{X}_k) = \begin{bmatrix} | & | & & | \\ \mathbf{g}_\alpha(x[k]) & \mathbf{g}_\alpha(x[k+1]) & \dots & \mathbf{g}_\alpha(x[k+m-1]) \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n\times m},$$

from which we construct two matrices

$$\mathbf{g}_p = \begin{bmatrix} \mathbf{g}_\alpha(\mathbb{X}_0) \\ \mathbf{g}_\alpha(\mathbb{X}_1) \end{bmatrix}, \quad \mathbf{g}_f = \begin{bmatrix} \mathbf{g}_\alpha(\mathbb{X}_2) \\ \mathbf{g}_\alpha(\mathbb{X}_3) \end{bmatrix}.$$

If $\text{rank}(\mathbf{g}_p) = 2n$ then we are able to construct a matrix $O = \mathbf{g}_f\mathbf{g}_p^T(\mathbf{g}_p\mathbf{g}_p^T)^{-1}\mathbf{g}_p$ (see proof of Theorem 1 in [12][4]). In the next step, the compact SVD of O is computed

$$O = U_r\Sigma_r V_r^T,$$

where $r = \text{rank}(O)$ is the truncation parameter of the matrices, i.e. $U_r \in \mathbb{R}^{2n\times r}$, $\Sigma_r \in \mathbb{R}^{r\times r}$ and $V_r \in \mathbb{R}^{m\times r}$. Based on Theorem 1 of [12], if $\text{rank}(\mathbf{g}_p) = 2n$ and $\text{rank}(\mathbf{K}G + R) = n$, then for $m \to \infty$ we have that

$$U_{r1}U_{r2}^\dagger \to \mathbf{K}, \quad \text{where } U = \begin{bmatrix} U_{r2} \\ U_{r1} \end{bmatrix},$$

with $U_{r1}, U_{r2} \in \mathbb{R}^{2n\times n}$. This means that, with enough samples data of $\mathbf{g}_\alpha$, we can compute the approximated Koopman operator $\mathbf{K}$ arbitrarily well, even with the presence of measurement noise in the observables.

The entire subDMD algorithm can be then summarized as follows. As for tDMD, we will make use of the MATLAB notation.

```
Algorithm 4: subDMD algorithm
Input: $\mathbb{X}_0 = \begin{bmatrix} | & | & & | \\ x[0] & x[1] & \dots & x[m-1] \\ | & | & & | \end{bmatrix}$, $\mathbb{X}_1 = \begin{bmatrix} | & | & & | \\ x[1] & x[2] & \dots & x[m] \\ | & | & & | \end{bmatrix}$,
       $\mathbb{X}_2 = \begin{bmatrix} | & | & & | \\ x[2] & x[3] & \dots & x[m+1] \\ | & | & & | \end{bmatrix}$, $\mathbb{X}_3 = \begin{bmatrix} | & | & & | \\ x[3] & x[4] & \dots & x[m+2] \\ | & | & & | \end{bmatrix}$.
Output: subDMD modes $\Phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_r \end{bmatrix}$
1 Define a library of observables $\mathbf{g}(x)$ as basis;
2 Construct the snapshot matrices $\mathbf{g}_p$ and $\mathbf{g}_f$;
3 Compute the matrix $O = \mathbf{g}_f\mathbf{g}_p^T(\mathbf{g}_p\mathbf{g}_p^T)^{-1}\mathbf{g}_p$;
4 Compute the compact SVD of $O = U_r\Sigma_r V_r^T$ and take
   $U_{r1} = U(1:n,:), U_{r2} = U(n+1:2n,:)$;
5 Compute the compact SVD of $U_{r1} = U\Sigma V^T$ and define the following matrix
   $A = U^T U_{r2} V\Sigma$;
6 Compute the eigenvalues $\Lambda$ and eigenvectors $w$ of $A$ from which we derive the dynamic
   modes $\Phi = \Lambda^{-1}U_{r2}V\Sigma^{-1}w$
```

For the sake of this work, we will use this algorithm only to compute a matrix approximation of the Koopman operator $\mathbf{K} \approx U_{r1}U_{r2}^\dagger$ which follows from Theorem 1 of [12].

3-5 Koopman Subspace Identification (KSI)

The first implementation of a subspace identification algorithm to compute $\mathbf{K}$ was done by Lian et al. [13]. More in detail, Lian et al. used a subspace identification algorithm named sequential PARSIM algorithm (see Algorithm 2 in Qin et al. [39]) to solve a least squares

[4]As defined in [12], matrix O can be seen as the orthogonal projection of the rows of $\mathbf{g}_f$ into the row space of $\mathbf{g}_p$.

problem in the observables space and compute $\mathbf{K}$. In this section, we will introduce the Koopman subspace identification (KSI), an algorithm based on [13], that uses the subspace identification technique described in the book of Verhaegen et al. [14, Section 9.2.2]. It has to be noted that Lian et al. in [13] did not introduce any noise in its implementation. Consequently, for this section, we will not consider noise in the design of KSI.

To apply subspace identification techniques in the space of observables $\mathscr{A}_{\mathscr{H}}$, we assume that the dynamics in it can be described through the following linear predictor:

$$\begin{aligned} y[k+1] &= Ay[k] \\ \mathbf{g}(x[k]) &= Cy[k], \end{aligned} \tag{3-17}$$

where $\mathbf{g}(x[k]) \in \mathbb{R}^n$ is the library of observables chosen as basis for the Koopman-invariant subspace $\mathscr{A}_{\mathscr{H}}$ and $y[k] \in \mathbb{R}^{n_y}$ is what we will define *state observables.* All the measurements from Eq. (3-17) are then gathered in the following *data equation* (see section 9.1 of [14])

$$H(\mathbf{x}) = \mathcal{O}Y, \tag{3-18}$$

where $\mathbf{x} = \begin{bmatrix} x[0] & x[1] & \dots & x[m-1] \end{bmatrix}$ is a single trajectory in the state space and $H(\mathbf{x})$ is a Hankel matrix of the observables measurements, i.e.

$$H(\mathbf{x}) = \begin{bmatrix} \mathbf{g}(x[0]) & \mathbf{g}(x[1]) & \dots & \mathbf{g}(x[m-1]) \\ \mathbf{g}(x[1]) & \mathbf{g}(x[2]) & \dots & \mathbf{g}(x[m]) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}(x[s-1]) & \mathbf{g}(x[s]) & \dots & \mathbf{g}(x[m+s-1]) \end{bmatrix}. \tag{3-19}$$

In Eq. (3-19), the parameter s is the number of sets of n rows and m is the number columns, i.e. the samples of $\mathbf{g}$. Matrix $\mathcal{O}$ is the observability matrix

$$\mathcal{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{s-1} \end{bmatrix}, \tag{3-20}$$

and Y is defined as

$$Y = \begin{bmatrix} y[0] & y[1] & \dots & y[m-1] \end{bmatrix}.$$

It has to be taken into account that the above data equation matrices $H(\mathbf{x}), \mathcal{O}$ and Y are suitable only if one trajectory in the space of observables is measured.

We will now present a generalization of the framework in Eq. (3-18) for multiple trajectories. To do so we follow the procedure introduced by Duchesne et al. [40], where subspace identification is adapted to work with multiple data sets. Let us have N_t trajectories picked in the state space. Each i-th trajectory will be defined as

$$\mathbf{x}^{(i)} = \begin{bmatrix} x^{(i)}[0] & x^{(i)}[1] & \dots & x^{(i)}[m-1] \end{bmatrix}, \quad \text{where } i = 1, \dots, N_t.$$

The Hankel matrix for multiple trajectories is defined as

$$\mathbb{H}(\mathbb{X}) = \begin{bmatrix} H(\mathbf{x}^{(1)}) & H(\mathbf{x}^{(2)}) & \dots & H(\mathbf{x}^{(N_t)}) \end{bmatrix}, \tag{3-21}$$

where

$$H(\mathbf{x}^{(i)}) = \begin{bmatrix} \mathbf{g}(x^{(i)}[0]) & \mathbf{g}(x^{(i)}[1]) & \dots & \mathbf{g}(x^{(i)}[m-1]) \\ \mathbf{g}(x^{(i)}[1]) & \mathbf{g}(x^{(i)}[2]) & \dots & \mathbf{g}(x^{(i)}[m]) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}(x^{(i)}[s-1]) & \mathbf{g}(x^{(i)}[s]) & \dots & \mathbf{g}(x^{(i)}[m+s-1]) \end{bmatrix}.$$

The observability matrix will remain the same as in Eq. (3-20), while the matrix of state observables measurements will be defined as follows

$$\mathbb{Y} = \begin{bmatrix} y^{(1)}[0] & \dots & y^{(1)}[m-1] & y^{(2)}[0] & \dots & y^{(N_t)}[m-1] \end{bmatrix}.$$

where $y^{(i)}[k]$ is the k-th sample of the i-th trajectory in the space of state observables. The data equation in Eq. (3-18) then becomes

$$\mathbb{H}(\mathbb{X}) = \mathcal{O}\mathbb{Y}. \tag{3-22}$$

Now that the framework is introduced, we can move to the procedure of identifying $\mathbf{K}$ through subspace identification. Looking at Eq. (3-22), we can see that the column space of $\mathbb{H}(\mathbb{X})$ is contained in the column space of the observability matrix [14, Section 9.2.2]. If the two column spaces are equal, we are then able to compute the linear predictor matrices in Eq. (3-17) from $\mathbb{H}(\mathbb{X})$. In order to have the two ranges equal, we first choose the number of state observables n_y, columns s and samples m such that $n_y < s \ll m$. Assuming that $\mathbb{Y}$ is full rank, and that the realization of the predictor in Eq. (3-17) is minimal, then we can affirm that $\text{range}(\mathbb{H}(\mathbb{X})) = \text{range}(\mathcal{O})$.

Lemma 3-5.1. Let $\mathbb{H}(\mathbb{X}) \in \mathbb{R}^{(n\cdot s)\times\beta}$ be a Hankel matrix of measurements $\mathbf{g}(x[k]) \in \mathbb{R}^n$ from the linear predictor

$$\begin{aligned} y[k+1] &= Ay[k] \\ \mathbf{g}(x[k]) &= Cy[k]. \end{aligned}$$

Let also $\mathcal{O} \in \mathbb{R}^{(n\cdot s)\times n_y}$ and $\mathbb{Y} \in \mathbb{R}^{n_y\times\beta}$ be the observability matrix of the predictor, and the matrix containing all the measurements of $y[k] \in \mathbb{R}^{n_y}$. If the following points are true

(i) $n_y < s \ll m$;

(ii) $\mathbb{Y}$ is full rank;

(iii) The realization of the predictor (A, C) is minimal[5];

then we have that $\text{range}(\mathbb{H}(\mathbb{X})) = \text{range}(\mathcal{O})$.

In what follows we will give a proof of what just stated in Lemma 3-5.1 (see also Section 9.2.2 of [14]).

[5]Meaning that (A, C) in Eq. (3-17) is controllable and observable [41, Theorem 2.2].

Proof. Being $\mathbb{Y}$ a full rank matrix of rank n_y and assuming that the system described Eq. (3-17) is minimally realized, we can state that $\text{rank}(\mathcal{O}) = n_y$. Knowing that $\mathbb{H}(\mathbb{X}) = \mathcal{O}\mathbb{Y}$ from Eq. (3-22), we can apply Sylvester's Inequality[6]

$$\text{rank}(\mathcal{O}) + \text{rank}(\mathbb{Y}) - n_y \leq \underbrace{\text{rank}(\mathcal{O}\mathbb{Y})}_{\text{rank}(\mathbb{H}(\mathbb{X}))} \leq \min(\text{rank}(\mathcal{O}), \text{rank}(\mathbb{Y})).$$

This results in

$$n_y + n_y - n_y \leq \underbrace{\text{rank}(\mathcal{O}\mathbb{Y})}_{\text{rank}(\mathbb{H}(\mathbb{X}))} \leq \min(n_y, n_y),$$

from which we can conclude that $\text{rank}(\mathbb{H}(\mathbb{X})) = n_y$ leading to $\text{range}(\mathbb{H}(\mathbb{X})) = \text{range}(\mathcal{O})$. □

If the two column spaces are the same, then we are able to obtain the observability matrix $\mathcal{O}$ up to a similarity transformation by computing the SVD of $\mathbb{H}(\mathbb{X})$. In other words, from the measurements of the library of observables $\mathbf{g}(x)$, we can compute the predictor matrices A and C, from Eq. (3-17), up to a similarity transformation (A_T, C_T). By computing the SVD of $\mathbb{H}(\mathbb{X})$ we get the following

$$\mathbb{H}(\mathbb{X}) = U\Sigma V^T,$$

from which we have

$$U_{n_y} = \mathcal{O}T = \begin{bmatrix} C_T \\ C_T A_T \\ \vdots \\ C_T A_T^{s-1} \end{bmatrix},$$

where U_{n_y} is the matrix resulting by taking the first n_y columns of U, and T is the similarity transformation. From U_{n_y} we can find C_T by taking the first n rows, i.e. using MATLAB notation $C_T = U_{n_y}(1:n,:)$. Matrix A_T can be found by solving the following overdetermined equation

$$U_{n_y}(1:(s-1)n,:)A_T = U_{n_y}(n+1:sn,:) \;\Rightarrow\; A_T = U_{n_y}(1:(s-1)n,:)^\dagger U_{n_y}(n+1:sn,:).$$

Now that the predictor matrices are available, it is possible to compute the matrix approximation of the Koopman operator

$$\mathbf{K} = C_T A_T C_T^\dagger.$$

The Koopman subspace identification can be summarized with the following algorithm

Algorithm 5: KSI algorithm

```
Input: $\mathbb{X}$
Output: Matrix approximation $\mathbf{K}$ of the Koopman operator
1 Choose $\mathbf{g}(x) \in \mathbb{R}^n$ and construct $\mathbb{H}(\mathbb{X})$ ;
2 Compute the SVD for $\mathbb{H}(\mathbb{X}) = U\Sigma V^T$ and take the first $n_y$ columns of $U$ resulting in $U_{n_y}$;
3 Assuming $\text{range}(\mathbb{H}(\mathbb{X})) = \text{range}(\mathcal{O})$ then compute
    $C_T = U_{n_y}(1:n,:)$, $A_T = U_{n_y}(1:(s-1)n,:)^\dagger U_{n_y}(n+1:sn,:)$;
4 Compute $\mathbf{K} = C_T A_T C_T^\dagger$.
```

[6] The Sylvester's Inequality (given in Lemma 2.1 in [14]), states that having two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ then

$$\text{rank}(A) + \text{rank}(B) - n \leq \text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B)).$$

Chapter 4

Evaluation of the Numerical Methods

The algorithms defined in Chapter 3 have not yet been compared to one another. There are cases like subDMD where its performances are presented against tDMD in [12]. Nevertheless, the remaining numerical methods listed in Chapter 3 are not considered. Additionally, KSI has not been evaluated in the presence of noise yet[1]. Therefore, to the extent of our knowledge, the first comprehensive comparison of these algorithms in the presence of noisy training data is given here. Moreover, the first performance results with noise in the measurements of the states are shown and discussed.

In this chapter, we will discuss the design and results of the experiments carried out to test the algorithms mentioned in Chapter 3. These experiments were designed to evaluate and compare the numerical methods for changing values of different parameters. More in detail we will show the results for an increasing number of training trajectories and an increasing standard deviation of additive noise in the observables and the measurement of the states.

4-1 Design of the Experiments

The design of the experiments to benchmark the algorithms described in Chapter 3 relies on three main steps:

1. Collecting and gathering the measurements of the system trajectories;
2. Computing the matrix approximation $\mathbf{K}$ of the Koopman operator;
3. Validating $\mathbf{K}$ with respect to the noise-free trajectories of the underlying nonlinear system.

For small amounts of data, the performances of matrix $\mathbf{K}$ can drastically change depending on the position of the randomly picked initial conditions of the training and validation data.

[1]In [13] the validation of their algorithm does not include the presence of noise in the measurements.

To get a clear picture of matrix $\mathbf{K}$ performance, we decided to implement the experiments with the Monte Carlo method (see Raychaudhuri et al. [42]). More in detail, the methodology is based on repeating steps 1, 2, and 3 for c times. Each time with different sets of initial conditions. This is because the collection of states measurements in step 1 comes from N_t trajectories, each of which has a randomly picked initial condition. To test the algorithms we chose a low number of initial conditions N_t to analyze their behaviour in this critical condition. The low amount of N_t will not cover the entire state space domain under observation. For this reason, the performance of $\mathbf{K}$ can considerably vary. A more in deep discussion regarding the Monte Carlo method is in Section 4-1-2.

In the following subsections, steps 1, 2, and 3 will be described. For the sake of simplicity in analyzing the trajectories of the underlying system, we decided to adopt bi-dimensional nonlinear systems in the experiments, i.e. $x \in \mathbb{R}^2$. Additionally, we will make use of the nomenclature below:

- x: state of the system governed by $\dot{x} = f(x)$ with $x \in \mathbb{R}^2$
- n: number of observables used as library of functions, i.e. $\mathbf{g}(x) = \begin{bmatrix} g_1(x) & \dots & g_n(x) \end{bmatrix}^T$, where $g_i : \mathbb{R}^2 \to \mathbb{R},\ i = 1, \dots, n$
- m_t: number of samples per trajectory for *training*
- N_t: number of trajectories for *training*
- m_v: number of samples per trajectory for *validation*
- N_v: number of trajectories for *validation*
- VAF: mean of the variance accounted for (marked line in the plots)
- VAFσ: standard deviation of the variance accounted for (filled area in the plots)
- RMSE: mean of the root mean square error
- RMSEσ: standard deviation of the root mean square error

4-1-1 Collecting and Gathering Measurement Data

Let us consider a general bi-dimensional system in continuous time such as Eq. (2-1):

$$\dot{x}(t) = f(x),\ x \in \mathbb{R}^2. \tag{4-1}$$

To gather measurements of the states, the system is solved through the Runge-Kutta method (RK4) [43]. This method relies on discretizing the continuous time system and solving it for the initial condition $x(0)$. More in detail, knowing the function f and the initial condition $x(0)$ we can write

$$x[k+1] = x[k] + \frac{1}{6}\Delta t(\kappa_1 + 2\kappa_2 + 2\kappa_3 + \kappa_4),\ \text{with}\, k = 0, \dots, m_t$$

where

$$\begin{aligned}\kappa_1 &= f(x[k]),\\ \kappa_2 &= f(x[k] + \Delta t\frac{\kappa_1}{2}),\\ \kappa_3 &= f(x[k] + \Delta t\frac{\kappa_2}{2}),\\ \kappa_4 &= f(x[k] + \Delta t\kappa_3),\end{aligned}$$

with $\Delta t = 0.01\,\mathrm{s}$. The method RK4 is then used to compute the trajectory evolution of N_t initial conditions. These initial conditions are confined in $X[0] \in \mathbb{U}^{2\times N_t}$ where $\mathbb{U} = \{x \in \mathbb{R} \mid -1 \leq x \leq 1\}$. In this work, we uniformly randomly picked the initial conditions from $\mathbb{U}$. The choice of $\mathbb{U}$ was made to limit the computational power needed for computing the observables[2]. For each k-th time step, the samples of each trajectory are placed in *snapshot matrices* $X[k] \in \mathbb{R}^{2\times N_t}$. When all the m_t snapshot matrices are computed with RK4 they are then gathered in

$$\mathbb{X} = \begin{bmatrix} X[0] & X[1] & \dots & X[m_t] \end{bmatrix}. \tag{4-2}$$

When all the state samples are measured, the data is lifted. In this process, matrix $\mathbb{X}$ is given as input to a library of functions $\mathbf{g} : \mathbb{R}^2 \to \mathbb{R}^n$. In this thesis, for the construction of $\mathbf{g}$ we chose to use a basis of monomials of the form $x_1^i x_2^j$ with $0 \leq i+j \leq d$ where d is the monomial degree. For a chosen parameter d, and having $x \in \mathbb{R}^2$, the number of observables n in $\mathbf{g}$ is equal to

$$n = \sum_{j=1}^{d+1} j.$$

The choice of the monomial basis was made to conform with Parmar et al.[3] [37, Table 1] and to easily retrieve the estimated states (see Section 4-1-2). The lifting step is used to bring the measurements of the dynamics of the states to the higher-order space of observables. If the choice of the library of observables forms a basis that spans a Koopman-invariant subspace, then we can correctly compute the matrix approximation $\mathbf{K}$ of the Koopman operator (see Section 2-3).

After the lifting step, the lifted state measurements are gathered into two types of matrices, depending on the used algorithm. For EDMD, NFEDMD, and tDMD, the data is gathered in matrices of the form

$$\begin{aligned}\mathbf{g}(\mathbb{X}) &= \begin{bmatrix} \mathbf{g}(X[0]) & \mathbf{g}(X[1]) & \dots & \mathbf{g}(X[m_t-1]) \end{bmatrix},\\ \mathbf{g}(\mathbb{X}') &= \begin{bmatrix} \mathbf{g}(X[1]) & \mathbf{g}(X[2]) & \dots & \mathbf{g}(X[m_t]) \end{bmatrix}.\end{aligned}$$

For subspace-identification-based algorithms (subDMD and KSI), we adopted the technique of Duchesne et al. [40] in which multiple trajectories are considered. To do this we construct

[2]The library of observables chosen is of monomial type (see below). If the measured dynamics of the system are higher than 1, then, for a high enough monomial degree, the computed values of the library can become quite impractical.

[3]As we will see in Section 4-2 the nonlinear systems under test are both polynomial based. Following [37, Table 1], a monomial basis is a good choice for the library of observables.

each Hankel matrix with different single trajectories in the state space. In other words, let us consider the i-th state trajectory

$$\mathbf{x}^{(i)} = \begin{bmatrix} x^{(i)}[0] & x^{(i)}[1] & \dots & x^{(i)}[m_t-1] \end{bmatrix}, \quad \text{where } i = 1, \dots, N_t.$$

Here $x^{(i)}[0]$ is a single initial condition in the state space. After passing $\mathbf{x}^{(i)}$ through the vector of observables $\mathbf{g}$, the lifted data is placed in the following Hankel matrix

$$H(\mathbf{x}^{(i)}) = \begin{bmatrix} \mathbf{g}(x^{(i)}[0]) & \mathbf{g}(x^{(i)}[1]) & \dots & \mathbf{g}(x^{(i)}[m_t-s-1]) \\ \mathbf{g}(x^{(i)}[1]) & \mathbf{g}(x^{(i)}[2]) & \dots & \mathbf{g}(x^{(i)}[m_t-s]) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{g}(x^{(i)}[s-1]) & \mathbf{g}(x^{(i)}[s]) & \dots & \mathbf{g}(x^{(i)}[m_t-1]) \end{bmatrix}. \tag{4-3}$$

This step is repeated for all the N_t trajectories. At the end of this process, all the Hankel matrices are stored in

$$\mathbb{H}(\mathbb{X}) = \begin{bmatrix} H(\mathbf{x}^{(1)}) & H(\mathbf{x}^{(2)}) & \dots & H(\mathbf{x}^{(N_t)}) \end{bmatrix}.$$

In this thesis, according to Takens' embedded theorem, we chose the number of row elements in Eq. (4-3) to be $s = 2n + 1$. Before describing the theorem, let us define what a *delay coordinate embedding* is. Section 2.3 of Kamb et al. [44] explains that the delay embedding of the entire trajectory of $\mathbf{g}(\mathbf{x}^{(i)})$ is defined as a new observable $\mathbf{g}_\Delta(\mathbf{x}^{(i)}) = (\mathbf{g}(x^{(i)}[0]),\ \mathbf{g}(x^{(i)}[1]),\ \dots,\ \mathbf{g}(x^{(i)}[s-1]))$. It is possible to infer that this is equal to the first column of the Hankel matrix in Eq. (4-3). The Takens' embedded theorem shows that it is possible to reconstruct the original state dynamics of the underlying nonlinear system through delay coordinate embedding $\mathbf{g}_\Delta(\mathbf{x}^{(i)})$ of the states if $s \geq 2n + 1$ (see [44, Section 2.3]).

Once all the training data is correctly gathered inside the matrices $\mathbf{g}(\mathbb{X})$ and $\mathbb{H}(\mathbb{X})$, we can apply the algorithms to compute $\mathbf{K}$ from them. An overview scheme of the training data generation is shown in Fig. 4-1.

Introducing Noise in the Measurements

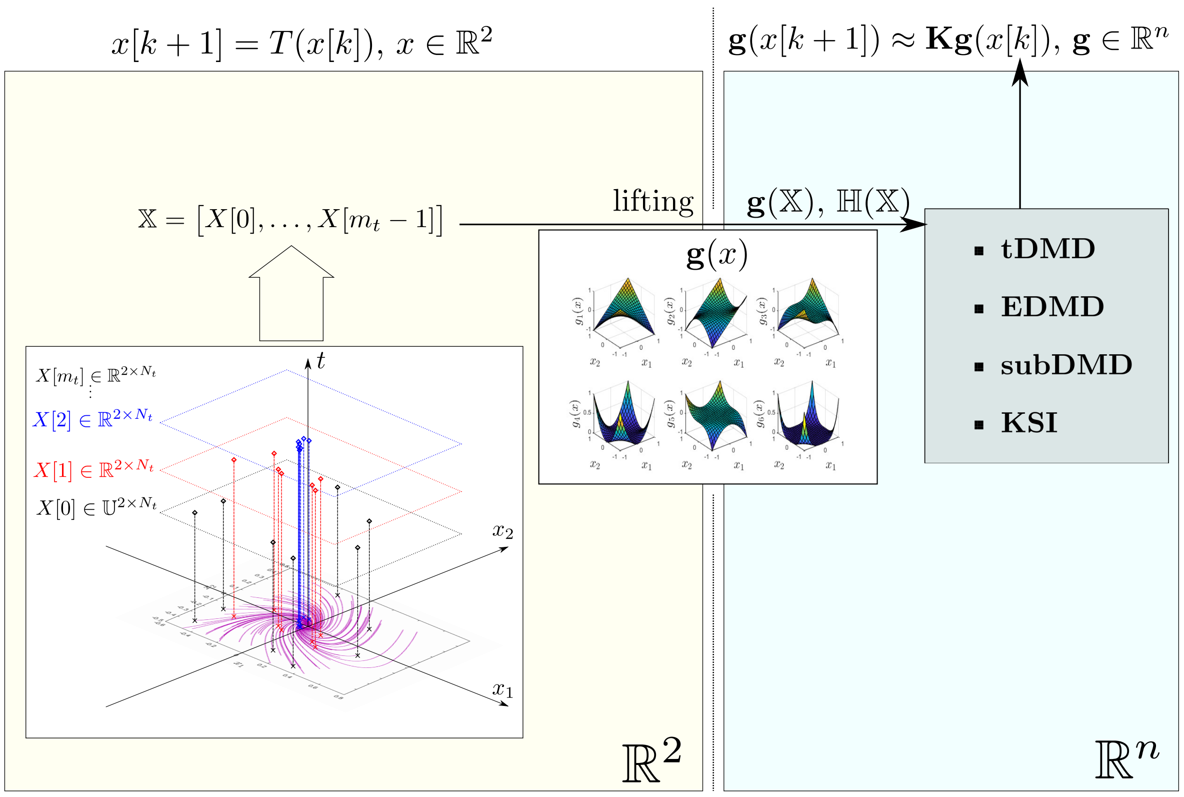
One of the main objectives of this thesis is to test the chosen algorithms in presence of noise in the measurements of both observables and states. In what follows we describe two types of noisy measurements for the training of the numerical methods.

Let us have the following observable

$$g_i^\omega[k] = g_i(x[k]) + \omega[k],\ i = 1, \dots, n, \tag{4-4}$$

where $\omega[k]$ is additive white Gaussian noise. More in detail we refer to $\omega[k]$ as an i.i.d. random variable with zero-mean normal distribution, i.e. $\omega[k] \sim \mathcal{N}(0, \sigma_\omega^2)$. The noise ω in Eq. (4-4) is defined as *observable measurement noise*. In case ω is introduced, we have that the basis of observables $\mathbf{g}$ is defined as $\mathbf{g}(x[k]) = \begin{bmatrix} g_1^\omega(x[k]) & \dots & g_n^\omega(x[k]) \end{bmatrix}^T$ which becomes

$$\mathbf{g}(x[k]) = \begin{bmatrix} g_1(x[k]) + \omega[k] \\ \vdots \\ g_n(x[k]) + \omega[k] \end{bmatrix}.$$



$x[k+1] = T(x[k]),\ x \in \mathbb{R}^2$
$\mathbf{g}(x[k+1]) \approx \mathbf{K}\mathbf{g}(x[k]),\ \mathbf{g} \in \mathbb{R}^n$
$\mathbb{X} = [X[0], \dots, X[m_t - 1]]$
lifting
$\mathbf{g}(\mathbb{X}),\ \mathbb{H}(\mathbb{X})$
$\mathbf{g}(x)$
tDMD
EDMD
subDMD
KSI
$X[m_t] \in \mathbb{R}^{2\times N_t}$
$X[2] \in \mathbb{R}^{2\times N_t}$
$X[1] \in \mathbb{R}^{2\times N_t}$
$X[0] \in \mathbb{U}^{2\times N_t}$
t
x_2
x_1
$\mathbb{R}^2$
$\mathbb{R}^n$

Figure 4-1: Scheme of the measurement, lifting and gathering of the training data

The second measurement noise that we consider acts directly on the states such that

$$g_i^v[k] = g_i(x[k] + v[k]),\ i = 1, \dots n, \tag{4-5}$$

where $v[k]$ is additive white Gaussian noise with $v[k] \sim \mathcal{N}(0, \sigma_v^2)$. Noise $v[k]$ is referred to as *state measurement noise*. Let us introduce v in the state measurements. For the i-th state trajectory we have

$$\mathbf{x}^{(i)} = \Big[x^{(i)}[0] + v^{(i)}[0],\ x^{(i)}[1] + v^{(i)}[1],\ \dots$$
$$\dots\ x^{(i)}[m_t - 1] + v^{(i)}[m_t - 1]\Big], \quad \text{where } i = 1, \dots, N_t.$$

In terms of snapshot matrices, we have that Eq. (4-2) becomes

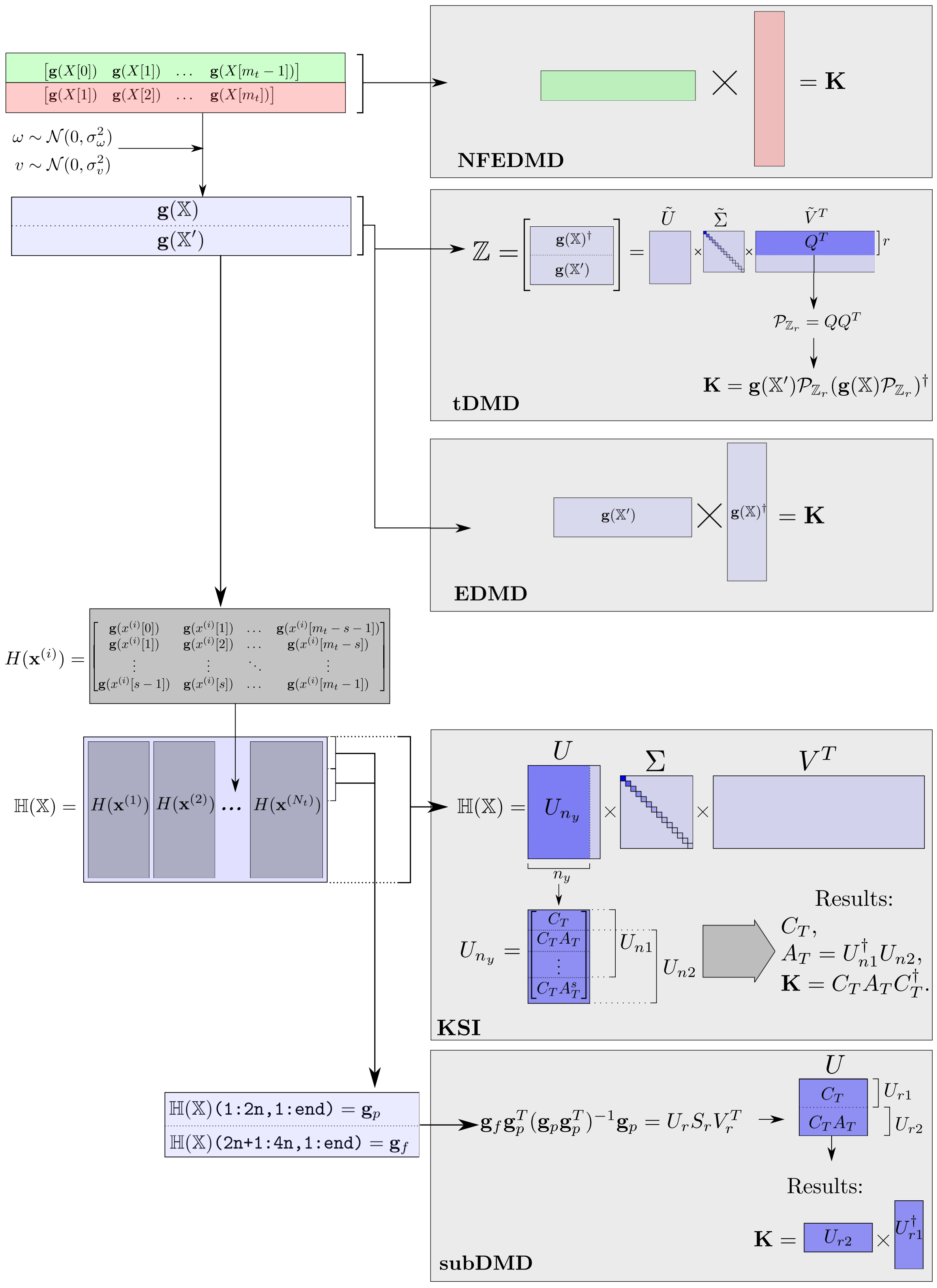
$$\mathbb{X} = \Big[X[0] + V[0],\quad X[1] + V[1],\quad \dots\quad X[m_t] + V[m_t]\Big],$$

where

$$V[k] = \Big[v^{(1)}[k],\quad v^{(2)}[k],\quad \dots\quad v^{(N_t)}[k]\Big],\ k = 1, \dots m_t.$$

The introduction of v then influences the library of observables which becomes $\mathbf{g}(x[k]) = \Big[g_1^v(x[k]) \quad \dots \quad g_n^v(x[k])\Big]^T$. The values of state measurement noise will be nonlinearly changed by the observables. We expect that this deformation will drastically deteriorate the performance of the algorithms.

The above-defined measurement noises will be introduced in the training data for the numerical methods, and the results will be discussed in Section 4-2.



$[\mathbf{g}(X[0]) \ \mathbf{g}(X[1]) \ \dots \ \mathbf{g}(X[m_t-1])]$
$[\mathbf{g}(X[1]) \ \mathbf{g}(X[2]) \ \dots \ \mathbf{g}(X[m_t])]$
$\omega \sim \mathcal{N}(0,\sigma_\omega^2)$
$v \sim \mathcal{N}(0,\sigma_v^2)$
$= \mathbf{K}$
NFEDMD
$\mathbf{g}(\mathbb{X})$
$\mathbf{g}(\mathbb{X}')$
$\mathbb{Z} =$
$\mathbf{g}(\mathbb{X})^\dagger$
$\tilde{U}$ $\tilde{\Sigma}$ $\tilde{V}^T$ Q^T r
$\mathcal{P}_{\mathbb{Z}_r} = QQ^T$
$\mathbf{K} = \mathbf{g}(\mathbb{X}')\mathcal{P}_{\mathbb{Z}_r}(\mathbf{g}(\mathbb{X})\mathcal{P}_{\mathbb{Z}_r})^\dagger$
tDMD
EDMD
$H(\mathbf{x}^{(i)}) =$
$\mathbf{g}(x^{(i)}[0])$ $\mathbf{g}(x^{(i)}[1])$ $\dots$ $\mathbf{g}(x^{(i)}[m_t-s-1])$
$\mathbf{g}(x^{(i)}[1])$ $\mathbf{g}(x^{(i)}[2])$ $\dots$ $\mathbf{g}(x^{(i)}[m_t-s])$
$\mathbf{g}(x^{(i)}[s-1])$ $\mathbf{g}(x^{(i)}[s])$ $\dots$ $\mathbf{g}(x^{(i)}[m_t-1])$
$\mathbb{H}(\mathbb{X}) =$
$H(\mathbf{x}^{(1)})$ $H(\mathbf{x}^{(2)})$ $\dots$ $H(\mathbf{x}^{(N_t)})$
U Σ V^T U_{n_y} n_y
$U_{n_y} =$
C_T $C_T A_T$ $C_T A_T^s$
U_{n1} U_{n2}
Results:
C_T,
$A_T = U_{n1}^\dagger U_{n2}$,
$\mathbf{K} = C_T A_T C_T^\dagger$.
KSI
$\mathbb{H}(\mathbb{X})$(1:2n,1:end) $= \mathbf{g}_p$
$\mathbb{H}(\mathbb{X})$(2n+1:4n,1:end) $= \mathbf{g}_f$
$\mathbf{g}_f\mathbf{g}_p^T(\mathbf{g}_p\mathbf{g}_p^T)^{-1}\mathbf{g}_p = U_r S_r V_r^T$
U_{r1} U_{r2}
Results:
$\mathbf{K} =$ U_{r2} $\times$ $U_{r1}^\dagger$
subDMD

Figure 4-2: Summary scheme of the implemented numerical methods of Chapter 3

4-1-2 Validation

To benchmark the algorithms, we decided to validate their performances on replicating the noise-free trajectories of the underlying nonlinear system. In other words, the benchmark result is based on the error between the model and the nonlinear system trajectories. With this in mind, the algorithms of Chapter 3 are implemented to determine only the approximated Koopman operator $\mathbf{K} \in \mathbb{R}^{n\times n}$, disregarding the computation of eigenvalues and modes. Matrix $\mathbf{K}$ is then used as a linear operator to compute the next step in the space of observables such that

$$\begin{aligned} z[k+1] &= \mathbf{K}z[k] \\ z[0] &= \mathbf{g}(x[0]), \end{aligned} \tag{4-6}$$

with $\mathbf{g}(x[0]) \in \mathbb{R}^n$ and $z[k] \in \mathbb{R}^n$. The objective is to compute a $\mathbf{K}$ such that $z[k]$ is equal to $\mathbf{g}(x[k])$. A summary scheme of the implemented algorithms in Chapter 3 is showed in Fig. 4-2.

Once the implemented algorithms of Chapter 3 compute their respective approximated Koopman operator $\mathbf{K}$, a validation step is carried out. The validation is based on first computing m_v samples of the trajectories of the underlying nonlinear system for each of the N_v randomly picked initial conditions in $X_v[0] \in \mathbb{U}^{2\times N_v}$. When all the N_v states trajectories are determined, $X_v[0]$ is lifted, resulting in $\mathbf{g}(X_v[0])$. We then use $\mathbf{g}(X_v[0])$ as initial condition $z[0]$ in Eq. (4-6) and solve it. If the computed approximated Koopman operator is satisfactory, then $z[k] \approx \mathbf{g}(x[k])$. This is done for each numerical method. When all the m_v trajectory samples are computed, the estimated states $\hat{x} \in \mathbb{R}^2$ are retrieved from z. Since we chose a monomial basis, there exists a matrix B such that $x[k] = B\mathbf{g}(x[k])$. Hence, the estimated state can be easily recovered from Eq. (4-6) with $\hat{x}[k] = Bz[k]$. More in detail, the monomial basis z of degree d will be as follows (see Section 4-1-1)

$$z[k] = \begin{bmatrix} 1 \\ \hat{x}_1[k] \\ \hat{x}_1^2[k] \\ \vdots \\ \hat{x}^d[k] \\ \hat{x}_2[k] \\ \vdots \\ \hat{x}_1^i[k]\hat{x}_2^j[k] \end{bmatrix}, \; k = 0, \dots m_v, \tag{4-7}$$

with $0 \le i + j \le d$. From Eq. (4-7) it is possible to recover $\hat{x}_1$ and $\hat{x}_2$ by taking the second element and the $(2+d)$-th element of $z[k]$ respectively. In other words, by using the notation of MATLAB we have

$$\begin{aligned} \hat{x}_1[k] &= [z[k]](2, :), \\ \hat{x}_2[k] &= [z[k]](2+d, :), \; k = 0, \dots m_v. \end{aligned}$$

The modeled dynamics of $\hat{x}$ are then validated with respect to the noise-free x of the underlying nonlinear system. This is done by using the root mean square error (RMSE) and the variance accounted for (VAF) between the two trajectories. The RMSE is proportional to the error between the real and the estimated state trajectory. The VAF computes, on a percentage scale, how much the dispersion of the error is with respect to the variance of the real trajectory. We chose to use both RMSE and VAF for two main reasons. The first reason is that VAF does not count for biases in the model estimated trajectory while RMSE does. The second is because VAF gives results on a scale from 0 to 100. This is particularly useful to have an overall qualitative measure of the correctness of a model, making it easier to compare the algorithms. The RMSE and VAF for a single trajectory are defined as follows

$$\mathrm{RMSE}_p = \sqrt{\frac{\sum_{k=0}^{m_v-1}(x_i[k]-\hat{x}_i[k])^2}{m_v}}, \tag{4-8}$$

$$\mathrm{VAF}_p = \left(1 - \frac{\mathrm{var}(x_i - \hat{x}_i)}{\mathrm{var}(x_i)}\right)100\%, \quad i = 1,2;\ p = 1,\dots N_v \tag{4-9}$$

where $\mathrm{var}(x_i)$ is the variance of the trajectory x_i such that

$$\mathrm{var}(x_i) = \frac{1}{m_v}\sum_{k=0}^{m_v-1}(x_i[k]-\mu)^2$$

with μ as the mean of x_i (see [14, Page 383]). We recall that the entire process of validation is repeated for N_v trajectories. The overall RMSE_j and VAF_j, of the j-th Monte Carlo simulation, result from the mean of all the N_v VAF_p and RMSE_p computed with Eq. (4-8) and Eq. (4-9) such that

$$\mathrm{RMSE}_j = \frac{1}{N_v}\sum_{q=1}^{N_v}\mathrm{RMSE}_q,$$

$$\mathrm{VAF}_j = \frac{1}{N_v}\sum_{q=1}^{N_v}\mathrm{VAF}_q.$$

Additionally, the standard deviation of both VAF_p and RMSE_p, with $p = 1,\dots N_v$, is computed as follows

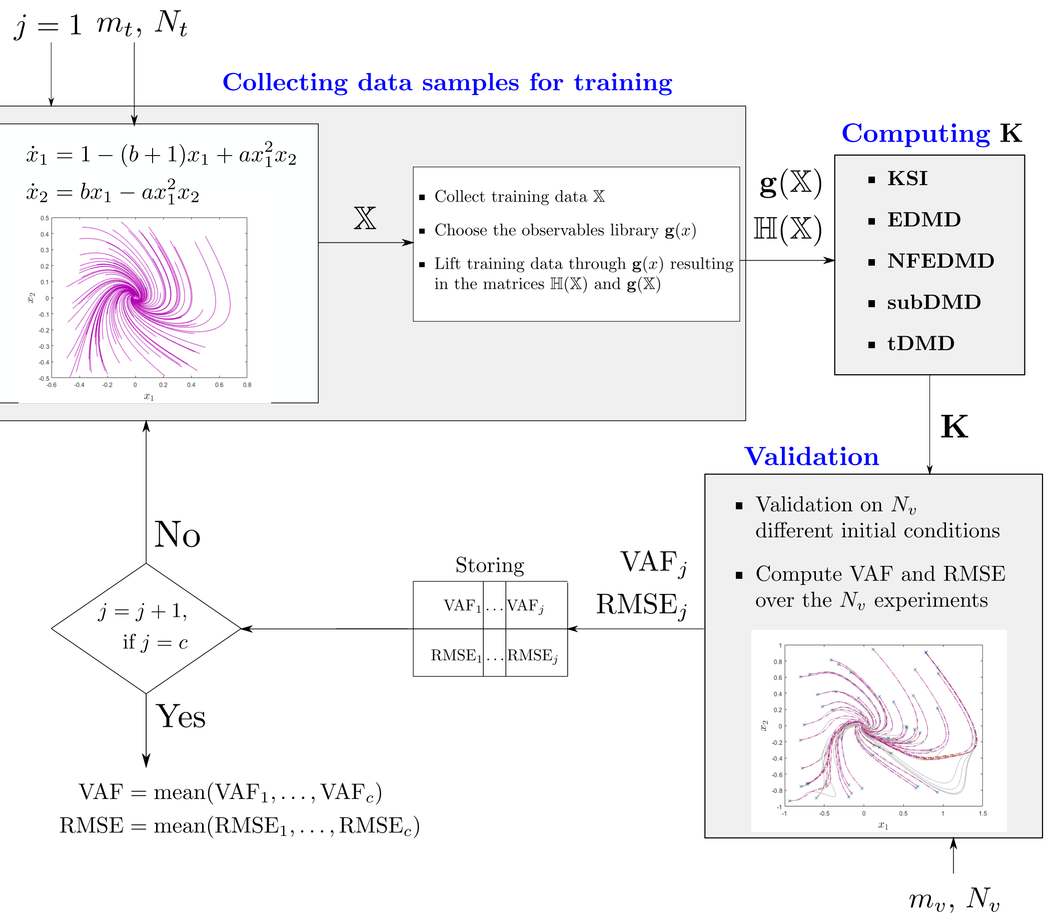
$$\mathrm{RMSE}\sigma_j = \sqrt{\frac{\sum_{q=1}^{N_v}(\mathrm{RMSE}_q - \mu_{\mathrm{RMSE}})}{N_v}},$$

$$\mathrm{VAF}\sigma_j = \sqrt{\frac{\sum_{q=1}^{N_v}(\mathrm{VAF}_q - \mu_{\mathrm{VAF}})}{N_v}}.$$

where μ_{RMSE} and μ_{VAF} are the mean of the RMSE_p and VAF_p ($\forall p = 1,\dots N_v$) respectively. This process will be repeated c times, i.e. the chosen number of Monte Carlo experiments.

The number of initial conditions N_t used for training the numerical methods might not be enough to cover the entire state subspace under analysis[4]. In real case scenarios, it is generally the case to have a high, medium, and low amount of training data. For low amounts, the performance of the algorithms can strongly fluctuate with randomly chosen initial conditions. An example can be that the initial conditions $X_t[0] \in \mathbb{U}^{2\times N_v}$ are positioned in a section of

[4]We remind the reader that this section of the state space is a square of dimension 2 and centered at the origin of the state space (see Section 4-1-1).



$j = 1$ m_t, N_t
Collecting data samples for training
$\dot{x}_1 = 1 - (b+1)x_1 + ax_1^2 x_2$
$\dot{x}_2 = bx_1 - ax_1^2 x_2$
$\mathbb{X}$
Collect training data $\mathbb{X}$
Choose the observables library $\mathbf{g}(x)$
Lift training data through $\mathbf{g}(x)$ resulting in the matrices $\mathbb{H}(\mathbb{X})$ and $\mathbf{g}(\mathbb{X})$
$\mathbf{g}(\mathbb{X})$
$\mathbb{H}(\mathbb{X})$
Computing $\mathbf{K}$
KSI
EDMD
NFEDMD
subDMD
tDMD
$\mathbf{K}$
Validation
Validation on N_v different initial conditions
Compute VAF and RMSE over the N_v experiments
VAF_j
RMSE_j
Storing
$\text{VAF}_1 \dots \text{VAF}_j$
$\text{RMSE}_1 \dots \text{RMSE}_j$
$j = j + 1$, if $j = c$
No
Yes
$\text{VAF} = \text{mean}(\text{VAF}_1, \dots, \text{VAF}_c)$
$\text{RMSE} = \text{mean}(\text{RMSE}_1, \dots, \text{RMSE}_c)$
m_v, N_v

Figure 4-3: Summary scheme of the benchmark framework

the state space that has a certain behaviour. When validating, the picked initial conditions in $X_v[0] \in \mathbb{U}^{2 \times N_v}$ are, instead, on another section with completely different behaviour. In these cases, the numerical algorithms can have a drastic deterioration of their performances leading to unsatisfactory values of VAF and RMSE. In summary, given the two sets of random initial conditions $X_v[0]$ and $X_t[0]$, the models' performances can drastically vary depending on them. With this in mind, we chose to adopt the Monte Carlo method by averaging the performance results over several experiments.

To cover as many resulting outputs as possible, the Monte Carlo method is adopted [42]. Its implementation is based on simply repeating, with different sets of initial conditions, the steps described in Section 4-1-1 and Section 4-1-2 for c times. The parameter c is chosen by the user. Having a set of fixed parameters m_t, N_t, m_v and N_v, a matrix containing all VAF_j and RMSE_j, $j = 1, \dots c$, is computed. When the Monte Carlo experiment is concluded the final VAF and RMSE, along with their standard deviations, are given by

$$\begin{aligned} \text{RMSE} &= \frac{1}{c}\sum_{j=1}^{c} \text{RMSE}_j, \quad \text{RMSE}\sigma = \frac{1}{c}\sum_{j=1}^{c} \text{RMSE}\sigma_j, \\ \text{VAF} &= \frac{1}{c}\sum_{j=1}^{c} \text{VAF}_j, \quad \text{VAF}\sigma = \frac{1}{c}\sum_{j=1}^{c} \text{VAF}\sigma_j. \end{aligned} \tag{4-10}$$

The values in Eq. (4-10) are considered the final performance values of the numerical algorithms for the chosen parameters m_t, N_t, m_v, and N_v.

A summary scheme of the benchmark procedure is given in Fig. 4-3

4-2 Benchmark Results of the Numerical Methods in Chapter 3

This section discusses the results of the benchmark experiments. For each subsection, the performances of the numerical algorithms for different bi-dimensional nonlinear systems will be shown. The benchmark results are presented as trajectories of the VAF and RMSE for an increasing parameter. For the VAF, both the mean (dark line) and the standard deviation (light area) are shown in the same plot, while the RMSE mean and its standard deviation (RMSE σ) are shown in different plots. This choice was made because there are cases in which RMSE σ is too high, making it difficult to present it in a plot like the filled one of the VAF.

4-2-1 Brusselator

In this subsection we will discuss the performances of the various numerical algorithms applied to the Brusselator, a nonlinear system described with

$$\begin{aligned} \dot{x}_1 &= 1 - (b-1)x_1 + ax_1^2 x_2, \\ \dot{x}_2 &= bx_1 - ax_1^2 x_2, \end{aligned} \tag{4-11}$$

with $x_1, x_2 \in \mathbb{U}$, $a = 1$, and $b = 0.7$. After centering the stable equilibrium point at the origin of the state space, the dynamics of Eq. (4-11) behave like in Fig. 4-4. We chose this system so to analyze the behaviour of the numerical algorithms in the presence of a stable equilibrium point.

Changing the Number of Training Trajectories

This experiment is designed to analyze the behaviour of the algorithms for an increasing number of training trajectories N_t. In other words, when the validation of the algorithms is finished for a certain value of N_t, the latter is increased and a new *cycle of evaluation* starts. The term *cycle of evaluation* refers to the process described in Section 4-1-1 and Section 4-1-2. This procedure is done while keeping all the other parameters fixed. In this way, we can analyze the dynamics of the VAF and RMSE with respect to N_t. The results showed in this subsection were computed with the fixed parameters in Table 4-1. The results of the first experiment are shown in Fig. 4-5, where no noise was introduced in neither the observables nor the states measurements. It is possible to see from Fig. 4-5 that the performances of the algorithms in replicating the trajectories of the Brusselator are quite the same apart from KSI. The KSI method has worse VAF and RMSE with respect to the other algorithms. The biggest difference can be seen with a low number of training trajectories ($N_t \leq 40$). Nevertheless, the performances between the various numerical methods are very similar after $N_t = 40$. Looking at the VAF variance of KSI in Fig. 4-5 it is possible to see that it is higher than the other

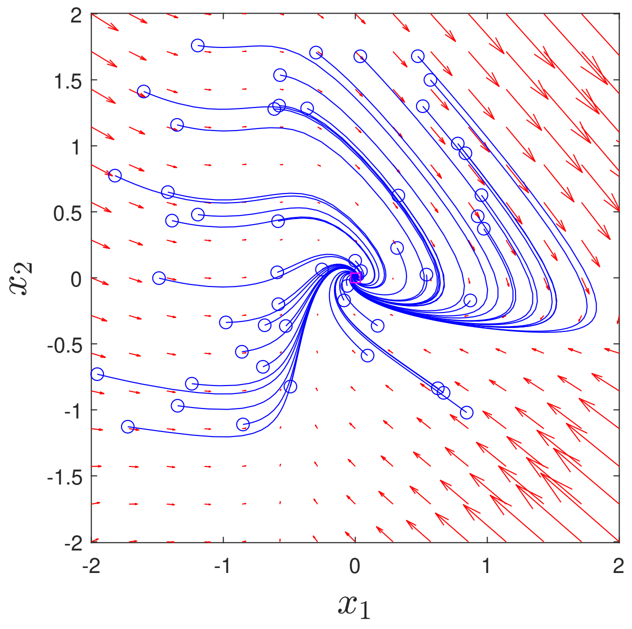


Figure 4-4: Phase portrait of the Brusselator

Parameter	Value
Number of observables used in the library (n)	15
Number of samples per training trajectory (m_t)	300
Number of validation trajectories (N_v)	10
Number of samples per validation trajectory (m_v)	1000
Number of Monte Carlo cycles (c)	50

Table 4-1: Fixed parameters for the experiment with changing N_t

algorithms even for $N_t \geq 40$. A possible explanation is that the KSI model overfits the given training data sets due to its high VAF variance (see [45, Page 32]).

Now that the case with no noise is covered, we introduce the observables measurement noise ω (see Eq. (4-4)) with $\sigma_\omega = 0.1$. For this experiment, we simply add the white Gaussian noise ω to the observables $\mathbf{g}$. The result of this addition is showed in Fig. 4-6. The difference with the previous case in Fig. 4-5 is quite significant. The predicted state trajectories of the algorithms have a considerable deterioration in presence of ω. However, two numerical methods still give very promising results: KSI and NFEDMD. Even with a low number of trajectories ($N_t < 60$) NFEDMD is the best performing algorithm. This is due to its availability of noise-free observables. This lets NFEDMD create a satisfactory model even with a small amount of training data. For $N_t \geq 60$ also KSI reaches a satisfactory prediction quality similar to NFEDMD. This is because KSI only uses the noisy observables measurements as training data. This makes it unaware of the noise ω. To overcome this drawback KSI needs more data than NFEDMD to compute an acceptable model. The similar performance of KSI and NFEDMD for $N_t \geq 60$ means that we can reach a satisfactory level of robustness even without noise-free measurements of the observables.

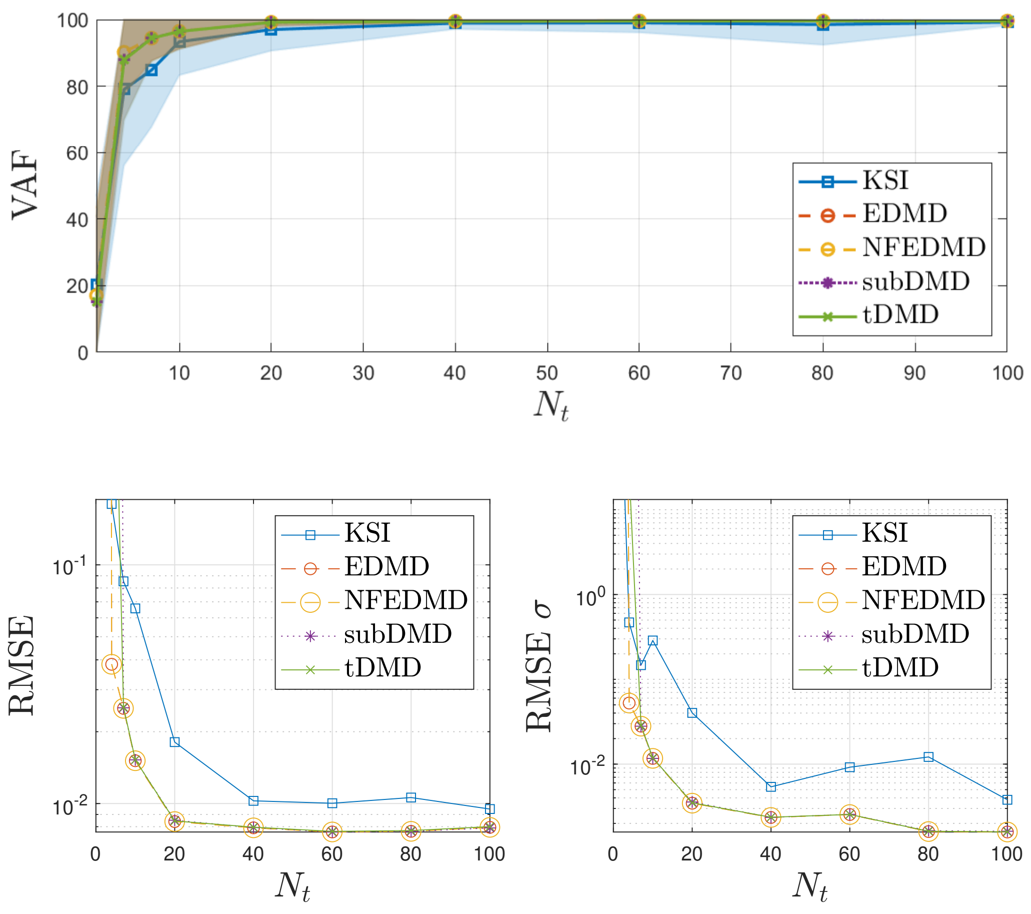


Figure 4-5: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, no noise is applied to the measurements, i.e. $\omega[k] = 0$ and $v[k] = 0 \ \forall k = 0, \dots, m_t$. The following parameters were set: $n = 15$, $m_t = 300$, $N_v = 10$, $m_v = 1000$ and $c = 50$

The RMSE dynamics of subDMD and tDMD are not shown in Fig. 4-6 as their error magnitude is too high[5] with respect to the other algorithms. These two algorithms might need a larger training data set to be comparable to the KSI and NFEDMD. The low amount of training data can considerably influence the quality of the algorithms in the presence of noise.

Let us now add noise v directly in the measurement of the states (see Eq. (4-5)) with $\sigma_v = 0.1$. In other words, we add v to the snapshot matrices $\mathbb{X}$. This means that the additive white Gaussian noise is introduced before lifting the data, a process that will distort it. The result of this addition is shown in Fig. 4-7. It is possible to see that the distortion of v drastically deteriorates the performances of all the algorithms apart from NFEDMD.

Similarly for Fig. 4-6, in the RMSE plots of Fig. 4-7 the subDMD and tDMD are not present because their error values are too high to be displayed. This can be due to the high amplitude of the measurement noise. We remind the reader that the state space under consideration is $x \in \mathbb{U}^2$ and the standard deviation of the noise is $\sigma_v = 0.1$. This strong presence of noise in the

[5]The steady value of RMSE is around 10^5

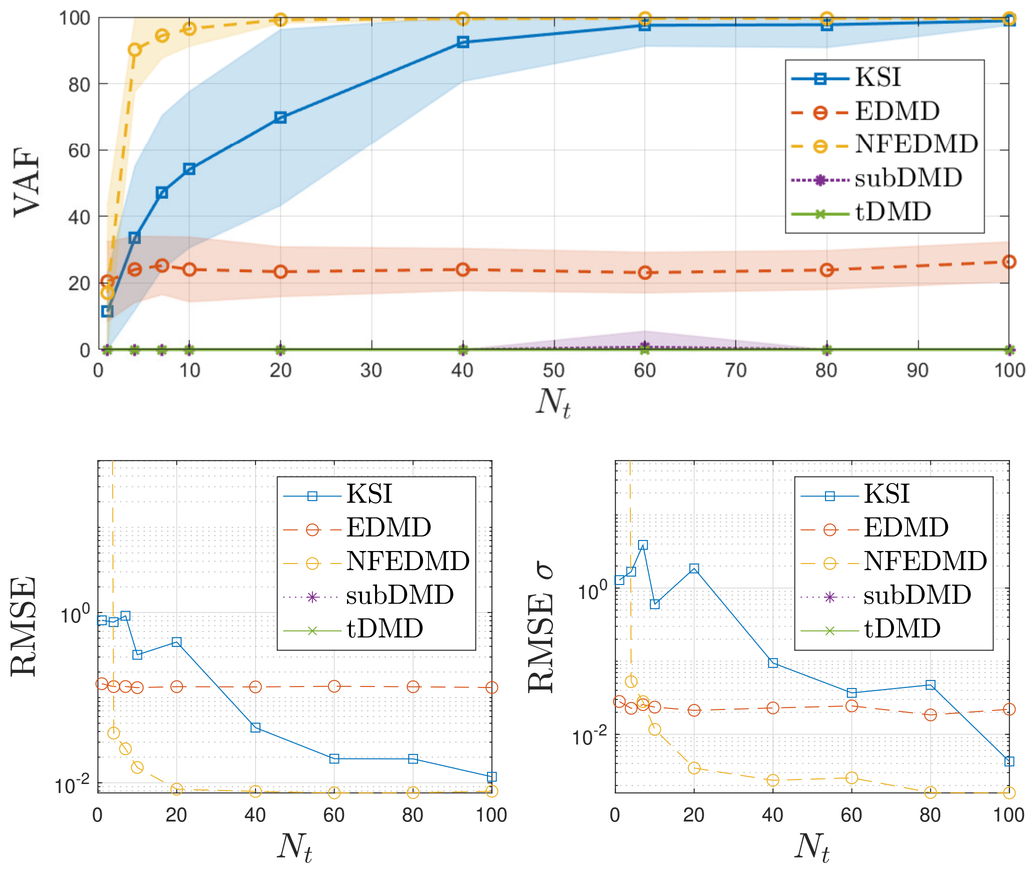


Figure 4-6: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the observables measurement noise was introduced with $\sigma_\omega = 0.1$. The following parameters were set: $n = 15$, $m_t = 300$, $N_v = 10$, $m_v = 1000$ and $c = 50$

training data can result in unsatisfactory models. Additionally, during the implementation of these two algorithms, we chose an order $r = n$. In both subDMD and tDMD the parameter r is used to truncate the order of the SVD step (see Section 3-1-1 and Section 3-4). The value of r can greatly influence the resulting model. The choice of setting $r = n$ is done to comply with the implementation of KSI that does not have any order truncation.

Changing the Standard Deviation of the Measurement Noises

With this experiment, we want to confirm the results above and see how robust the algorithms are for different values of the standard deviation of v and ω. The fixed parameters for this test category are listed in Table 4-2.

We start this second type of experiment by increasing the standard deviation σ_ω of the observable measurement noise ω every time the evaluation cycle finishes. Looking at the plot of Fig. 4-8 it is possible to see that KSI performance is very similar to NFEDMD up to $\sigma_\omega = 0.07$. After $\sigma_\omega = 0.07$, the deterioration of the KSI model becomes quite relevant. Nevertheless, these robustness results make KSI a promising technique capable to reach the level of NFEDMD.

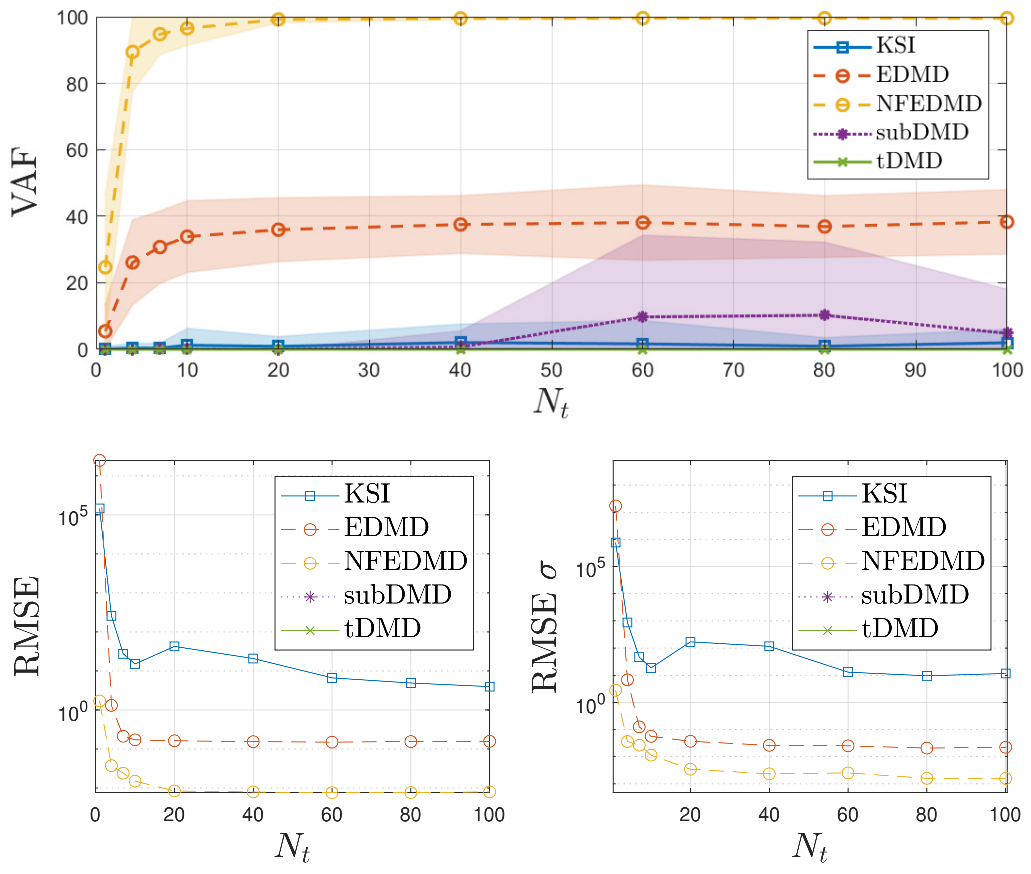


Figure 4-7: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the states measurement noise was introduced with $\sigma_v = 0.1$. The following parameters were set: $n = 15$, $m_t = 300$, $N_v = 10$, $m_v = 1000$ and $c = 50$

Now let us analyze the behaviour of the models in case of an increasing standard deviation σ_v of the state measurement noise v. The results of this experiment are given in the plots of Fig. 4-9. The NFEDMD shows again excellent performances, followed by the EDMD and subDMD. For what regards the other techniques, the introduction of v drastically deteriorates their ability to replicate the noise-free trajectories. As we assumed for the previous experiment category, this is due to the distortion of v from the lifting process. To conclude the observations, the VAF plot of Fig. 4-9 shows that the subDMD performance should be better than KSI, while its RMSE is worse. The better performance of VAF with respect to RMSE might be due to a bias in the computation of the model of subDMD introduced by the distorted noise.

Parameter	Value
Number of observables used in the library (n)	15
Number of samples per training trajectory (m_t)	300
Number of training trajectories (N_t)	50
Number of validation trajectories (N_v)	10
Number of samples per validation trajectory (m_v)	1000
Number of Monte Carlo cycles (c)	50

Table 4-2: Fixed parameters for the experiment with a changing standard deviation of the measurement noises

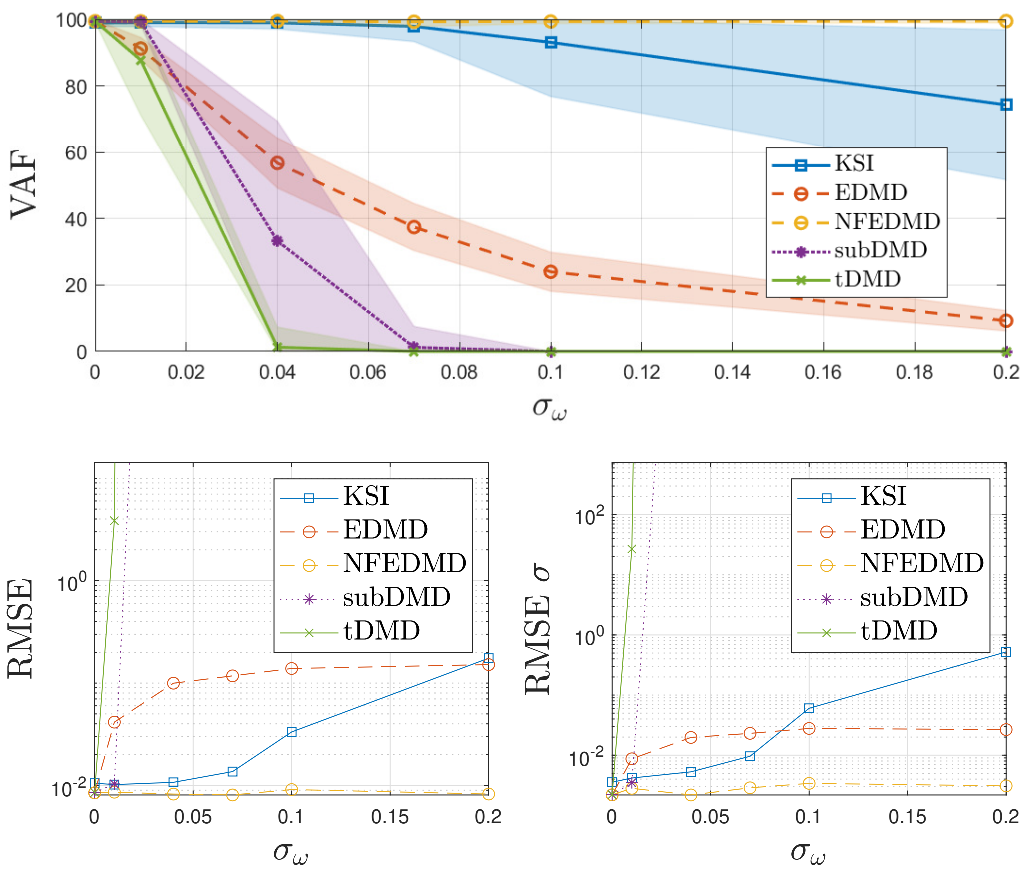


Figure 4-8: Dynamics of the VAF and RMSE for different values of observables measurement noise standard deviation σ_ω. The following parameters were set: $n = 15$, $m_t = 300$, $N_t = 50$, $N_v = 10$, $m_v = 1000$ and $c = 50$

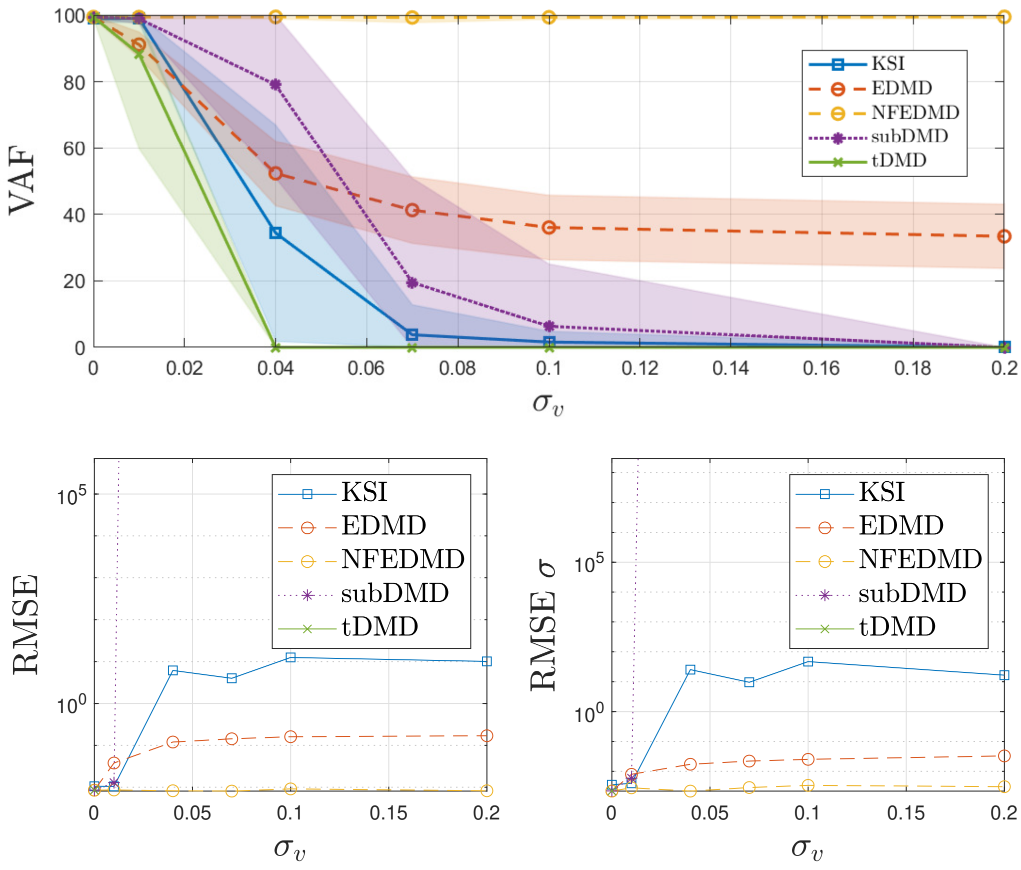


Figure 4-9: Dynamics of the VAF and RMSE for different values of state measurement noise standard deviation σ_v. The following parameters were set: $n = 15$, $m_t = 300$, $N_t = 50$, $N_v = 10$, $m_v = 1000$ and $c = 50$

4-2-2 Van Der Pol Oscillator

This subsection briefly covers the benchmark experiments applied to the Van Der Pol oscillator. This nonlinear system is described by the following 2-dimensional ordinary differential equation

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \zeta(1 - x_1^2)x_2 - x_1, \end{aligned} \tag{4-12}$$

with $x_1, x_2 \in \mathbb{U}$, and $\zeta = 0.5$. The behavior of Eq. (4-12) is shown in the phase portrait of Fig. 4-10. We chose this system for three main reasons. The first is the presence of a limit cycle. The second is that if we choose $x[0] \in \mathbb{U}$ the trajectories will follow an unstable behaviour due to the equilibrium point at the origin. The third is, because of this unstable behaviour, the state trajectories will evolve outside $\mathbb{U}$. For a too high monomial degree, this can drastically increase the values of the observables library. Nevertheless, the chosen monomial degree is 4 which still gives appreciable lifted dynamics. Testing the algorithms on the Van der Pol oscillator will give us insights into their performance in the presence of the three characteristics listed above.

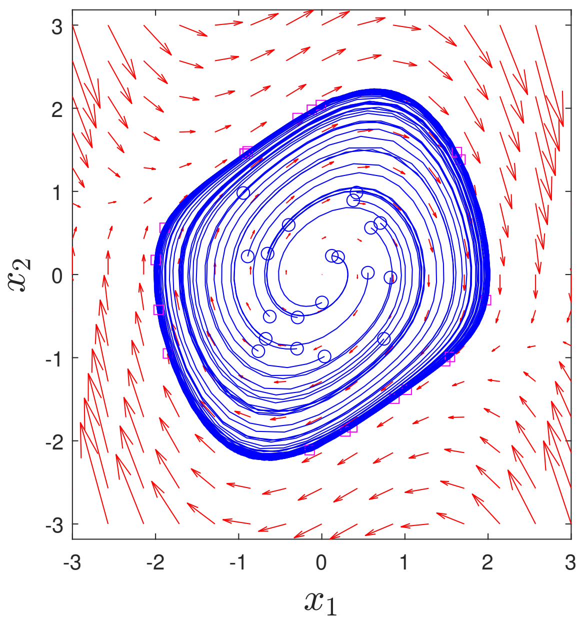


Figure 4-10: Phase portrait of the Van Der Pol oscillator

Changing the Number of Training Trajectories

In this experiment category we check the dynamics of VAF and RMSE for an increasing amount of training trajectories with the fixed parameters of Table 4-3. The resulting performances, given the fixed parameters in Table 4-3, are showed in Fig. 4-11 and Fig. 4-12 for $\sigma_\omega = 0.1$ and $\sigma_v = 0.1$ respectively. We can see from these plots that subDMD replicates the real trajectories of the Van der Pol oscillator better than the Brusselator. As a matter of fact,

Parameter	Value
Number of observables used in the library (n)	15
Number of samples per training trajectory (m_t)	300
Number of validation trajectories (N_v)	10
Number of samples per validation trajectory (m_v)	1000
Number of Monte Carlo cycles (c)	50

Table 4-3: Fixed parameters for the experiment with a changing N_t

in Fig. 4-6 and Fig. 4-7 subDMD was not even displayed as its RMSE was too high. Apart from the improvement of subDMD, we can see an overall worsening in the performances of the algorithms. Nevertheless, NFEDMD is still the best performing solution, followed by KSI.

To conclude we want to emphasize that, like in the Brusselator experiments, the addition of state measurement noise v gives erratic results in the models of the algorithms (see Fig. 4-7 and Fig. 4-12). This is due to the distortion of v caused by the lifting process, which significantly influences the computed models.

Changing the Standard Deviation of the Measurement Noises

This experiment analyzes the robustness performance of the algorithms in the presence of either ω or v. The resulting plots are showed in Fig. 4-13 and Fig. 4-14. To ease the comparison with the Brusselator we put in Fig. 4-15 the VAF dynamics of both Van der Pol oscillator and Brusselator. As we discussed in the previous experiment, there is a worsening of the performances in replicating the Van der Pol oscillator trajectories. This can be seen by simply looking at the no noise case where the VAF is at around 80% while for the Brusselator is 100% in Fig. 4-15. This is probably due to the limit cycle of the system which might be difficult to model. Another possibility is the number of observables used as library. To lift the Van der Pol oscillator trajectories we chose the same amount of observables ($n = 15$) used for the Brusselator. The parameter n is chosen through trial and error, which might give a non-optimal value, leading to lower quality models.

A lower deterioration rate of subDMD can be seen in Fig. 4-15. More in detail, the VAF has a lower decrease rate with an increasing standard deviation of the noise (both for v and ω).

Parameter	Value
Number of observables used in the library (n)	15
Number of samples per training trajectory (m_t)	300
Number of validation trajectories (N_v)	10
Number of samples per validation trajectory (m_v)	1000
Number of Monte Carlo cycles (c)	50

Table 4-4: Fixed parameters for the experiment with a changing standard deviation of the measurements noises

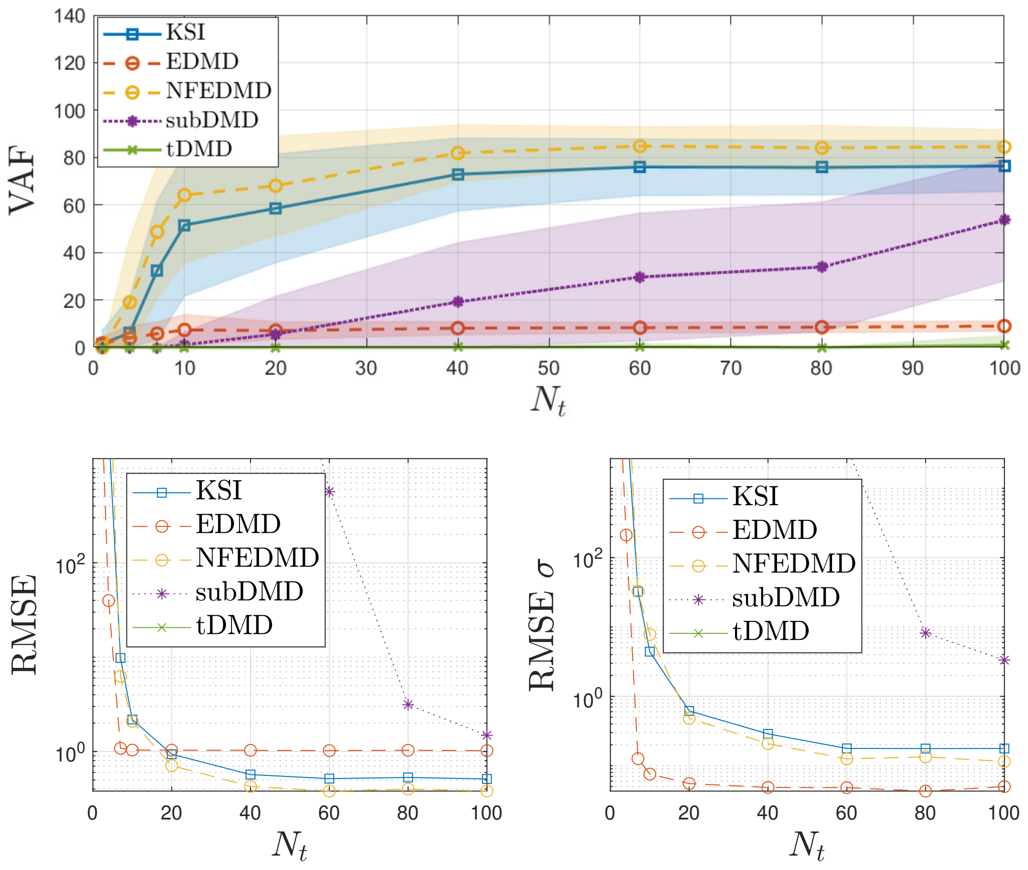


Figure 4-11: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the observables measurement noise was introduced with $\sigma_\omega = 0.1$. The following parameters were set: $n = 15$, $m_t = 300$, $N_v = 10$, $m_v = 1000$ and $c = 50$

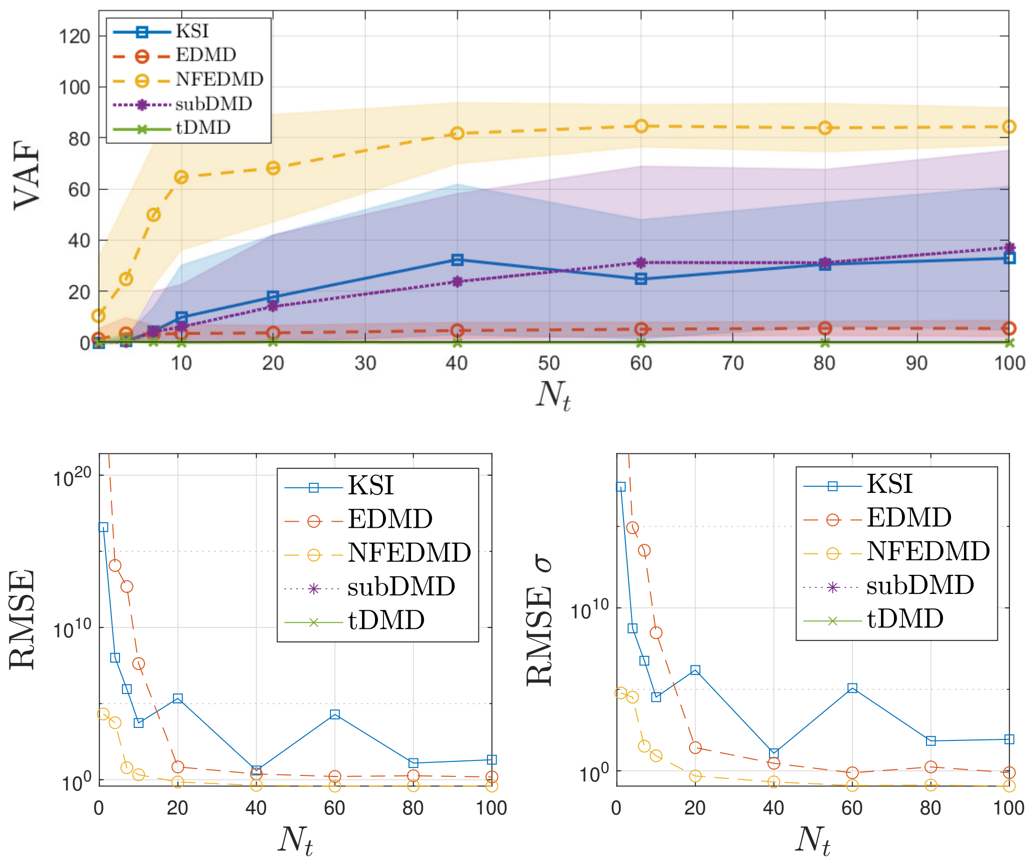


Figure 4-12: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the states measurement noise was introduced with $\sigma_v = 0.1$. The following parameters were set: $n = 15$, $m_t = 300$, $N_v = 10$, $m_v = 1000$ and $c = 50$

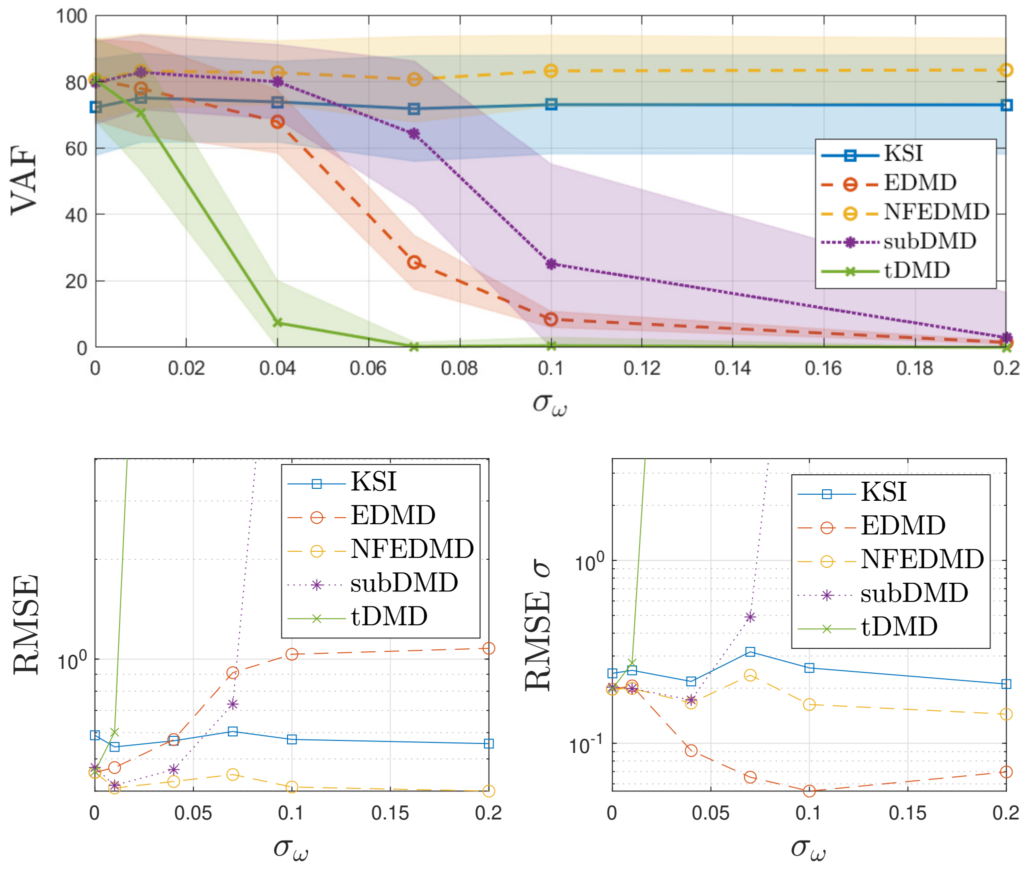


Figure 4-13: Dynamics of the VAF and RMSE for different values of observables measurement noise σ_ω. The following parameters were set: $n = 15$, $m_t = 300$, $N_t = 50$, $N_v = 10$, $m_v = 1000$ and $c = 50$

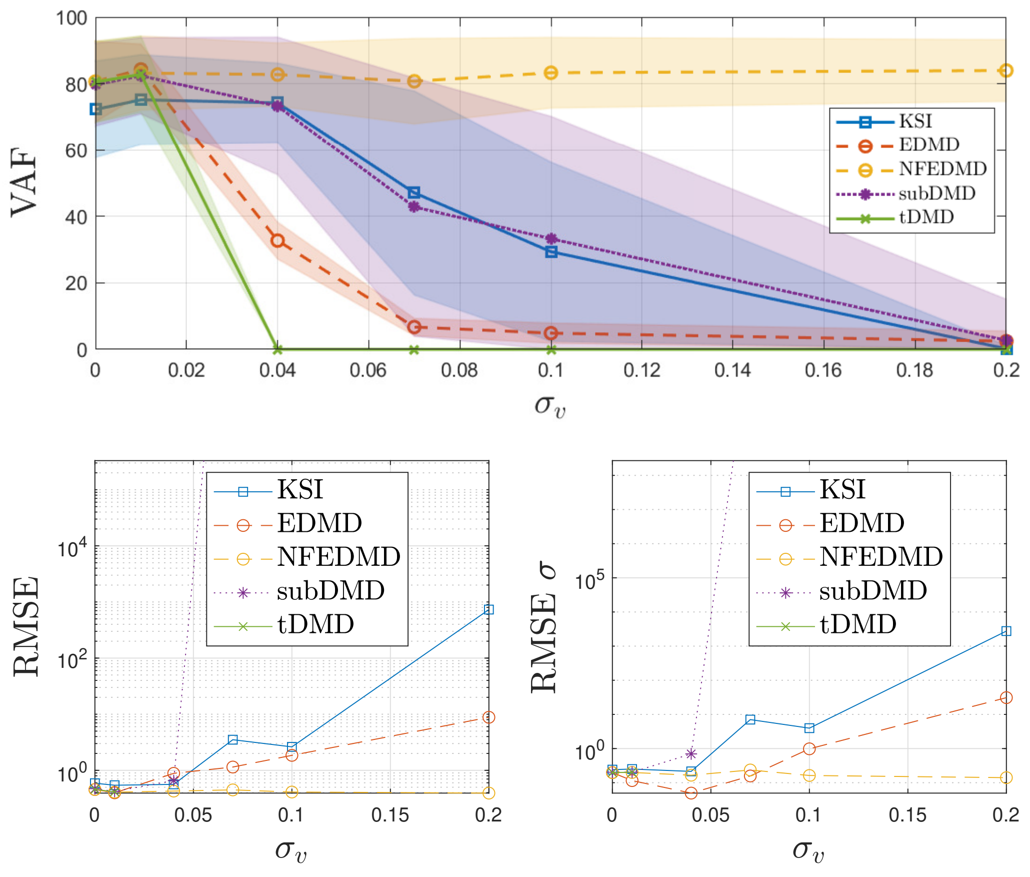


Figure 4-14: Dynamics of the VAF and RMSE for different values of the state measurement noise σ_v. The following parameters were set: $n = 15$, $m_t = 300$, $N_t = 50$, $N_v = 10$, $m_v = 1000$ and $c = 50$

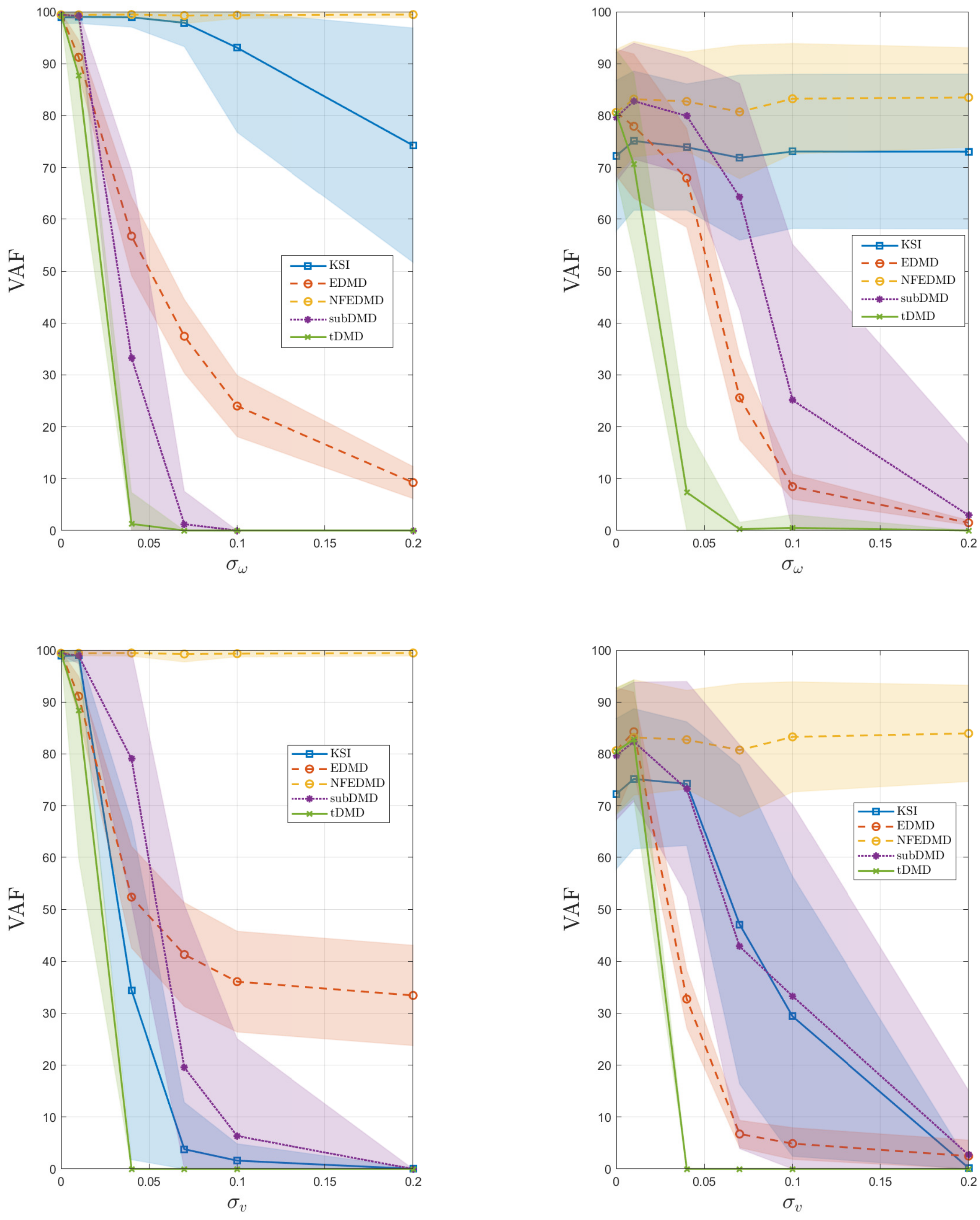


Figure 4-15: Overall VAF dynamics of numerical methods for the Brusselator (left) and the Van der Pol oscillator (right) with an increasing standard deviation of observables measurement noise σ_ω (top) and state measurement noise σ_v (bottom)

Chapter 5

Introducing the Reduced Order KSI with Randomized Singular Value Decomposition

From Chapter 4 we concluded that KSI is a promising technique against the observables measurement noise ω (see Fig. 4-8 and Fig. 4-13). To improve this numerical method we implement two main upgrades. In this chapter, we discuss the enhancements that we designed for KSI during the period of this thesis. More in detail, we introduce a reduced-order realization of KSI with the purpose to compute a more robust model to measurement noise. Additionally, we implement a randomized singular value decomposition to compute the observability matrix up to a similarity transformation. This will decrease the computational times of KSI.

5-1 Reduced-Order KSI (roKSI) in Presence of Measurement Noise

We saw in Chapter 4 that the presence of measurement noise can have a big influence on the resulting model computed by KSI. A possible way to solve this issue is based on eliminating the singular values components ς (see Theorem 3-1.1) that are due to the added measurement noise. In other words, we reduce the order of the state observables y in the KSI model.

We start by introducing the nomenclature based on Section 3-5. Let us have the KSI system in Eq. (3-17) with noisy measurements such that

$$\begin{aligned} y[k+1] &= Ay[k] \\ \mathbf{g}(x[k]) &= Cy[k] + \omega[k], \end{aligned} \tag{5-1}$$

where $\mathbf{g}(x[k]) \in \mathbb{R}^n$ is the library of observables chosen , $y[k] \in \mathbb{R}^{n_y}$ are the state observables and $\omega[k] \sim \mathcal{N}(0, \sigma_\omega^2)$ is additive white Gaussian noise. The measurements of the state observables and additive noise are collected into the matrices $\mathbb{Y}$ and $\mathbb{W}$ respectively. Matrix $\mathbb{W}$ is defined as follows

$$\mathbb{W} = \begin{bmatrix} W_1 & W_2 & \dots & W_{N_t} \end{bmatrix}$$

where

$$W_i = \begin{bmatrix} \omega^{(i)}[0] & \omega^{(i)}[1] & \dots & \omega^{(i)}[m_t - s - 1] \\ \omega^{(i)}[1] & \omega^{(i)}[2] & \dots & \omega^{(i)}[m_t - s] \\ \omega^{(i)}[s-1] & \omega^{(i)}[s-2] & \dots & \omega^{(i)}[m_t - 1] \end{bmatrix}$$

gathers the noise measurements of the i-th trajectory. The data equation of Eq. (5-1) is then defined as

$$\mathbb{H}(\mathbb{X}) = \mathcal{O}\mathbb{Y} + \mathbb{W}.$$

where $\mathbb{H}(\mathbb{X}) \in \mathbb{R}^{s \cdot n \times m_t \cdot N_t}$ is the matrix defined in Eq. (3-21), and $\mathcal{O}$ is the observability matrix. We recall that s indicates the number of groups of n rows present in $\mathbb{H}(\mathbb{X})$, i.e. the number of time delays in the Hankel matrix.

Now we show that it is possible to retrieve the system matrices A and C of Eq. (5-1) even if the measurements are noisy. We start by stating Lemma 9.3 of Verhaegen et al. [14] for our current situation:

Lemma 5-1.1. Let us have the system in Eq. (5-1), with the state observables measurements matrix $\mathbb{Y}$ satisfying

$$\text{rank}\left(\lim_{m_t\to\infty} \frac{1}{m_t}\left(\mathbb{Y}\mathbb{Y}^T\right)\right) = n_y,$$

along with $\mathbb{W}$ satisfying

$$\lim_{m_t\to\infty} \frac{1}{m_t}\left(\mathbb{W}\mathbb{W}^T\right) = \sigma_\omega^2 I_{sn},$$

where I_{sn} is an identity matrix of dimension $sn \times sn$. In the computation of the SVD (see Theorem 3-1.1) of matrix

$$\lim_{m_t\to\infty} \frac{1}{m_t}\left(\mathbb{H}(\mathbb{X})\mathbb{H}(\mathbb{X})^T\right) = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_{n_y}^2 + \sigma_\omega^2 I_{sn_y} & 0 \\ 0 & \sigma_\omega^2 I_{sn-n_y} \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} \tag{5-2}$$

we have that the $n_y \times n_y$ diagonal matrix $\Sigma_{n_y}^2$ contains the nonzero singular values of matrix

$$\mathcal{O}\left(\lim_{m_t\to\infty} \frac{1}{m_t}\left(\mathbb{Y}\mathbb{Y}^T\right)\right)\mathcal{O}^T.$$

Matrix U_1 from Eq. (5-2) then satisfies

$$\text{range}(U_1) = \text{range}(\mathcal{O}).$$

The proof of this lemma is given in [14, Page 308].

We can infer from Lemma 5-1.1 that in case we have additive white noise in the measurements we can distinguish the disturbed dynamics from noise. In other words, we split the singular values related to the disturbed states observables y from the ones related to the noise ω. If we can find the right order n_y of the disturbed system, then we can compute from U_1 the column space of the observability matrix. From here the matrices A and C can be computed up to a similarity transformation through the KSI algorithm in Section 3-5.

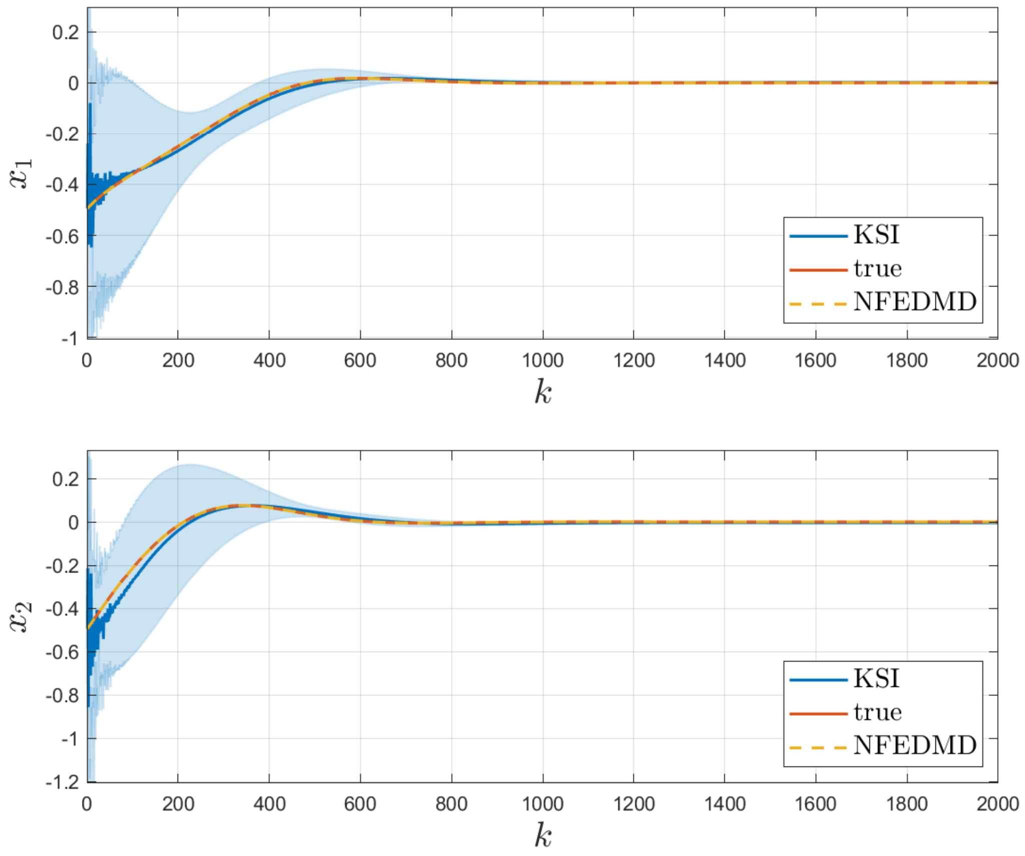


Figure 5-1: Trajectories of KSI and NFEDMD in the presence of observables measurement noise ω ($\sigma_\omega = 0.1$) with respect to the Brusselator behaviour

The distinction between the n_y disturbed singular values and the $sn - n_y$ noise singular values depends on the gap between n_y and $n_y + 1$ (see [14, Page 310]). If the gap is big enough then the distinction is easily determinable. To help us find this gap, the VAF and RMSE can be computed for each reduced-order n_y, and choose the best performing one. We will see in the following example that another possible way to determine n_y is by looking at when the singular values become constant.

Example 2. Let us make a practical example by adopting the Brusselator as an underlying system. In this example, only the observables measurement noise ω with $\sigma_\omega = 0.1$ is introduced. The training is done with $N_t = 10$ training trajectories and $m_t = 1000$ samples per trajectory. Additionally, a 4th order monomial basis is used as library of observables, i.e. $n = 15$. In Fig. 5-1 it is possible to see the trajectories result of a full order KSI ($n = 15$, $n_y = 15$) along with NFEDMD. The trajectories of KSI show the degrading influence of ω to the prediction properties of the approximated Koopman operator $\mathbf{K}$. In Fig. 5-2 we can also see the singular values ς of Eq. (5-2) on a logarithmic scale. After seeing the poor performance of KSI in Fig. 5-1, we wish to truncate the order n_y to decrease the effect of ω on the computation of $\mathbf{K}$. To do so, we rely on the singular values of Fig. 5-2 along with the VAF and RMSE for each order n_y chosen. These performance measures values are shown in Fig. 5-3, from which we can conclude that $n_y = 4$ is the best choice for the order of the

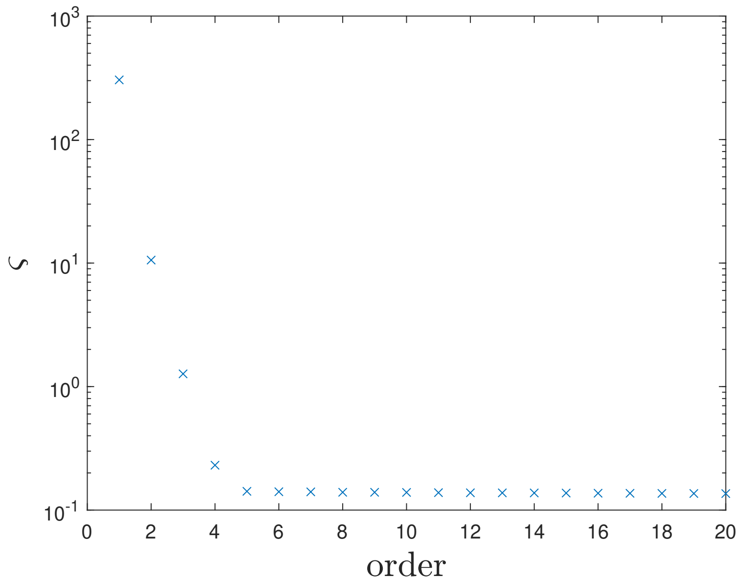


Figure 5-2: Singular values ς of Eq. (5-2)

state observables y. This was already deductible from Fig. 5-2. Now that n_y is chosen we can compare the resulting KSI trajectories with respect to NFEDMD. The trajectories of the reduced-order KSI (roKSI) are shown in Fig. 5-4, where we can see the improved performance of roKSI against the full order version in Fig. 5-1.

$\triangle$

The implementation of the roKSI is based on running the KSI algorithm with different values of the truncated n_y order. After computing the VAF for all the n_y, the one with the best performance is used to compute matrix $\mathbf{K}$ of roKSI. As we showed in Example 2, it is also possible to look at the singular values, like in Fig. 5-2, and truncate the order of n_y when ς becomes low enough and constant.

5-2 Introducing Randomized Singular Value Decomposition in KSI (KSIrsvd)

One of the most important drawbacks of the deterministic SVD is its high cost in terms of floating-point operations [46, Section 1.4.1]. Especially in the case of big matrices decomposition, this issue can lead to high computational times. Taking into account that the data matrices that we work with, e.g. $\mathbb{H}(\mathbb{X})$, can be quite big in size ($\sim 10^2 \times 10^4$), the programs running times can become impractical. From Section 5-1 we understood that behind the noisy lifted measurements we can find dominant dynamics. These can be used to approximately reconstruct the behaviour of the underlying nonlinear system. In other words, we truncate the order $n_y \leq n$ of our lifted system (see Eq. (5-1)) creating a low-rank approximation of the measurement data. With this in mind, we decided to adopt the randomized singular value decomposition (rSVD) to decrease the computational times of the KSI algorithm.

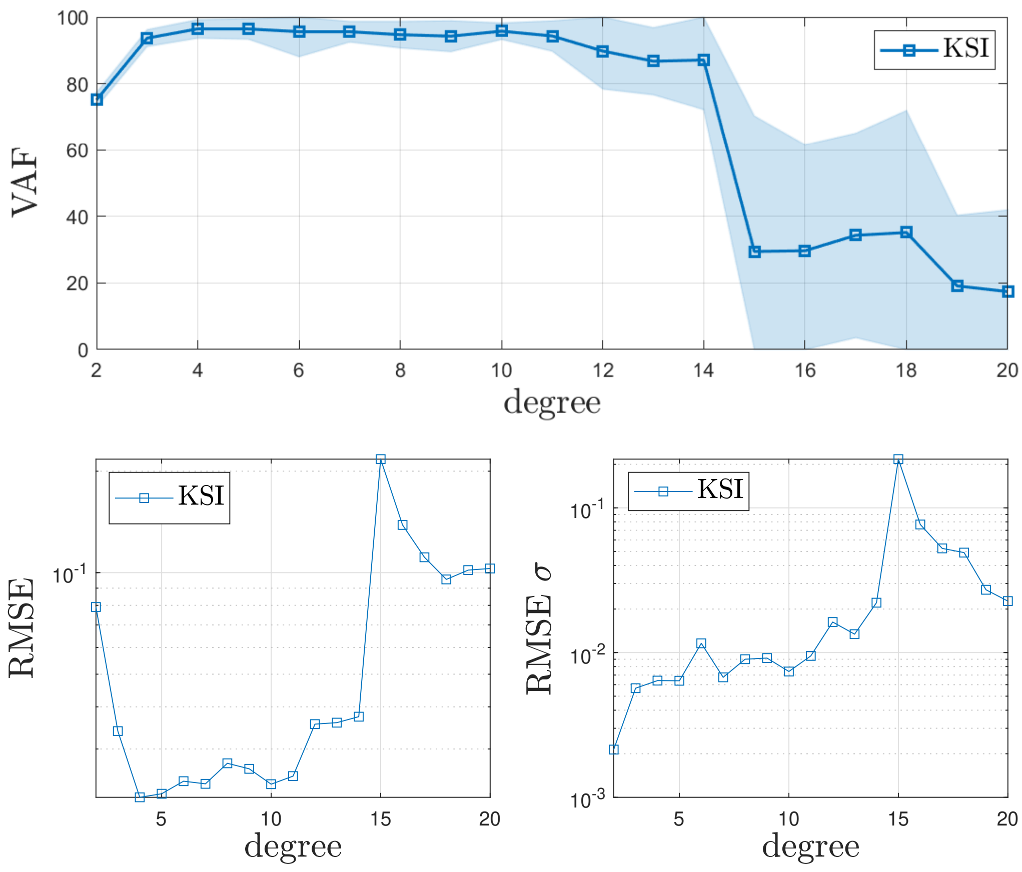


Figure 5-3: VAF and RMSE of the KSI trajectories with respect to the true ones of the Brusselator

Before describing the implementation of the rSVD, we introduce the theory of randomized matrix approximations based on the work of Halko et al. in [46, Section 1.2]. Let us have a matrix $A \in \mathbb{R}^{a\times b}$ and its SVD is of the form $A = U_A\Sigma V^T$ (see Theorem 3-1.1). The decomposition of A can be seen as a 2 stage process: in the first stage we find a $a \times r$ matrix Q such that

$$A \approx QQ^TA.$$

We need Q to have all its columns orthonormal, i.e. $QQ^T = I$, and with as few columns as possible. In the second stage we first form a matrix $B = Q^TA$, then, by computing its (deterministic) SVD, we get

$$B = U_B\Sigma V^T.$$

To finish the decomposition we determine the singular left vectors matrix $U_A = QU_B$.

The main improvement of introducing randomization in the process discussed above lies in the computation of matrix Q in the first stage. Assuming that $\text{rank}(A) = r$, the first step is to find a $b \times r$ matrix P, with r a chosen fixed parameter, such that the resulting Z in

$$Z = AP,$$

approximately spans the range of A. If we decide to take each column of P as a random Gaussian vector with each entry i.i.d we have that Z will span the range of A. This is

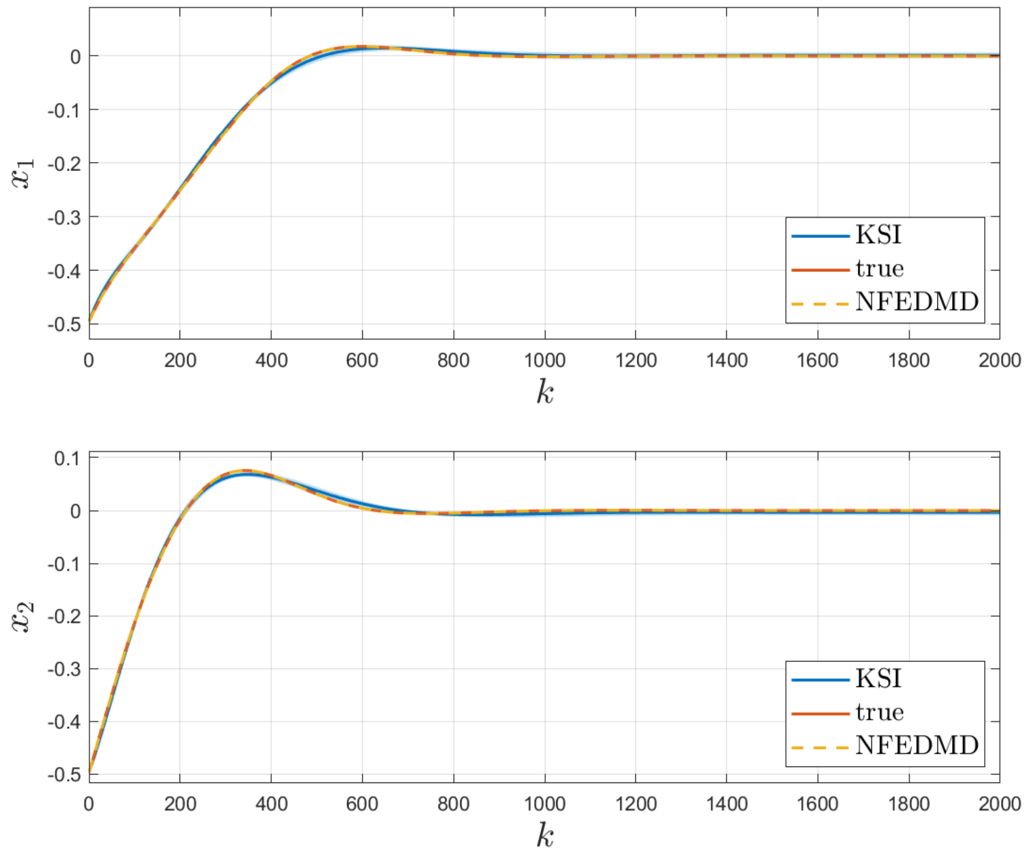


Figure 5-4: Trajectories of the reduced order KSI ($n_y = 4$) and NFEDMD in the presence of observables measurement noise ω ($\sigma_\omega = 0.1$) with respect to the Brusselator behaviour

because the columns of P are i.i.d. which means that they are also linearly independent with probability one. This results in the columns of Z being also linearly independent, spanning the range of A (see [46, Section 1.3.2]). There might be the case where the assumption on the rank of A is wrong and $\text{rank}(A) > r$. In this situation, we need P to span an r dimensional subspace of A. To do so, we can increase our chances of spanning this subspace by adding p columns to P, making it a $b \times (r+p)$ matrix. The constant p is called *oversampling parameter* [2, Section 1.8]. Generally the rule of thumb is to set $p = 5$ or $p = 10$ for good results [46].

Now that we have a matrix Z spanning the subspace of A that interests us, we need to compute Q. We remind that the columns of Q need to be orthonormal. With this in mind, a possible solution is to compute the economized QR decomposition of Z (see [47, Section 2]) such that

$$Z = QR. \tag{5-3}$$

From Eq. (5-3) we determine matrix Q which will be used in the second stage to carry out the SVD of matrix A. In Halko et al. [46, Section 1.2], the number of floating-point operations, *flops*, is compared between SVD and rSVD. For a $a \times b$ matrix A the SVD has a number of flops of the order $O(\text{rank}(A)ab)$. The rSVD, instead, can reach an order of flops of $O(ab\log(\text{rank}(A)) + (a+b)\text{rank}(A)^2)$.

The rSVD algorithm that we implemented was taken from the work of Brunton et al. [2, Section 1.8] and is shown below. In this algorithm we replace matrix A with matrix $\mathbb{H}(\mathbb{X})$ of Eq. (3-21). This is done to comply with the nomenclature of Section 3-5.

Algorithm 6: Randomized SVD

```
Input: $\mathbb{H}(\mathbb{X}) \in \mathbb{R}^{(s \cdot n)\times(N_t \cdot m_t)}$, the low-rank parameter $n_y \leq n$, with $n_y = r$, and the
        oversampling parameter $p$
Output: The SVD of $\mathbb{H}(\mathbb{X})$
1 Construct the random projection $P \in \mathbb{R}^{(s \cdot n)\times(n_y+p)}$;
2 Compute $Z = (\mathbb{H}(\mathbb{X})\mathbb{H}(\mathbb{X})^T)\mathbb{H}(\mathbb{X})P$;
3 Compute QR decomposition of $Z = QR$;
4 Form $B = Q^T\mathbb{H}(\mathbb{X})$;
5 Compute the SVD of $B = U_B \Sigma V^T$;
6 Compute $U = QU_B$ where $\mathbb{H}(\mathbb{X}) = U\Sigma V^T$.
```

Step 2 was taken from [2, Eq. 1.40] and is used to have a matrix $(\mathbb{H}(\mathbb{X})\mathbb{H}(\mathbb{X})^T)\mathbb{H}(\mathbb{X})$ with a faster decay of the singular values with respect to $\mathbb{H}(\mathbb{X})$. This is done in case the singular value of $\mathbb{H}(\mathbb{X})$ decay slowly, leading to difficulties in finding the right truncation order.

The deterministic SVD of KSI is replaced with rSVD (see Algorithm 5). To see the performance of KSI with rSVD, we tested it on the Brusselator and checked its computational time with respect to the number of training trajectories N_t given in input. A 4th order monomial basis is used as observables library. Each training trajectory had 100 samples. No measurement noise is introduced, and the truncating-order parameter is set at $n_y = n$. The programming environment used is MATLAB 2019b and it was run on an Intel(R) Core(TM) i7-8565U CPU 1.80GHz-1.99 GHz. With an oversampling of $p = 5$, the result is shown in Fig. 5-5 in comparison with KSI using the deterministic SVD. Here it is possible to see the higher efficiency of the rSVD in terms of computation time.

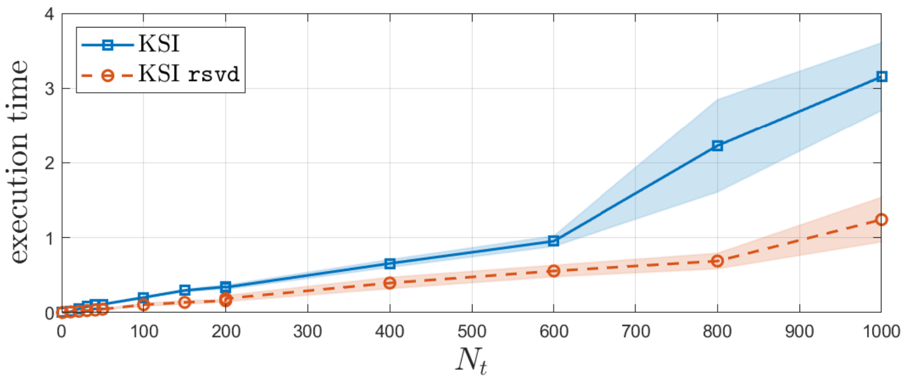


Figure 5-5: Execution times of the KSI with rSVD compared to KSI with deterministic SVD with respect to the number of training trajectories N_t given

5-3 Benchmark Results with Improved KSI

In this section, we will show the performance results of the KSI algorithm updated with the improvements discussed above. The resulting algorithm has a reducing-order routine, i.e. a routine that selects the rank parameter n_y, and adopts the rSVD for the computation of the observability matrix. We will refer to this algorithm as the reduced order KSI with randomized singular value decomposition (roKSIrsvd). The reducing order routine of roKSIrsvd computes the RMSE[1] for different $\mathbf{K}$ resulting from different values of n_y. We then choose the n_y and $\mathbf{K}$ with the lowest RMSE value. Algorithm 7 summarizes the roKSIrsvd by integrating the concepts introduced in Section 5-1 and Section 5-2. Looking at the results showed in Chapter 4, we decided to compare roKSIrsvd with KSI and NFEDMD in terms of VAF and RMSE. This decision was taken as the last two algorithms are the best performing ones with respect to the others listed in Chapter 3. Here we will evaluate roKSIrsvd applied to the Brusselator. The VAF and RMSE will be computed with respect to the number of trajectories N_t, and the standard deviations σ_v and σ_ω of the measurement noises. The results of the first benchmark with an increasing N_t are showed in Fig. 5-7 and Fig. 5-8 with the additions of noise ω and v respectively. Especially in the case of additive noise in the state measurements (Fig. 5-8) roKSIrsvd outperforms KSI. For what regards Fig. 5-7, roKSIrsvd computes a better $\mathbf{K}$ than KSI until $N_t \approx 50$. After that KSI has enough training data to compute a more performing model. This can be due to the adoption of rSVD which computes an approximated version of the SVD. The closeness to the NFEDMD performance makes the roKSIrsvd one of the most robust algorithms in this thesis tested in the framework defined in Chapter 4. A second proof of the very good robustness of roKSIrsvd is shown in the plots of Fig. 5-9 and Fig. 5-10. Here the performance values are computed with respect to an increasing standard deviation of the measurement noises ω and v.

The main drawback of roKSIrsvd is its processing time. The reducing-order routine increases the time needed to compute the resulting approximated Koopman operator. This is because there is an internal cycle that validates $\mathbf{K}$ for different values of r. Instead of computing $\mathbf{K}$ once, we compute it $n-1$ times, therefore increasing the computational times. But if the optimal reduced-order parameter r is already known, then the roKSIrsvd is faster than the KSI thanks to the rSVD. In the case of unknown n_y, a quantitative result that proves the higher computational times of roKSIrsvd is given in Fig. 5-6.

In conclusion, we can confirm that reducing the order of the KSI model and computing the decomposition of $\mathbb{H}(\mathbb{X})$ with rSVD improves the robustness of the algorithm to noise and its computational times[2]. As the performance of roKSIrsvd is comparable to NFEDMD, we can also affirm that it outperforms the other listed algorithms of Chapter 3 in the benchmark framework of Chapter 4.

[1]The RMSE generally mirrors the VAF, i.e. when the latter increases, the former decreases. We chose to use the RMSE instead of VAF to take into account possible biases.

[2]In case n_y is already known.

Algorithm 7: Reduced order KSI with randomized singular value decomposition (roKSIrsvd)

```
**Input:** $\mathbb{X}$
**Output:** Matrix approximation $\mathbf{K}$ of the Koopman operator
1  Set the order truncating parameter $r = 2$, choose $\mathbf{g}(x) \in \mathbb{R}^n$, and construct
   $\mathbb{H}(\mathbb{X}) = \left[ H(\mathbb{X}_1) \quad H(\mathbb{X}_2) \quad \dots \quad H(\mathbb{X}_{N_t}) \right]$, from Eq. (3-21);
2  while $r \leq n$ do
3      Compute the rSVD (see Section 5-2) of $\mathbb{H}(\mathbb{X}) = U\Sigma V^T$ and take the first $r$ columns
       of $U$ resulting in $U_r$;
4      Assuming range($\mathbb{H}(\mathbb{X})$) = range($\mathcal{O}$) then compute
       $C_T = U_{n_y}(1:n, 1:r)$, $A_T = U_r(1:(s-1)n, 1:r)^\dagger U_r(n+1:sn, 1:r)$;
5      Compute $\mathbf{K}_r = C_T A_T C_T^\dagger$;
6      Validate $\mathbf{K}_r$ and compute the RMSE($r$) with respect to the real system trajectories;
7      $r = r + 1$;
8  end
9  Find $n_y = \arg\min_r \text{RMSE}(r)$ with $r = 2, \dots, n$;
10 The resulting Koopman matrix approximation is $\mathbf{K} = \mathbf{K}_{n_y}$.
```

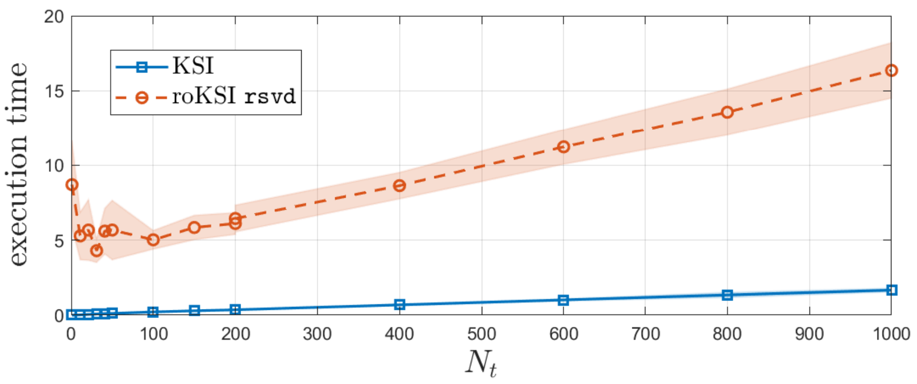


Figure 5-6: Execution times of the roKSIrsvd compared to KSI with respect to the number of training trajectories N_t

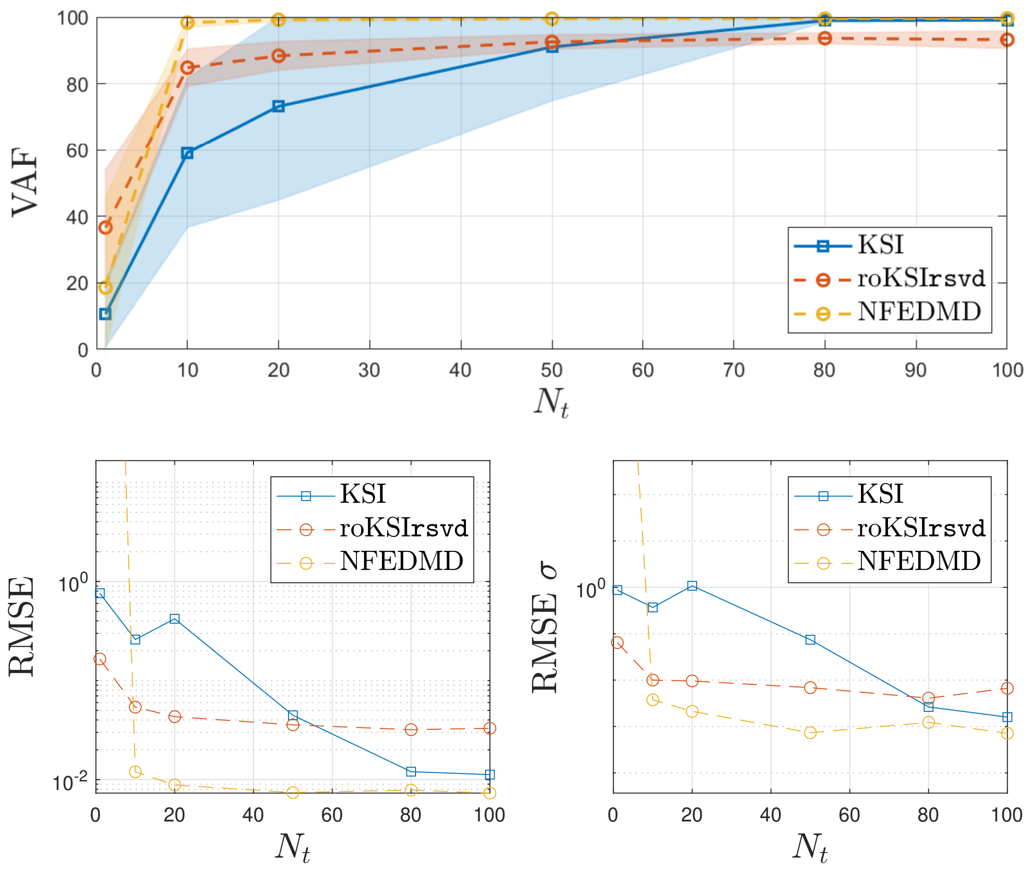


Figure 5-7: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the observables measurement noise ω was introduced with $\sigma_\omega = 0.1$. The following parameters were set: $d = 4$ (or $n = 15$), $m_t = 300$, $N_v = 100$, $m_v = 1000$ and $c = 10$

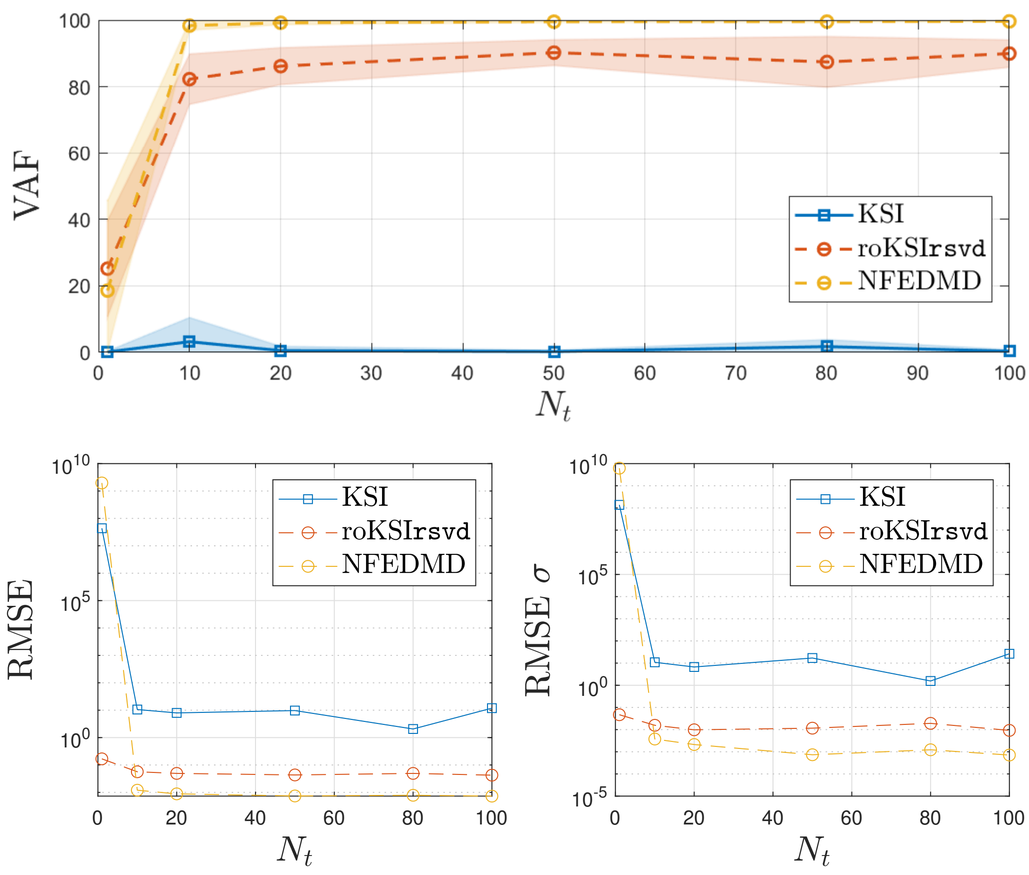


Figure 5-8: Dynamics of the VAF and RMSE for different values of training trajectories N_t. Here, only the state measurement noise v was introduced with $\sigma_v = 0.1$. The following parameters were set: $d = 4$ (or $n = 15$), $m_t = 300$, $N_v = 100$, $m_v = 1000$ and $c = 10$

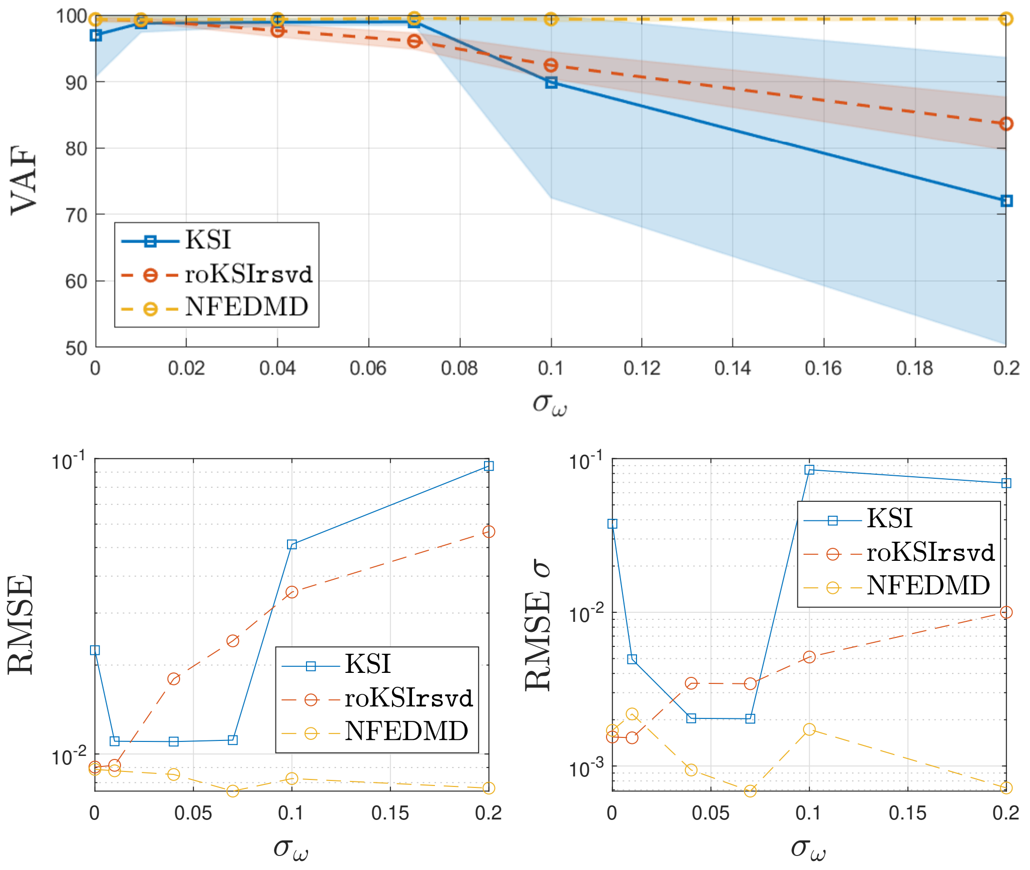


Figure 5-9: Dynamics of the VAF and RMSE for different values of observables measurement noise standard deviation σ_ω. The following parameters were set: $d = 4$ (or $n = 15$), $m_t = 300$, $N_t = 50$, $N_v = 100$, $m_v = 1000$ and $c = 10$

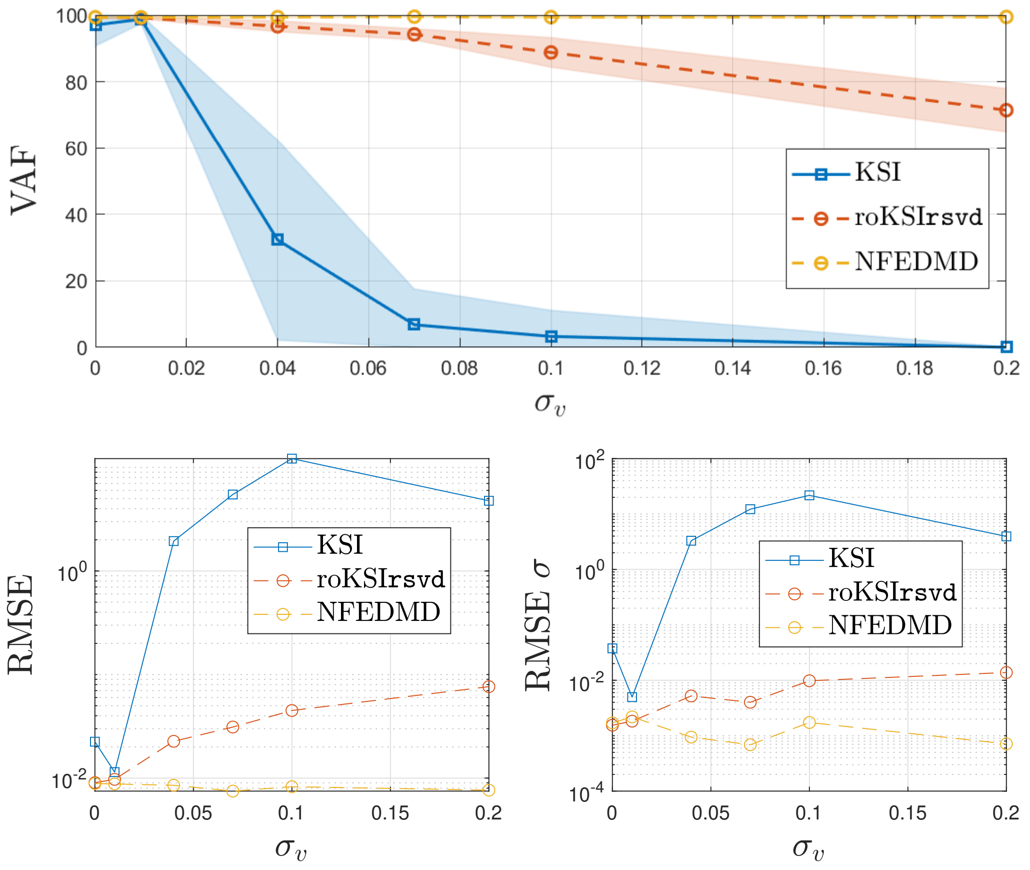


Figure 5-10: Dynamics of the VAF and RMSE for different values of state measurement noise standard deviation σ_v. The following parameters were set: $d = 4$ (or $n = 15$), $m_t = 300$, $N_t = 50$, $N_v = 100$, $m_v = 1000$ and $c = 10$

Chapter 6

Conclusions

In Chapter 2 we introduced the main concepts that form the theoretical pillars behind the Koopman operator framework. This gave us the knowledge needed to describe the algorithms designed to compute matrix approximations of this composition operator. More in detail, in Chapter 3 we explained the functioning of tDMD [17], EDMD [1], subDMD [12], NFEDMD, and KSI. As we introduced in Chapter 1, one of the main issues in the computation of prediction models with machine learning techniques is noise [15]. The ability to determine a well approximated Koopman operator is crucial in the case of noisy training data. This is why this thesis focused on this issue by evaluating the above-mentioned algorithms and improving the most promising one.

In Chapter 4 we evaluated and discussed the performance of the state-of-the-art numerical techniques introduced in Chapter 3 in the presence of noisy training data sets. In particular, we applied additive white Gaussian noise in two different places: state measurements and observable measurements. While the observable measurement noise will preserve its additive white Gaussian noise features, the state measurement noise will loose these features because of the lifting process. This difference is also the reason why we chose to add noise in these two particular places. The quality of the models was quantified by computing the VAF and RMSE of the prediction error. We showed that the performance of subDMD, tDMD, and EDMD can drastically degrade with the introduction of measurement noise (either in the states or observables). On the other hand, the NFEDMD exhibited outstanding performances as it uses noise-free training data. KSI, instead, showed good robustness results when white Gaussian noise was added to the observables, while it computed impractical models in case of noise in the states. During the overall evaluation analysis, particular attention was given to noisy state measurements. This proved to be the most model-degrading case. We assumed that the reason for which the algorithms perform so poorly is that the state measurement noise is distorted nonlinearly by the lifting process. This can result in a non-white-Gaussian type of additive noise, which can lead to worse models.

The KSI technique is a promising framework to make improvements on thanks to its good robustness results against observables measurement noise. With this in mind, we implemented two main upgrades on the algorithm: a reduced-order routine, and the randomized singular value decomposition (rSVD). We named the resulting numerical method reduced order Koopman subspace identification with randomized singular value decomposition (roKSIrsvd). With the robustness of a lower order model and the speed of the rSVD, roKSIrsvd reaches the performance levels of the optimal NFEDMD algorithm. The main advantage of the roKSIrsvd is that it does not require any knowledge of the noise-free observables. The drawback of using this algorithm lies in the computational times. The roKSIrsvd needs to run a validation cycle to check the RMSE for each reduced-order degree value. This drastically increases the time needed to find the best approximated Koopman operator. But if the reduced-order parameter is already known before, then the computation times are faster than KSI thanks to the rSVD.

In this thesis, we showed that many state-of-the-art numerical methods can be quite sensitive to noise and deteriorate significantly, especially in the case of noise in the states. This issue is solved by designing algorithms with good robustness characteristics such as the proposed roKSIrsvd. We showed that the best performing technique in this thesis is the NFEDMD compared to the other algorithms. The main drawback of this numerical method is that it requires knowledge of the noise-free observables. To overcome this drawback, we chose the most promising algorithm that needed only noisy observables and improved it. The choice led to the adoption of KSI which, after the upgrades, became the roKSIrsvd. We tested the latter against the NFEDMD. The results showed that, without any knowledge of the noise-free observables, we can almost achieve the appreciable performance of NFEDMD with roKSIrsvd.

Future Work

The evaluation framework was implemented with a monomial basis as library of observables. While our choice is fixed, techniques based on neural networks can be used to learn a suitable function basis that spans Koopman-invariant subspaces. An example is showed in the work of Takeishi et al. [48]. These techniques can be used to compute a more appropriate observables library in our framework. Along with this adaptive approach, many other types of functions can be tested as library. A few examples are the Bernstein polynomials, Hermite polynomials, radial basis functions, and Fourier basis.

In the paper of Proctor et al. [11] a DMD version with control input was designed and named DMDc. The book of Verhaegen et al. [14] describes subspace identification considering also the presence of possible inputs. With this in mind, KSI and roKSIrsvd can be upgraded to model also the effect of control inputs on the dynamics of the observables. This can lead to a more real-world-oriented algorithm that can be used on a wider range of practical applications.

The noise adopted in this thesis was always additive white Gaussian noise, apart from the resulting state measurement noise after the lifting process. It would be interesting to test these algorithms against other types of noise, e.g. Brownian noise or pink noise.

To finish the future work list, the writer believes that the most effective way to test an algorithm is by applying it to real-case scenarios. This is to say that it would be interesting to test the described numerical methods on data sets from actual measurements of a real system, e.g. neuro-recordings, electric power systems data, and measurements from fluid dynamics experiments.

Bibliography

[1] 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*, vol. 25, no. 6, pp. 1307–1346, 2015.

[2] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control.* USA: Cambridge University Press, 1st ed., 2019.

[3] Y. Lan and I. Mezić, “Linearization in the large of nonlinear systems and Koopman operator spectrum,” *Physica D: Nonlinear Phenomena*, vol. 242, no. 1, pp. 42–53, 2013.

[4] I. Mezić, “Koopman operator, geometry, and learning,” *arXiv*, 2020.

[5] B. O. Koopman, “Hamiltonian systems and transformation in hilbert space,” *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[6] B. O. Koopman and J. v. Neumann, “Dynamical systems of continuous spectra,” *Proceedings of the National Academy of Sciences*, vol. 18, no. 3, pp. 255–263, 1932.

[7] M. Budišić, R. Mohr, and I. Mezić, “Applied Koopmanism,” *Chaos*, vol. 22, no. 4, 2012.

[8] I. Mezic, “Spectral properties of dynamical systems, model reduction and decompositions,” *Nonlinear Dynamics*, vol. 41, pp. 309–325, 2005.

[9] P. J. Schmid, “Dynamic mode decomposition of numerical and experimental data,” *Journal of Fluid Mechanics*, vol. 656, p. 5–28, 2010.

[10] C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. S. Henningson, “Spectral analysis of nonlinear flows,” *Journal of Fluid Mechanics*, vol. 641, p. 115–127, 2009.

[11] J. L. Proctor, S. L. Brunton, and J. N. Kutz, “Dynamic mode decomposition with control,” *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[12] N. Takeishi, Y. Kawahara, and T. Yairi, “Subspace dynamic mode decomposition for stochastic koopman analysis,” *Phys. Rev. E*, vol. 96, p. 033310, Sep 2017.

[13] Y. Lian and C. N. Jones, "Learning Feature Maps of the Koopman Operator: A Subspace Viewpoint," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 860–866, IEEE.

[14] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach.* Cambridge University Press.

[15] S. Gupta and A. Gupta, "Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review," *Procedia Computer Science*, vol. 161, pp. 466–474, 2019.

[16] S. T. M. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley, "Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition," *Experiments in Fluids*, vol. 57, no. 3, p. 42.

[17] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, "De-biasing the dynamic mode decomposition for applied Koopman spectral analysis of noisy datasets," *Theoretical and Computational Fluid Dynamics*, vol. 31, no. 4, pp. 349–368.

[18] C. K. R. T. Jones, *Whither Applied Nonlinear Dynamics?*, pp. 631–645. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[19] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[20] A. Mauroy, I. Mezic, and Y. Susuki, *The Koopman Operator in Systems and Control Concepts, Methodologies, and Applications: Concepts, Methodologies, and Applications.* 2020.

[21] S. Wahls, "Sc42135_6_koopman_operators_i_2020-handout-nup." `https://brightspace.tudelft.nl/d2l/le/content/195024/viewContent/1635708/View`, 2020.

[22] A. Mauroy and I. Mezić, "Global Stability Analysis Using the Eigenfunctions of the Koopman Operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[23] H. Arbabi, "Introduction to Koopman operator theory of dynamical systems," 2019.

[24] Y. Susuki, I. Mezic, F. Raak, and T. Hikihara, "Applied Koopman operator theory for power systems technology," *Nonlinear Theory and Its Applications, IEICE*, vol. 7, no. 4, pp. 430–459.

[25] S. L. Brunton, "Notes on Koopman Operator Theory," *arXiv*, no. 3, pp. 1–37, 2018.

[26] M. Georgescu and I. Mezić, "Building energy modeling: A systematic approach to zoning and model reduction using Koopman Mode Analysis," *Energy and Buildings*, vol. 86, pp. 794–802.

[27] H. Arbabi and I. Mezić, "Study of dynamics in post-transient flows using Koopman mode decomposition," *Physical Review Fluids*, vol. 2, no. 12, pp. 1–31.

[28] C. W. Rowley, I. Mezi, S. Bagheri, P. Schlatter, and D. S. Henningson, "Spectral analysis of nonlinear flows," *Journal of Fluid Mechanics*, vol. 641, no. Rowley 2005, pp. 115–127, 2009.

[29] J. Mann and J. N. Kutz, "Dynamic Mode Decomposition for Financial Trading Strategies." Comment: 18 pages, 7 figures. arXiv admin note: text overlap with arXiv:1506.00564.

[30] M. Korda and I. Mezic, "Optimal construction of Koopman eigenfunctions for prediction and control," *IEEE Transactions on Automatic Control*, pp. 1–1, 2020.

[31] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLoS ONE*, vol. 11, no. 2, pp. 1–19, 2016.

[32] Y. Susuki and I. Mezić, "Nonlinear Koopman modes and power system stability assessment without models," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 899–907, 2014.

[33] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Philadelphia, PA, USA: SIAM-Society for Industrial and Applied Mathematics, 2016.

[34] A. Chatterjee, "An introduction to the proper orthogonal decomposition," *Current Science*, vol. 78, no. 7, pp. 808–817, 2000.

[35] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.

[36] J. Nathan Kutz, J. L. Proctor, and S. L. Brunton, "Applied Koopman theory for partial differential equations and data-driven modeling of spatio-temporal systems," *Complexity*, vol. 2018, no. ii, 2018.

[37] N. Parmar, H. Refai, and T. Runolfsson, "A Survey on the Methods and Results of Data-Driven Koopman Analysis in the Visualization of Dynamical Systems," *IEEE Transactions on Big Data*, pp. 1–17, 2020.

[38] `DeepAI`, "White noise (statistics)."
https://deepai.org/machine-learning-glossary-and-terms/white-noise.

[39] S. J. Qin, W. Lin, and L. Ljung, "A novel subspace identification approach with enforced causal models," *Automatica*, vol. 41, no. 12, pp. 2043–2053.

[40] L. Duchesne, E. Feron, J. D. Paduano, and M. Brenner, "Subspace identification with multiple data sets," *1996 Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–9.

[41] B. Schutter, "Minimal state-space realization in linear system theory: An overview," *Journal of Computational and Applied Mathematics*, vol. 121, no. 1, pp. 331–354.

[42] S. Raychaudhuri, “Introduction to monte carlo simulation,” in *2008 Winter Simulation Conference*, pp. 91–100, 2008.

[43] E. W. Weisstein, “Runge-kutta method.” https://mathworld.wolfram.com/Runge-KuttaMethod.html. From MathWorld–A Wolfram Web Reseource.

[44] M. Kamb, E. Kaiser, S. L. Brunton, and J. Nathan Kutz, “Time-delay observables for koopman: Theory and applications,” *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 2, pp. 886–917, 2020.

[45] K. P. Burnham, D. R. Anderson, and K. P. Burnham, *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. New York: Springer, 2nd ed ed., 2002.

[46] N. Halko, P.-G. Martinsson, and J. A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions,” *arXiv:0909.4061 [math]*, Dec. 2010.

[47] N. Boullé and A. Townsend, “A generalization of the randomized singular value decomposition,” *arXiv:2105.13052 [cs, math, stat]*, May 2021. Comment: 13 pages, 4 figures.

[48] N. Takeishi, Y. Kawahara, and T. Yairi, “Learning koopman invariant subspaces for dynamic mode decomposition,” in *NIPS*, 2017.