Department of Mathematics
TUM School of Computation, Information and Technology
Technical University of Munich

Dynamic Mode Decomposition Variants: A Critical Comparative Analysis

Chair of Numerics of Complex Systems

Giacomo Lupo

Thesis for the attainment of the academic degree

Master of Science

at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisor:
Prof. Dr. Oliver Junge

Advisors:
Prof. Dr. Oliver Junge

Submitted:
Munich, 26th of November, 2024

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 26th of November, 2024 Giacomo Lupo

Zusammenfassung

Diese Arbeit untersucht verschiedene Varianten der Dynamischen Modenzerlegung (Dynamic Mode Decomposition, DMD) und ihre Anwendung auf das Lorenz-System, ein Paradebeispiel für chaotische Dynamik. DMD, eine datengetriebene Methode, basiert auf der Koopman-Operator-Theorie und ermöglicht die spektrale Analyse nichtlinearer dynamischer Systeme. Trotz ihrer Vielseitigkeit hat die Standard-DMD-Methode Einschränkungen, wie die Empfindlichkeit gegenüber Rauschen und Herausforderungen bei der Analyse hochdimensionaler Daten, die durch die Entwicklung spezialisierter Varianten adressiert wurden.

Im Verlauf der Arbeit werden sechs DMD-Varianten detailliert hergeleitet, analysiert und miteinander verglichen: Low-Rank DMD (lrDMD), Sparse DMD (spDMD), Optimized DMD (OptDMD), Extended DMD (EDMD), measure-preserving EDMD (mpEDMD) mit QR-Faktorisierung und EDMD mit Dictionary Learning (EDMD-DL). Fast jede Variante wurde auf das Lorenz-System angewandt, um ihre Stärken und Schwächen zu untersuchen und deren Eignung für die Analyse chaotischer Systeme zu evaluieren.

Zu den wichtigsten Ergebnissen gehören neuartige Visualisierungen der Eigenfunktionen des Koopman-Operators mittels EDMD mit Thin-Plate Splines sowie eine umfassende spektrale Analyse des Lorenz-Systems unter Verwendung von mpEDMD. Die formalisierte Herleitung von mpEDMD mit QR-Faktorisierung stellt einen weiteren wichtigen Beitrag dar, indem sie eine literarische Lücke für die Verwendung dieser robusteren Methode füllt. Besonders hervorzuheben ist die erstmalige Anwendung von EDMD-DL auf das Lorenz-System, die das Potenzial von Machine-Learning-Methoden zur Verbesserung der DMD-Analyse auf chaotischen Systemen aufzeigt. Diese Ergebnisse tragen nicht nur zum besseren Verständnis des Lorenz-Systems bei, sondern bieten auch eine solide Grundlage für die zukünftige Anwendung von DMD-Methoden auf komplexe dynamische Systeme.

Abschließend zeigt diese Arbeit, dass die Kombination aus DMD-Varianten und modernen numerischen Ansätzen eine leistungsstarke Methodik zur Untersuchung nichtlinearer Dynamiken darstellt. Gleichzeitig wird die Bedeutung klarer Herleitungen und didaktischer Präsentation betont, um den Zugang zu diesen Methoden für zukünftige Forschungen zu erleichtern.

Contents

1 Introduction

Historical background

The Koopman operator, discovered by B.O. Koopman in 1931 [7], is a linear operator that enables the study of nonlinear dynamical systems through the linear evolution of observables. While the idea of linearizing nonlinear processes is highly appealing, the infinite-dimensional nature of the Koopman operator and the need for knowledge of the system's governing functions historically made its application challenging. As a result, Koopman's spectral analysis remained largely dormant until the pioneering works of Rowley (2009) [33] and Schmid (2010), who developed Dynamic Mode Decomposition (DMD) [36] as a practical tool for approximating the Koopman operator. By connecting DMD with the Arnoldi approach and Koopman's theory, they established DMD as a data-driven technique for decomposing complex dynamical behaviors into coherent spatiotemporal modes. Since then, DMD has gained popularity for its ability to analyze high-dimensional systems and extract dominant patterns without requiring a full mathematical model of the underlying dynamics. This versatility has led to successful applications across fields such as fluid mechanics, where DMD helps identifying coherent flow structures; meteorology, where it captures and predicts atmospheric patterns, as exemplified by its application to the Lorenz system; neuroscience, where it analyzes brain dynamics; and epidemiology, where it aids in modeling the spread of infectious diseases. DMD's adaptability to different types of data has made it an essential tool for uncovering structure in complex systems. Despite its effectiveness, DMD has certain limitations. It is notably susceptible to noise, which can distort the accuracy of the extracted eigenvalues, and suffers from the curse of dimensionality when handling high-dimensional snapshot data, often resulting in spectral pollution. Additionally, DMD's reliance on linear observables limits its ability to capture complex nonlinear dynamics in many systems. Such challenges have led to the development of various DMD variants designed to improve robustness, computational efficiency, and adaptability to nonlinear features. In this thesis, we derive the foundational Koopman spectral analysis and Dynamic Mode Decomposition (DMD), establishing their mutual connection. We then introduce six DMD variants, discussing their qualities, limitations, and applications to the Lorenz system in addressing the challenges mentioned above.

Structure of the thesis

The thesis organized into seven chapters, each focusing on a different aspect of the Koopman operator, Dynamic Mode Decomposition (DMD), and its various variants. In Chapter 1, we introduce the fundamental concepts of Koopman spectral analysis, providing two analytical examples: one for a linear dynamical system and another for periodic sequences. These examples illustrate the application of the Koopman operator in simple settings, setting the stage for more complex developments in later chapters.

Chapter 2 delves into the derivation of DMD using both the Arnoldi approach and Schmid's method. We discuss the relationship between the Arnoldi method and Koopman's spectral analysis and establish the equivalency between Arnoldi and Schmid's formulations. Additionally, we introduce Exact DMD to set the groundwork for the following chapter on Extended DMD (EDMD).

In Chapter 3, we focus on two DMD variants, Low-rank DMD (lrDMD) [20] and Sparse DMD (SpDMD) [21], designed to address the curse of dimensionality by seeking a smaller number of dynamic modes. These methods aim at improving computational efficiency while preserving the most significant dynamic behaviours of the system.

Chapter 4 introduces Optimized DMD (OptDMD) [3], a variant developed to mitigate DMD's sensitivity to noise. The optimization techniques incorporated in this method help improving the robustness and accuracy of mode extraction in noisy datasets.

In Chapter 5, we derive Extended DMD (EDMD) [43] and kernelized EDMD [44], exploring their connection with Exact DMD. We also discuss different choices for constructing dictionaries in the context of kernelized EDMD, which enhances its ability to approximate the Koopman operator in more complex systems.

Chapter 6 addresses Measure-Preserving EDMD (mpEDMD) [11] and its QR version, which are tailored for analyzing continuous spectra of measure-preserving dynamical systems. This chapter emphasizes the convergence of these variants to spectral measures, making them a powerful tool for studying continuous dynamical behaviours.

Finally, Chapter 7 explores EDMD with Dictionary Learning (EDMD-DL) [26], a variant that utilizes neural networks to parametrize the dictionary. This method aims at discovering nearly Koopman-invariant dictionaries, improving the predictive power and accuracy of the analysis.

Contributions

Particularly noteworthy are the following results: the application of EDMD with thin-plate splines to the Lorenz system provides novel visualizations of its eigenfunctions (see Example 6.3.2); the analysis of the continuos spectra of the Lorenz system via mpEDMD (see Example 7.4.1) of Colbrook in [10] has been enhanced by directly linking it to the theory of the Lorenz system; the formalized derivation of the QR-factorization (see Subsection 7.1.2) for mpEDMD fills a gap in the literature, providing a rigorous framework for studying measure-preserving systems; additionally, the pioneering application of EDMD with Dictionary Learning (EDMD-DL) to the Lorenz system (see Example 8.3.1) requires careful parameter tuning, demonstrating the strength of EDMD-DL and offering valuable insights into the dynamics of chaotic systems. These results highlight the utility of DMD-based methods in analyzing complex, high-dimensional systems and offer promising directions for future research in various scientific fields. Moreover, this thesis can serve as a learning tool for future researchers, with the derivations of the algorithms carefully explained step-by-step. By providing detailed and accessible explanations, it aims to facilitate the understanding and application of DMD and its variants in a variety of contexts, encouraging further exploration of these methods.

2 Koopman Spectral Analysis

The Koopman operator, discovered and analyzed in 1931 by B.O. Koopman in [7], is an infinite-dimensional operator, whose relevance raised with the pioneering work of P.J. Schmid and Sesterhenn in [35] by developing the dynamic mode decomposition algorithm. In the following section we explore the principles of Koopman spectral analysis (following the work of C.W. Rowling et al. in [33] and P.J. Schmid in [36]) to outline the relation with the DMD algorithm in the next chapter.

2.1 Theory

Setting: Let $\mathcal{M} \subset \mathbb{R}^n$ be an n-dimensional manifold and $f : \mathcal{M} \to \mathcal{M}$ be a discrete nonlinear dynamical system such that for any initial condition $x_0 \in M$, the temporal trajectory is given by

$$\begin{aligned} x_{k+1} &= f(x_k) \quad \forall k \in \mathbb{N}_0 \\ \implies x_k &= f^k(x_0) \ \forall k \in \mathbb{N}. \end{aligned} \tag{2.1}$$

Definition 1 (Koopman Operator) *Let the dynamical system from (2.1) and the operator $\mathcal{K}$ act on all scalar-valued functions $\phi : M \to \mathbb{C}$ (also called observables) in the following manner:*

$$\mathcal{K}\phi(x) = \phi(f(x)) \tag{2.2}$$

*This operator is called **Koopman operator**. It practically allows us to predict the next observables in the sense of the next remark, this is only possible however, if the equations of the dynamical system (2.1) are known.*

Remark (observables): We can imagine these scalar-valued functions $\phi : M \to \mathbb{C}$ as measurements. If one has grid points in a flow field, the observables could be the velocity components at those grid points. Applying the Koopman operator to any of these velocity components at one of the grid points will give us the velocity component in the next time step at that given grid point.

Linearity: The Koopman operator $\mathcal{K}$ is *linear*, since for any two observables $\theta_1, \theta_2 : M \to \mathbb{C}$ and any two scalars $\alpha, \beta \in \mathbb{C}$ the following holds:

$$\mathcal{K}(\alpha\theta_1 + \beta\theta_2)(x) = (\alpha\theta_1 + \beta\theta_2)(f(x)) = \alpha\theta_1(f(x)) + \beta\theta_2(f(x)) = \alpha\mathcal{K}\theta_1(x) + \beta\mathcal{K}\theta_2(x) \tag{2.3}$$

This property is extremely important: while the dynamical system (2.1) is nonlinear, making it very hard to analyze, the linearity of the Koopman operator allows a spectral analysis. This benefit comes however with a cost: while the dynamical system evolves on the finite dimensional manifold $\mathcal{M}$; the Koopman operator acts on the space of all observables $\mathcal{F}$, which is infinite dimensional.

Definition 2 (Koopman eigenfunctions, eigenvalues and modes) *Because of the linearity of the Koopman operator $\mathcal{K}$, we can assume there exist eigenfunctions $\phi_i : M \to \mathbb{R}$ with associated eigenvalues $\lambda_i \in \mathbb{C}$, such that it holds:*

$$\mathcal{K}\phi_i(x) = \lambda_i\phi_i(x) \ \ \forall \ i \in \mathbb{N} \tag{2.4}$$

as mentioned above, $\mathcal{K}$ is infinite dimensional, the number of eigenfunctions and eigenvalues can hence be infinite (and this is mostly the case!), we will however see one example where we can find all eigenvalues and eigenfunctions, hence being able to analytically decompose the Koopman operator.

Let $x \in M$ be a a full-state observable about a flow field at a particular time and $g : M \to \mathbb{C}^p$ a p-dimensional vector of observables $g_i : M \to \mathbb{C}\ \forall i \in \{1, 2, ..., p\}$ and suppose that each component lies in span$\{(\phi_i)_{i\in\mathbb{N}}\}$, then we can expand these observables in the following manner:

$$g_i(x) = \sum_{j=1}^{\infty} \phi_j(x) \cdot v_{j_i}$$

such that $v_{j_i} \in \mathbb{C}\ \forall j \in \mathbb{N}\ \forall i \in \{1, 2, ..., p\}$.

If we then define the vectors $v_j = (v_{j_1}, v_{j_2}, ...v_{j_p})^\top \in \mathbb{C}^p$, we can also expand the vector valued function $g(x)$:

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_p(x) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{\infty} \phi_j(x) \cdot v_{j_1} \\ \sum_{j=1}^{\infty} \phi_j(x) \cdot v_{j_2} \\ \vdots \\ \sum_{j=1}^{\infty} \phi_j(x) \cdot v_{j_p} \end{bmatrix} = \sum_{j=1}^{\infty} \phi_j(x) \cdot \begin{bmatrix} v_{j_1} \\ v_{j_2} \\ \vdots \\ v_{j_p} \end{bmatrix} = \sum_{j=1}^{\infty} \phi_j(x) \cdot v_j \tag{2.5}$$

Using this expansion, we are hence able to apply the Koopman operator on g, recalling (2.3) and (2.4) and starting at the initial state x_0, it is easy to see that

$$\mathcal{K}^k g(x_0) = g(x_k) = \sum_{j=1}^{\infty} \lambda_j^k \phi_j(x) \cdot v_j \tag{2.6}$$

*The eigenfunctions $\phi_i : M \to \mathbb{C}$ are then called **Koopman eigenfunctions**, the vectors $v_i \in \mathbb{C}^p$ are called **Koopman modes** and the eigenvalues $\lambda_i \in \mathbb{C}$ are called **Koopman eigenvalues**.*

Remark: From expansion (2.5), it is easy to recognize that the eigenvalues $\lambda_j \in \mathbb{R}$ characterize the temporal behaviour of the Koopman modes. Indeed, taking the polar representation of the eigenvalues $\lambda_j = R_j \cdot e^{i\theta_j}$, the magnitude R_j does represent the growth rate of the Koopman mode v_j, while the phase θ_j represent its frequency.

2.2 Examples

2.2.1 Linear Dynamical Systems

Consider a matrix $A \in \mathbb{C}^{n\times n}$ and the linear dynamical system (for an arbitrary initial state $x_0 \in \mathbb{C}^n$)

$$x_{k+1} = Ax_k\ \forall k \in \mathbb{N}_0$$

C.W. Rowley et al. [33] and S. Klus et al. [38] both provide eigenfunctions of the Koopman operator for this simple case, we start with the eigenvalues provided in [38] and show the equivalence to the ones provided in [33].

To this end, we consider an eigenvalue λ_j of the matrix A and its associated left eigenvector $w_j \in \mathbb{C}^{1\times n}$, hence satisfying the equation:

$$w_j A = \lambda_j w_j \tag{2.7}$$

now we consider the observable $\phi_j(x) = w_j x$, this is a Koopman eigenfunction, taking into account the equation above and applying the Koopman operator:

$$\mathcal{K}\phi_j(x) = \phi_j(Ax) = w_j Ax = \lambda_j w_j x = \lambda_j \phi_j(x)$$

We can hence see that, in the linear case, the eigenvalues of A are also Koopman eigenvalues and can establish the relation between the left eigenvectors and the eigenfunctions of the Koopman operator.

It is however important to notice that there are infinitely many eigenfunctions of the Koopman operator, since two product of two eigenfunctions is an eigenfunction too: let $\lambda_i,\ \lambda_j \in \mathbb{C}$ be any two eigenvalues

of A; $\phi_i, \phi_j : \mathbb{C}^n \to \mathbb{C}$ be their associated eigenfunctions and take the new function $\phi_{ij}(x) = \phi_i(x)\phi_j(x)$, applying the Koopman operator, there holds

$$\mathcal{K}\phi_{ij}(x) = \phi_{ij}(Ax) = \phi_i(Ax)\phi_j(Ax) = \lambda_i\phi_i(x)\lambda_j\phi_j(x) = \lambda_i\lambda_j\phi_i(x)\phi_j(x) = \lambda_i\lambda_j\phi_{ij}(x) \tag{2.8}$$

so the function $\phi_{ij} : \mathbb{C}^n \to \mathbb{C}$ is an eigenfunction of the Koopman operator with the associated Koopman eigenvalue $\lambda_i\lambda_j$.

From this, it easy to prove that, for any multiindex $a = (a_1, a_2, ..., a_n) \in \mathbb{N}_0^n$ the function $\phi_a(x) = \prod_{i=1}^n \phi_i(x)^{a_i}$ is a Koopman eigenfunction with $\prod_{i=1}^n \lambda_i^{a_i}$ as associated Koopman eigenvalue.

We now establish the equivalence with the solution provided in [33]: applying the transpose conjugate operator $*$ to (2.7), we obtain the equation

$$A^* w_j^* = \overline{\lambda_j} w_j^*,$$

we can hence see that the transpose conjugate of the left eigenvector w_j is a right eigenvector of the adjoint matrix A; we now recall the definition of the standard inner product of $\mathbb{C}^n$, given by $\langle x, y\rangle = x^\intercal \overline{y}$ to see that:

$$\begin{aligned}\phi_j(x) &= w_j x \\ &= x^\intercal w_j^\intercal \\ &= \langle x, w_j^*\rangle\end{aligned}$$

which is exactly the result provided by C.J. Rowley et al. in [33].

If we define the observables $g_i(x) = x_i \ \forall i \in \{1, 2, ...n\}$, where $x_i \in \mathbb{C}$ represents the i-th component of the full-state vector $x \in \mathbb{C}^n$ and the matrix A has full rank n (and hence a full set of right eigenvalues $(v_1, v_2, ..., v_n)$), then we can expand the full state vector-valued function $g(x) = (g_1(x), ..., g_n(x))^\intercal$ via (2.5) as

$$g(x) = x = \sum_{i=1}^{n} \phi_i(x) \cdot v_i \tag{2.9}$$

Hence, if A has full rank, the Koopman modes of the full-state observable coincide with the right eigenvectors of A.

Visualization of an example

We consider the matrix $A \in \mathbb{R}^{2\times 2}$ given by

$$A = \begin{bmatrix} 0.17 & 0.36 \\ -0.24 & 0.83 \end{bmatrix}$$

this matrix has eigenvalues $\lambda_1 = 0.35$ and $\lambda_2 = 0.65$ with their associated normalized left eigenvectors $v_1 = [-\frac{4}{5}, \frac{3}{5}]$ and $v_2 = [\frac{1}{\sqrt{5}}, -\frac{2}{\sqrt{5}}]$, the two functions $\psi_1(x) = v_1 x$ and $\psi_2(x) = v_2 x$ are hence the two basic Koopman eigenfunctions. As shown in (2.8), the product of any two Koopman eigenfunctions ϕ and ψ with Koopman eigenvalues λ and μ is a Koopman eigenfunction with Koopman eigenvalue $\lambda\mu$, the first 8 most impactful Koopman eigenfunction are visualized in Figure 2.1.

2.2.2 Periodic Trajectories

We now consider the particular case of one solution of (2.1) being given by a periodic sequence $S = \{x_0, x_1, ..., x_{m-1}\}$ such that $x_{m+k} = x_k \in \mathbb{C}^n \ \forall k \in \mathbb{N}_0$.

The most intuitive way to analyze such a solution is to take the Discrete Fourier Transform, defining new vectors $\{\hat{x}_0, \hat{x}_1, ..., \hat{x}_{m-1}\}$ which satisfy the equation

$$x_k = \sum_{j=0}^{m-1} e^{\frac{i2\pi jk}{m}} \cdot \hat{x}_j \ \forall k \in \{0, 1, ..., m-1\} \tag{2.10}$$

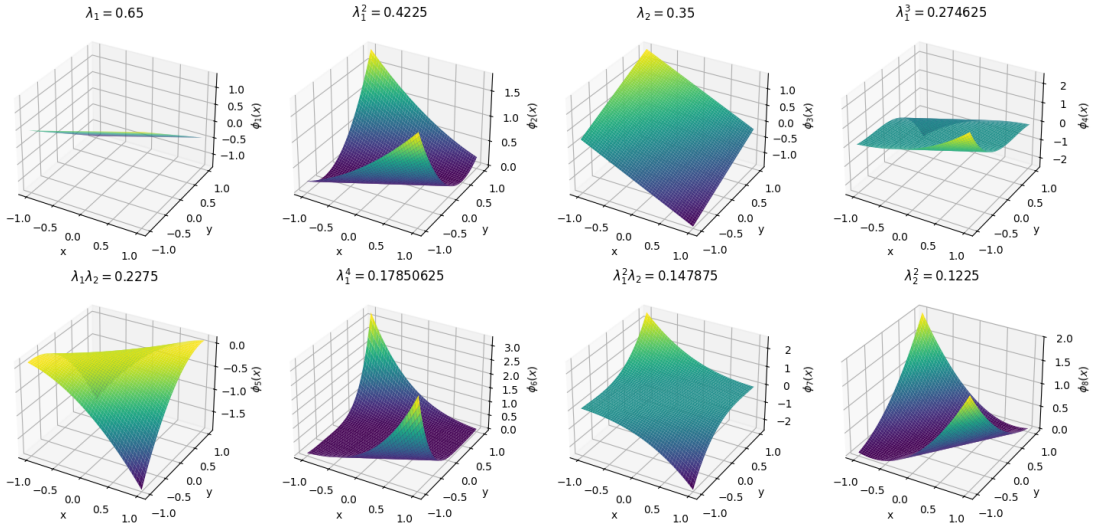

Figure 2.1 Eigenfunctions of the linear dynamical system

Now we define the functions $\phi_j(x_k) = e^{\frac{i2\pi jk}{m}} \ \forall j,k \in \{0,1,...,m-1\}$ and apply the Koopman operator

$$\mathcal{K}\phi_j(x_k) = \phi_j(f(x_k)) = \phi_j(x_{k+1}) = e^{\frac{i2\pi j(k+1)}{m}} = e^{\frac{i2\pi j}{m}} e^{\frac{i2\pi jk}{m}} = e^{\frac{i2\pi j}{m}} \phi(x_k)$$

The function ϕ_j is hence a Koopman eigenfunction with the associated eigenvalue $\lambda_j = e^{\frac{i2\pi j}{m}}$, we can hence rewrite (2.10) as

$$x_k = \sum_{j=0}^{m-1} \phi_j(x_k)\hat{x}_j$$

We notice the similarity of the equation above with the expansions (2.5) and (2.9), indeed if we restrict the phase space of the Koopman eigenvalues to the trajectory S, then all the Koopman eigenvalues $\lambda_j = e^{\frac{i2\pi j}{m}}$ are correctly identified and the Koopman modes of the full-state of the orbit S coincide with the vectors $\{\hat{x}_0, \hat{x}_1, ..., \hat{x}_{m-1}\}$ given by the Discrete Fourier Transform.

Remark: This holds of course only restricting the space to the trajectory S, not for the whole dynamical system (2.1)!

3 Dynamic Mode Decomposition (DMD)

As we have seen in the last chapter, the linearity of the Koopman operator allows the very desirable expansions (2.5) and (2.6), however the infinite-dimensionality of the Koopman operator does represent a big obstacle and to apply the operator one needs to have prior knowledge of the underlying function f in (2.1). To this end, P.J. Schmid and J. Sesterhenn developed the dynamical mode decomposition (DMD) algorithm in [35], which allows to numerically approximate the most important Koopman modes and eigenvalues without having prior knowledge of the ruling function f, effectively allowing us to make predictions of measurements for the next time step with a certain degree of accuracy.

3.1 Arnoldi approach

In this first section, still following the work of Rowley in [33], we derive the DMD algorithm for the Arnoldi approach (described by Y. Saad in [34]) and the singular value decomposition approach (provided by P.J. Schmid and J. Sesterhenn in [35]) and show its connections to the Koopman analysis of the previous chapter.

3.1.1 Derivation of the algorithm

The basic idea of DMD is the following: starting at some initial condition $x_0 \in \mathbb{R}^n$ we first collect m iterations developed by the dynamical system (2.1), such that we have a sequence of snapshots $\{x_0, x_1, ..., x_m\}$, we then *assume* that the snapshots are linearly dependent, i.e. we assume the existence of a matrix $A \in \mathbb{R}^{n\times n}$, such that the linear dynamical system:

$$x_{k+1} = Ax_k, \ \forall k \in \{0, ..., m-1\} \tag{3.1}$$

$$x_{k+1} \approx Ax_k, \ \forall k \geq m \tag{3.2}$$

can approximate the true dynamical system (2.1) at best. As we have seen in Example 2.2.1, in linear dynamical systems the eigenvalues of the matrix A are also Koopman eigenvalues, hence our goal is to obtain an eigendecomposition of A to approximate the Koopman modes. Because of the high dimensionality n of the snapshot x_i, it is not easy (and sometimes not even possible!) to compute the whole matrix A, instead we try to compute an lower dimensional companion matrix $C \in \mathbb{R}^{m\times m}$ which is related to the matrix A, stacking the snapshots in the two matrices (where the lower index is the index of the first snapshot and the upper index is the index of the last snapshot of the matrix)

$$\Psi_0^{m-1} = [x_0, x_1, x_2, ..., x_{m-1}] \in \mathbb{R}^{n\times m}$$
$$\Psi_1^{m} = [x_1, x_2, x_3, ..., x_m] \in \mathbb{R}^{n\times m}$$

and taking into account assumption (3.2) we see that

$$\Psi_1^m = [x_1, x_2, x_3, ..., x_m] \approx [Ax_0, Ax_1, Ax_2, ..., Ax_{m-1}] = A\Psi_0^{m-1}$$

the matrix $C \in \mathbb{R}^{m\times m}$ we are searching for must hence satisfy

$$\Psi_1^m \approx A\Psi_0^{m-1} \approx \Psi_0^{m-1}C \tag{3.3}$$

This dimensionality reduction is mathematically and physically explained by the fact that the amount of the major impactful forces (the dynamical modes) ruling the dynamical system will be very low compared to the dimension of one snapshot, it is hence fair to assume that, taking enough snapshots, we will be able to represent the next iteration as a linear representation of the previous iterations, such that assumption (3.2) is justified.

Special case: linear dependence of the m-th iteration

We first consider the special case where the m-th iterate x_m is spanned by the vectors $\{x_0, Ax_0, ..., A^{m-1}x_0\}$, i. e. there exists a vector $c = (c_0, c_2, ..., c_{m-1})^\intercal \in \mathbb{R}^m$ such that

$$x_m = Ax_{m-1} = c_0x_0 + c_1x_1 + ... + c_{m-1}x_{m-1} = \Psi_0^{m-1}c \tag{3.4}$$

then it holds, for $e_j \in \mathbb{R}^m$ being the jth unit vector,

$$\begin{aligned} \Psi_1^m = A\Psi_0^{m-1} &= [x_1, x_2, ..., x_m] \\ &= [\Psi_0^{m-1}e_2, \Psi_0^{m-1}e_3, ..., \Psi_0^{m-1}c] \\ &= \Psi_0^{m-1}[e_2, e_3, ..., c] = \Psi_0^{m-1}C \\ \implies \Psi_1^m = A\Psi_0^{m-1} &= \Psi_0^{m-1}C, \end{aligned} \tag{3.5}$$

where C is the companion matrix of the following form:

$$C = [e_2, e_3, ..., c] = \begin{bmatrix} 0 & 0 & \dots & 0 & c_0 \\ 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_{m-1} \end{bmatrix} \in \mathbb{R}^{m\times m} \tag{3.6}$$

Notice that, as long as $c_0 \neq 0$, the matrix C has full rank m, otherwise it has rank m-1.

Connection to Koopman modes and values: We now show that, in this special case, the eigenvalues of C are a subset of eigenvalues of A and that the eigenvectors of C are related to those of A, to this end let $a \in \mathbb{R}^m$ be an eigenvector of C with the associated eigenvalue λ, hence

$$Ca = \lambda a$$

We then recall equation (3.5) and multiply with a from the right to see that

$$A\Psi_0^{m-1}a = \Psi_0^{m-1}Ca = \Psi_0^{m-1}\lambda a = \lambda\Psi_0^{m-1}a$$

The vector $v = \Psi_0^{m-1}a$ is then an eigenvector of A with the associated eigenvalue λ, hence confirming that the eigenvalues of C are also eigenvalues of A and establishing the linear relation of their eigenvectors.

Generalization to residual equations

Now we can generalize for the case that x_m is not in $\text{span}\{x_0, x_1, ..., x_{m-1}\}$, equation (3.4) doesn't hold anymore and there exists a residual $r \in \mathbb{R}^m$ such that

$$x_m = Ax_{m-1} = c_0x_0 + c_1x_1 + ... + c_{m-1}x_{m-1} + r = \Psi_0^{m-1}c + r$$

and hence (3.5) transforms into the equation

$$\Psi_1^m = A\Psi_0^{m-1} = \Psi_0^{m-1}C + re_m^\intercal \tag{3.7}$$

In this case the eigenvalues and eigenvectors of C do not coincide with the ones of A, but rather approximate them.

Definition 3 (Ritz values and vectors) *The eigenvalues of the companion matrix C are called* ***Ritz values*** *while the eigenvectors are called* ***Ritz vectors****.*

Remark (minimizing the residual r): To assure that the residual r is minimal, the vector $c \in \mathbb{R}^m$ has to be a solution of the optimal problem

$$\min_{c \in \mathbb{R}^m} J(c) = \min_{c \in \mathbb{R}^m} \|x_m - \Psi_0^{m-1} c\|_2^2$$

This problem is easily solvable by transforming the objective with some simple algebraic steps

$$\begin{aligned} J(c) &= \|x_m - \Psi_0^{m-1} c\|_2^2 \\ &= (x_m - \Psi_0^{m-1} c)^\top (x_m - \Psi_0^{m-1} c) \\ &= x_m^\top x_m - c^\top \Psi_0^{m-1\top} x_m - x_m^\top \Psi_0^{m-1} c + c^\top \Psi_0^{m-1\top} \Psi_0^{m-1} c \\ &= x_m^\top x_m - (x_m^\top \Psi_0^{m-1} c)^\top - x_m^\top \Psi_0^{m-1} c + c^\top \Psi_0^{m-1\top} \Psi_0^{m-1} c \\ &= x_m^\top x_m - 2 x_m^\top \Psi_0^{m-1} c + c^\top \Psi_0^{m-1\top} \Psi_0^{m-1} c \end{aligned}$$

We can then take the gradient and, setting it equal to zero, find the solution

$$\begin{aligned} &\nabla J(c) = 2\Psi_0^{m-1\top} x_m + 2\Psi_0^{m-1\top} \Psi_0^{m-1} c = 0 \\ &\implies c = (\Psi_0^{m-1\top} \Psi_0^{m-1})^\dagger \Psi_0^{m-1\top} x_m \end{aligned} \tag{3.8}$$

where $(\Psi_0^{m-1\top} \Psi_0^{m-1})^\dagger$ is the Moore-Penrose inverse of the matrix $\Psi_0^{m-1\top} \Psi_0^{m-1}$.

3.1.2 Algorithm of the Arnoldi approach

Here we provide the algorithm of the Arnoldi approach to find the empirical Ritz values and Ritz vectors from a sequence $\{x_0, x_1, ..., x_m\}$:

Step 1: Stack the first m snapshots in the matrix $\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}]$

Step 2: Find the optimal vector $c = (\Psi_0^{m-1\top} \Psi_0^{m-1})^\dagger \Psi_0^{m-1\top} x_m$

Step 3: Assemble the companion matrix

$$C = [e_2, e_3, ..., e_m, c] = \begin{bmatrix} 0 & 0 & \dots & 0 & c_0 \\ 1 & 0 & \dots & 0 & c_1 \\ 0 & 1 & \dots & 0 & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & c_{m-1} \end{bmatrix}$$

Step 4: Compute the eigendecomposition of C, for T containing the eigenvectors of C as columns and Λ being the diagonal matrix of the eigenvalues of C, we get the equation

$$\begin{aligned} &CT = T\Lambda \\ \Leftrightarrow\ &C = T\Lambda T^{-1} \end{aligned} \tag{3.9}$$

Step 5: The Ritz vectors are then the columns of the matrix KT and the Ritz values are the diagonal components of the matrix Λ

3.1.3 Modal reconstruction and connection to Koopman

Theorem 4 *Consider a sequence $\{x_0, x_1, ..., x_m\}$ and let λ_j, v_j be the empirical Ritz values and vectors, assuming that the Ritz values are distinct, then*

$$x_k = \sum_{i=1}^{m} \lambda_i^k v_i \ \forall k \in \{0, 1, ..., m-1\} \tag{3.10}$$

$$x_m = \sum_{i=1}^{m} \lambda_i^m v_i + r \tag{3.11}$$

Proof: We first notice that (3.10) can be rewritten in matrix form as

$$\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}] = [v_1, v_2, \dots, v_m] \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^m \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^m \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_m & \lambda_m^2 & \dots & \lambda_m^m \end{bmatrix} = V V_{and}^{\lambda} \tag{3.12}$$

Now it is important to notice that the Vandermonde matrix V_{and} is deeply related with the companion matrix C, indeed V_{and}^{λ} diagonalizes C (because the eigenvalues λ_i are distinct) and hence is the inverse matrix T^{-1} from equation (3.9). Recalling that $V = KT$, equation (3.12) follows automatically, lastly equation (3.11) follows from (3.7), since

$$AK = [x_1, x_2, ..., x_m] = KC + re_m^{\top} = KT\Lambda T^{-1} + re_m^{\top} = V\Lambda V_{and}^{\lambda} + re_m^{\top}$$

□

Connection to the Koopman operator Consider a vector-valued function of observables $g : M \to \mathbb{R}$ for the dynamical system (2.1), if we have a temporal sequence of observables $\{g(x_0), g(x_1), \dots, g(x_m)\}$ we can use the Arnoldi algorithm from the previous subsection to obtain its Ritz values $\tilde{\lambda}_i$ and vectors $\tilde{v}_i$ and, applying Theorem 4, it holds

$$g(x_k) = \sum_{i=1}^{m} \tilde{\lambda}_i^{\,k} \tilde{v}_i \ \forall k \in \{0, 1, ..., m-1\}$$

$$g(x_m) = \sum_{i=1}^{m} \tilde{\lambda}_i^{\,m} \tilde{v}_i + r$$

The similarity with the Koopman expansion (2.6) is astonishing: the Ritz values and vectors rule the temporal behaviour of the sequence of snapshots in a very similar manner to the Koopman eigenvalues and modes. The obvious difference is that this approach gives us a finite sum and the mth iterate x_m has some error given by the term r, in this sense we get the best approximation via the orthogonal projection of the matrix Ψ_0^{m-1} given by (3.8).

Example: Periodic sequences

Let us have a sequence of snapshots $\{x_0, x_1, ..., x_m\}$ such that, for a vector-valued function of observables $g : M \to \mathbb{R}$, it holds $g(x_m) = g(x_0)$, then we have a special case of linear dependence where the vector $c = (c_0, c_2, ..., c_{m-1}) = (1, 0, ..., 0)$ satisfies equation (3.4), the companion matrix C is then given by

$$C = [e_2, e_3, ..., e_{m-1}, e_1] = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \in \mathbb{R}^{m \times m}$$

It is then easy to prove that the eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_m\}$ of this matrix C are then given by the mth roots of 1, hence $\lambda_j = e^{\frac{i2\pi j}{m}} \ \forall j \in \{1, 2, ..., m\}$, in accordance with Example 2.2.2.

3.2 SVD-Approach

3.2.1 Derivation of the algorithm

We again assume that the snapshots are generated by a linear dynamical system as in equation (3.2), using the same notation for the matrices of stacked snapshots, it holds:

$$\begin{aligned} & \Psi_1^m = [x_1, x_2, ..., x_m] = [Ax_0, Ax_1, ..., Ax_{m-1}] = A[x_0, x_1, ..., x_{m-1}] = A\Psi_0^{m-1} + re_m^\top \\ \implies & \Psi_1^m = A\Psi_0^{m-1} \end{aligned} \tag{3.13}$$

Now, instead of directly computing A, we first compute the Singular Value Decomposition (SVD) of the matrix $\Psi_0^{m-1} \in \mathbb{R}^{n\times m}$,

$$\Psi_0^{m-1} = U\Sigma V^* \tag{3.14}$$

such that

$$\begin{aligned} & U \in \mathbb{C}^{n\times n} \ s.t. \ U^*U = UU^* = I, \\ & V \in \mathbb{C}^{m\times m} \ s.t. \ V^*V = VV^* = I, \\ & \Sigma = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{C}^{n\times m}, \end{aligned}$$

where the scalars $\lambda_j \in \mathbb{C} \ \forall j \in \{1, 2, ..., m\}$ are the singular values of the matrix Ψ_0^{m-1}. Applying (3.14) on (3.13) and exploiting the fact that U and V are unitary, we obtain the companion matrix A_{dmd}:

$$\begin{aligned} & AU\Sigma V^* = \Psi_1^m \\ \implies & AU = \Psi_1^m V\Sigma^{-1} \\ \implies & A_{dmd} = U^*AU = U^*\Psi_1^m V\Sigma^{-1} \end{aligned} \tag{3.15}$$

Definition 5 (DMD modes and values) *The matrix $A_{dmd} = U^*AU$ is related to the matrix A via a similarity transform: let λ and y be given such that $A_{dmd}y = \lambda y$, then for the vector $\phi = Uy$, it holds*

$$A\phi = UA_{dmd}U^*Uy = UA_{dmd}y = U\lambda y = \lambda Uy = \lambda\phi.$$

*The vector ϕ is hence an eigenvector of A with the associated eigenvalue λ, ϕ is called **dmd mode** and λ is called **dmd value**.*

3.2.2 Connection between Arnoldi and SVD approach

We now establish the connection between the two approaches, following the work of M. R. Jovanović et al. in [20]: to this end, we recall equation (3.7)

$$\Psi_1^m = A\Psi_0^{m-1} = \Psi_0^{m-1}C + re_m^\top$$

By this representation we can think of C as the projection of the matrix A on an m-dimensional subspace of $\mathbb{C}^n$: instead of directly computing C, we can establish it by minimizing the Frobenius norm of the residual matrix, hence finding the solution of the optimization problem

$$\min_{C\in\mathbb{C}^{m\times m}} \|\Psi_1^m - \Psi_0^{m-1}C\|_F^2 \tag{3.16}$$

Where we recall that the Frobenius norm of a matrix Q is determined by:

$$\|Q\|_F^2 = trace(Q^*Q) = trace(QQ^*) \tag{3.17}$$

we now take the economy-sized SVD of the matrix Ψ_0^{m-1}, such that:

$$\begin{aligned}
&\Psi_0^{m-1} = U\Sigma V^* \\
&\text{where:} \\
&\quad U \in \mathbb{C}^{n\times m} \text{ such that } U^*U = I, \\
&\quad V \in \mathbb{C}^{m\times m} \text{ such that } V^*V = VV^* = I, \\
&\quad \Sigma = diag(\{\lambda_i\}) \in \mathbb{C}^{m\times m}.
\end{aligned}$$

Now we notice that, if a matrix U is unitary, then for the Frobenius norm of a matrix Q the following holds:

$$\|UQ\|_F^2 = trace((UQ)^*(UQ)) = trace(Q^*U^*UQ) = trace(Q^*Q) = \|Q\|_F^2. \tag{3.18}$$

Using (3.18) we can manipulate the optimization problem (3.16) in the following manner:

$$\begin{aligned}
\min_{C\in\mathbb{C}^{m\times m}} \|\Psi_1^m - \Psi_0^{m-1}C\|_F^2 &= \min_{C\in\mathbb{C}^{m\times m}} \|\Psi_1^m - U\Sigma V^*C\|_F^2 \\
&= \min_{C\in\mathbb{C}^{m\times m}} \|U^*\Psi_1^m - \Sigma V^*C\|_F^2 \\
&= \min_{C\in\mathbb{C}^{m\times m}} \|U^*\Psi_1^m V - \Sigma V^*CV\|_F^2 \\
&= \min_{C\in\mathbb{C}^{m\times m}} \|(U^*\Psi_1^m V\Sigma^{-1} - \Sigma V^*CV\Sigma^{-1})\Sigma\|_F^2
\end{aligned}$$

Defining the similarity transform of the companion matrix C as $F = \Sigma V^*CV\Sigma^{-1} = TCT^{-1}$ for $T = \Sigma V^*$, it is straightforward to see that the optimization problem (3.16) is equivalent to

$$\min_{F\in\mathbb{C}^{m\times m}} \|(U^*\Psi_1^m V\Sigma^{-1} - F)\Sigma\|_F^2. \tag{3.19}$$

So we can conclude that the optimal rank-m solution of (3.19) is given by the matrix

$$A_{dmd} = U^*\Psi_1^m V\Sigma^{-1}$$

which is exactly the result of P.J. Schmid in [36] given by equation (3.15), hence establishing the connection between the Arnoldi Approach and the SVD one. While the Arnoldi approach directly returns the matrix C, making it suitable to compare the results with the Koopman operator for analytical cases (such as the example of periodic solutions), the biggest advantage of the SVD approach is its robustness towards measurement noise in real life applications: as already mentioned, in most dynamical systems the number of relevant dynamic modes is lower than the dimension of one snapshot, there are thus eigenvalues which are not related to the dynamical system (this phenomenon is called spectral pollution). Using the SVD approach we are able to truncate the rank of the solution, hence avoiding underlying modes created by the noise and the high dimensionality of the snapshots.

Remark (Economy-sized SVD): It is possible to take the economy sized SVD instead of the complete one to find the optimal rank-m solution because of the Eckart–Young–Mirsky theorem holding for the Frobenius norm!

3.2.3 Modal Reconstruction and optimal amplitudes

For the identification of the optimal amplitudes and the subsequent reconstruction of the training data, we orient ourselves to the work of Jovanović in [21].

The companion matrix $A_{dmd} \in \mathbb{C}^{m\times m}$ is the optimal rank-m representation of the mapping $A \in \mathbb{C}^{n\times n}$ defined by the dynamical system (3.2) on the subspace created by the columns of Ψ_0^{m-1}; on such a subspace, the dynamics are governed by

$$z_{t+1} = A_{dmd} z_t \tag{3.20}$$

such that the for the m-dimensional vectors $z_t \ \forall t \in \mathbb{N}$ are related to the n-dimensional snapshots via the matrix U, i.e. $x_t \approx U z_t$.

Suppose that the matrix A_{dmd} has m linearly independent eigenvectors $\{y_1, y_2, ..., y_m\}$ with their associate eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_m\}$ then, for $Y = [y_1, y_2, ..., y_m]$ and $D_\mu = diag(\{\mu_i\}_{i\in[1,...,m]})$ the eigendecomposition holds:

$$A_{dmd} Y = Y D_\mu$$

because all columns of Y are linearly independent, Y is invertible, hence we can diagonalize F_{dmd} in the following manner:

$$A_{dmd} = Y D_\mu Y^{-1} = [y_1, y_2, ..., y_m] \begin{bmatrix} \lambda_1 & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & 0 \\ 0 & 0 & \lambda_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \lambda_m \end{bmatrix} \begin{bmatrix} \tilde{y}_1^* \\ \tilde{y}_2^* \\ \tilde{y}_3^* \\ \vdots \\ \tilde{y}_m^* \end{bmatrix} \tag{3.21}$$

where the vectors $\{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_m\}$ represent the eigenvectors of the matrix A_{dmd}^*.

Notice that, with the diagonalization formula (3.21), for the powers of A_{dmd} holds:

$$\begin{aligned} A_{dmd}^2 &= Y D_\mu Y^{-1} Y D_\mu Y^{-1} \\ &= Y D_\mu^2 Y^{-1}, \\ \implies A_{dmd}^{k+1} &= A_{dmd}^k A_{dmd} \\ &= Y D_\mu^k Y^{-1} Y D_\mu Y^{-1} \\ &= Y D_\mu^{k+1} Y^{-1}. \end{aligned}$$

Starting with any initial condition $z_0 \in \mathbb{C}^m$, the solution to (3.20) is determined by

$$z_t = A_{dmd}^t z_0 = Y D_\mu^t Y^{-1} z_0 = \sum_{k=1}^{m} y_k \lambda_k^t \tilde{y}_k^* z_0 = \sum_{k=1}^{m} y_k \lambda_k^t \alpha_k,$$

where $\alpha_k = \tilde{y}_k^* z_0$ is the i-th modal contribution of the initial condition z_0. We can hence reconstruct the high-dimensional snapshots $x_t \in \mathbb{C}^n$ as a linear representation of DMD modes $\phi_k = U y_k \ \forall k \in \{1, 2, ..., m\}$:

$$x_t \approx U z_t = U \sum_{k=1}^{m} y_k \lambda_k^t \alpha_k = \sum_{k=1}^{m} U y_k \lambda_k^t \alpha_k = \sum_{k=1}^{m} \phi_k \lambda_k^t \alpha_k, \ \ t \in \{0, 2, ..., m-1\} \tag{3.22}$$

Every scalar α_k can be hence thought of as the amplitude of the corresponding DMD mode ϕ_k. Using equation (3.22), we can reconstruct the matrix of stacked snapshots Ψ_0^{m-1}:

$$\Psi_0^{m-1} \approx \Phi D_\alpha V_{and}^\lambda = [\phi_0, \phi_1, ..., \phi_{m-1}] \begin{bmatrix} \alpha_1 & 0 & 0 & \dots & 0 \\ 0 & \alpha_2 & 0 & \dots & 0 \\ 0 & 0 & \alpha_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \alpha_m \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{m-1} \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_m & \lambda_m^2 & \dots & \lambda_m^{m-1} \end{bmatrix} \tag{3.23}$$

Where we can see that the Vandermonde matrix V_{and}^λ governs the temporal evolution in a very similar manner to equation (3.12) in the Arnoldi approach.

Determination of the optimal amplitudes

In order to find the optimal vector of amplitudes $\alpha = [\alpha_1, \alpha_2, ..., \alpha_m]$, we formulate the following optimization problem to minimize the residual:

$$\min_{\alpha \in \mathbb{C}^m} \|\Psi_0^{m-1} - \Phi D_\alpha V_{and}^\lambda\|_F^2 \tag{3.24}$$

We define the function $J(\alpha) = \|\Psi_0^{m-1} - \Phi D_\alpha V_{and}^\lambda\|_F^2$, using economy-sized SVD $\Psi_0^{m-1} = U\Sigma V^*$, the fact that U is unitary and recalling $\Phi = UY$, we transform the optimization problem (3.24) in the following manner:

$$\begin{aligned}
\min_{\alpha \in \mathbb{C}^m} J(\alpha) &= \min_{\alpha \in \mathbb{C}^m} \|U\Sigma V^* - UYD_\alpha V_{and}^\lambda\|_F^2 \\
&= \min_{\alpha \in \mathbb{C}^m} \|U^*U\Sigma V^* - U^*UYD_\alpha V_{and}^\lambda\|_F^2 \\
&= \min_{\alpha \in \mathbb{C}^m} \|\Sigma V^* - YD_\alpha V_{and}^\lambda\|_F^2
\end{aligned} \tag{3.25}$$

We now show that the optimization problem (3.25) is convex, we recall the following properties of the trace operator:

P1: The trace operator is cyclic:

$$trace(ABC) = trace(CAB) = trace(BCA)$$

P2: For the product of a matrix Q and a diagonal matrix $D_\alpha = diag\{\alpha\}$, it holds:

$$trace(QD_\alpha) = (diag\{Q\})^\intercal \alpha = (\overline{diag\{Q\}})^* \alpha$$

P3: For vectors $\alpha \in \mathbb{C}^n$ and $\beta \in \mathbb{C}^m$ and matrices $A, B \in \mathbb{C}^{m \times n}$, it holds:

$$trace(D_\beta^* A D_\alpha B^\intercal) = \beta^* (A \circ B)\alpha$$

Now, recalling (3.17), it holds:

$$\begin{aligned}
J(\alpha) =& \|\Sigma V^* - YD_\alpha V_{and}^\lambda\|_F^2 \\
=& trace((\Sigma V^* - YD_\alpha V_{and}^\lambda)^*(\Sigma V^* - YD_\alpha V_{and}^\lambda)) \\
=& trace((V\Sigma^* - V_{and}^{\lambda *} D_\alpha^* Y^*)(\Sigma V^* - YD_\alpha V_{and}^\lambda)) \\
=& trace(V_{and}^{\lambda *} D_\alpha^* Y^* YD_\alpha V_{and}^\lambda - V_{and}^{\lambda *} D_\alpha^* Y^* \Sigma V^* - V\Sigma^* YD_\alpha V_{and}^\lambda + V\Sigma^* \Sigma V^*) \quad \textbf{(P1)} \\
=& trace(D_\alpha^* Y^* YD_\alpha V_{and}^\lambda V_{and}^{\lambda *} - Y^* \Sigma V^* V_{and}^{\lambda *} D_\alpha^* - V_{and}^\lambda V\Sigma^* YD_\alpha + V^* V\Sigma^* \Sigma) \\
=& trace(D_\alpha^* Y^* YD_\alpha V_{and}^\lambda V_{and}^{\lambda *}) - trace(Y^* \Sigma V^* V_{and}^{\lambda *} D_\alpha^*) - trace(V_{and}^\lambda V\Sigma^* YD_\alpha) + trace(\Sigma^* \Sigma) \quad \textbf{(P2, P3)} \\
=& \alpha^*((Y^*Y) \circ (\overline{V_{and}^\lambda V_{and}^{\lambda *}}))\alpha - (\overline{diag\{V_{and}^\lambda V\Sigma^* Y\}})^* \alpha - \alpha^* (\overline{diag\{V_{and}^\lambda V\Sigma^* Y\}}) + trace(\Sigma^* \Sigma) \\
\implies J(\alpha) =& \alpha^* P\alpha - q^*\alpha - \alpha^* q + s
\end{aligned} \tag{3.26}$$

Where $P = (Y^*Y) \circ (\overline{V_{and}^\lambda V_{and}^{\lambda *}})$, $q = \overline{diag\{V_{and}^\lambda V\Sigma^* Y\}}$ and $s = trace(\Sigma^* \Sigma)$, the function $J(\alpha)$ is quadratic and hence convex: we can then calculate the gradient of J with respect to α and set it equal to 0 in order to find the optimal vector of amplitudes $\alpha_{dmd} \in \mathbb{C}^m$:

$$\begin{aligned}
\nabla J(\alpha) &= 2P\alpha - 2q = 0 \\
&\implies P\alpha - q = 0 \\
&\implies \alpha_{dmd} = P^{-1} q
\end{aligned} \tag{3.27}$$

with this vector, we can hence reconstruct the stacked vectors using (3.23):

$$\Psi_0^{m-1} \approx \Phi D_{\alpha_{dmd}} V_{and}^\lambda$$

3.2.4 Algorithm of the SVD approach

We provide here the algorithm of the SVD approach to fully reconstruct the data matrix Ψ_0^{m-1} from a sequence of snapshots $\{x_0, x_1, ..., x_m\}$:

Step 1: Stack the snapshots in the matrices $\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}]$ and $\Psi_1^m = [x_1, x_2, ..., x_m]$

Step 2: Compute the economy-sized SVD of the matrix Ψ_0^{m-1}

$$\Psi_0^{m-1} = U\Sigma V^*$$

Step 3: Compute the companion matrix $A_{dmd} = U^*AU = U^*\Psi_1^m V\Sigma^{-1}$ and its eigendecomposition:

$$A_{dmd}Y = YD_\lambda$$
$$A_{dmd} = YD_\lambda Y^{-1}$$

then compute the matrix of DMD modes $\Psi = UY$.

Step 4: Compute the Vandermonde matrix:

$$V_{and}^\lambda = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{m-1} \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_m & \lambda_m^2 & \dots & \lambda_m^{m-1} \end{bmatrix}$$

Step 5: Determine the vector of optimal amplitudes α_{dmd}:

$$\alpha_{dmd} = ((Y^*Y) \circ (\overline{V_{and}^\lambda V_{and}^{\lambda *}}))^{-1}(\overline{diag\{V_{and}^\lambda V\Sigma^* Y\}})$$

Step 6: Reconstruct the stacked snapshots:

$$\Psi_0^{m-1} \approx \Phi D_{\alpha_{dmd}} V_{and}^\lambda$$

3.3 Exact DMD

In this section, we will display the exact DMD algorithm: a generalization of the standard DMD algorithm, which allows us to analyze non-sequential data snapshots developed by Tu et al. in [41].

3.3.1 Derivation of the algorithm

While standard DMD requires a sequential set of data vectors $\{x_0, ..., x_m\}$, we relax the restrictions on the data and consider the data as a set of pairs $\{(x_k, y_k)\}_{k=1}^m$ where $y_n = f(x_n)$, we then define the two data matrices

$$X = [x_1, ..., x_m], \quad Y = [y_1, ..., y_m]$$

Then, similarly to (3.2), we assume the following linear relation

$$y_k = \hat{A}x_k \tag{3.28}$$

for an unknown matrix A.

As we know, the matrix minimizing the term $\|Y - \hat{A}X\|_F$ is given by $A = YX^\dagger$, the eigendecomposition of A gives us thus the exact DMD modes and values.

As in the SVD method, we are going to compute a companion matrix $\tilde{A}$, which is a similarity transform of the matrix A: and using assumption (3.28), it holds

$$Y = AX \tag{3.29}$$

then, computing the SVD $X = U\Sigma V^*$

$$\begin{aligned} Y &= AU\Sigma V^* \\ \implies YV\Sigma^{-1} &= AU \\ \implies \tilde{A} &= U^*YV\Sigma^{-1} = U^*AU. \end{aligned}$$

Theorem 6 *For each eigenvalue of the companion matrix $\tilde{A}$ with the corresponding eigenvector w, the vector given by*

$$\phi = \frac{1}{\lambda} YV\Sigma^{-1}w \tag{3.30}$$

is an eigenvector of the matrix A with the corresponding eigenvalue λ. Furthermore, all nonzero eigenvalues of A are identified via the eigendecomposition of $\hat{A}$

Proof: Defining the Moore-Penrose pseudoinverse via the SVD $X^\dagger = V\Sigma^{-1}U^*$ we can rewrite

$$A = YV\Sigma^{-1}U^* = BU^*,$$

where we define $B = YV\Sigma^{-1}$, we can hence rewite the companion matrix as

$$\hat{A} = U^*Y\Sigma^{-1}V = U^*B.$$

Now let (λ, w) be an eigenpair of $\hat{A}$, such that $Aw = \lambda w$ with $\lambda \neq 0$, then for the vector $\phi = \frac{1}{\lambda}Bw$ it holds

$$A\phi = \frac{1}{\lambda}BU^*Bw = \frac{1}{\lambda}B\hat{A}w = \frac{1}{\lambda}B\lambda w = Bw = \lambda\phi.$$

(λ, ϕ) is hence an eigenpair of the matrix A with $\phi \neq 0$, cause otherwise it would hold $Bw = 0$ and hence $U * Bw = \hat{A}w = 0$, which would result in $\lambda = 0$ (a contradiction),

We now show that all nonzero eigenvectors of A are identified: let (λ, ϕ) be an eigenpair of A with $\lambda \neq 0$ and define $w = U^*\phi$, then

$$\hat{A}w = U^*BU^*\phi = U^*A\phi = U^*\lambda\phi = \lambda U^*\phi = \lambda w.$$

(λ, w) is hence an eigenpair of $w = U^*\phi \neq 0$, since elsewise $A\phi = BU^*\phi = 0$, resulting in $\lambda = 0$ (a contradiction). □

We now characterize the conditions under which the operator $A = YX^\dagger$ satisfies $Y = AX$. We emphasize that this does not require that Y is generated from X through linear dynamics defined by A; we place no restrictions on the data pairs (x_k, y_k). To this end, we introduce the following definition.

Definition 7 (Linear consistency) *Two $n \times m$ matrices X and Y are linearly consistent, if, whenever $Xc = 0$ holds, then $Yc = 0$ holds as well. In other words, the nullspace of X $\mathcal{N}(X)$ is contained in the nullspace of Y $\mathcal{N}(Y)$.*

With the next theorem, we are going to show that linear consistency is a sufficient condition for the operator A to relate the data sets Y and X through equation (3.29).

Theorem 8 *Define the operator $A = YX^\dagger$, then $Y = AX$ if and only if X and Y are linearly consistent.*

Proof: We want to prove the following equivalence

$$Y = AX \iff X \text{ and } Y \text{ are linearly consistent}$$

"⇒": Suppose that $Y = AX$ holds and that X and Y are not linearly consistent, then there exists a vector $v \in \mathbb{C}^m$, such that $Xv = 0$ and $Yv \neq 0$, but this would mean $Yv \neq 0 = AXv = Yv$, a contradiction.
"⇐": We now suppose X and Y to be linear consistent, i.e. $\mathcal{N}(X) \subseteq \mathcal{N}(Y)$, then

$$Y - AX = Y - YX^{\dagger}X = Y(I - X^{\dagger}X)$$

We now recall that $X^{\dagger}X$ is the orthogonal projection onto the range of X^* (denoted $R(X^*)$), so $I - X^{\dagger}X$ is the orthogonal projection onto $\mathcal{R}(X^*)^{\perp} = \mathcal{N}(X)$. Thus, since $\mathcal{N}(X) \subseteq \mathcal{N}(Y)$, it follows that $Y(I - X^{\dagger}X) = 0$, so $Y = AX$. □

Relation with standard DMD

As it is easy to notice, the Exact DMD algorithm is extremely similar to the SVD approach of standard DMD, the only real difference between the two is that, for an eigenpair (λ, w) of the companion matrix $\hat{A}$, the DMD mode is defined as

$$\hat{\phi} = Uw, \tag{3.31}$$

while the exact DMD mode is defined as

$$\phi = \frac{1}{\lambda} YV\Sigma^{-1}w \tag{3.32}$$

with the next theorem, we are going to establish the relation between $\hat{\phi}$ and ϕ.

Theorem 9 *Let (λ, w) be an eigenpair of $\hat{A}$ with $\lambda \neq 0$ and let $\mathbb{P}_X = XX^{\dagger}$ denote the orthogonal projection onto the image of X. Then $\hat{\phi}$ given by (3.31) is an eigenvector of $\mathbb{P}_X A$ with eigenvalue λ. Furthermore, if ϕ is given by (3.32), then $\hat{\phi} = \mathbb{P}_X \phi$.*

Proof: Using the SVD $X = U\Sigma V^*$ we can rewrite the orthogonal projection in the following manner

$$\mathbb{P}_X = XX^{\dagger} = U\Sigma V^* V\Sigma^{-1}U^* = UU^*,$$

then it holds

$$\mathbb{P}_X A\hat{\phi} = UU^*BU^*Uw = UU^*BU^*Uw = UAw = \lambda Uw,$$

hence $\hat{\phi}$ is an eigenvector of $\mathbb{P}_X A$. Moreover, for ϕ it holds

$$\mathbb{P}_X \phi = \frac{1}{\lambda} UU^*YV\Sigma^{-1}w = \frac{1}{\lambda} UU^*Bw = \frac{1}{\lambda} U\hat{A}w = Uw = \hat{\phi},$$

□

We could hence establish the relation between $\hat{\phi}$ and ϕ: $\hat{\phi}$ is simply the projection of the exact DMD mode determined onto the range of X.

3.3.2 Exact DMD algorithm

We provide here the Exact DMD algorithm to obtain the Exact DMD modes and eigenvalues

Step 1: Arrange the data pairs in the two matrices

$$X = [x_1, ..., x_m], \quad Y = [y_1, ..., y_m]$$

Step 2: Compute the SVD of the matrix X

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

Step 3: Compute the eigendecomposition of the companion matrix $\hat{A} = U^* Y V \Sigma^{-1}$

$$\hat{A}W = YD_\lambda$$

where the columns of W are the eigenvectors of $\hat{A}$ and $D_\lambda = diag(\{\lambda_k\}_{k=1}^m)$ is the diagonal matrix containing the eigenvalues of $\hat{A}$

Step 4: Compute the matrix containing exact DMD modes

$$\Phi = YV\Sigma^{-1}WD_{\lambda^{-1}} \tag{3.33}$$

3.4 Examples

3.4.1 Periodic trajectories

We will now proceed in a similar manner as Tutorial 1 of the PyDMD documentation website[1] but, to avoid having to use Hankel preprocessing on the data, we will modify the f_1 and f_2 functions in the following way:

$$f_1(x,t) = \text{sech}(x+3)e^{2.3it}$$
$$f_2(x,t) = 2\text{sech}(x)\tanh(x)e^{2.8it}$$

consider, for a vector of initial states $x_0 = [x_{0_1}, ..., x_{0_n}]$ and a time step Δt, the sequence of snapshots

$$\begin{aligned}
\Psi_k &= [f_1(x_{0_1}, k\Delta t) + f_2(x_{0_1}, k\Delta t), ..., f_1(x_{0_1}, k\Delta t) + f_2(x_{0_m}, k\Delta t)] \text{ for } k \in \{0, ..., m-1\} \\
&= F_1(x_0, k\Delta t) + F_2(x_0, k\Delta t), \\
\implies \Psi_{k+1} &= e^{2.3i\Delta t}F_1(x_0, k\Delta t) + e^{2.8i\Delta t}F_2(x_0, k\Delta t) \\
&= e^{2.3(k+1)i\Delta t}F_1(x_0, 0) + e^{2.8(k+1)i\Delta t}F_2(x_0, 0)
\end{aligned} \tag{3.34}$$

where $F_1, F_2 : \mathbb{R}^n \to \mathbb{R}^n$ are vector-valued function, such that $F_i(x_0, \Delta t) = [f_i(x_{0_1}, k\Delta t), ..., f_i(x_{0_n}, k\Delta t)]$. From (3.34), it is clear that the terms $F_1(x_0, 0)$ and $F_2(x_1, 0)$ are the Ritz modes corresponding to the Ritz values $\lambda_1 = e^{2.3it}$ and $e^{2.8it}$.

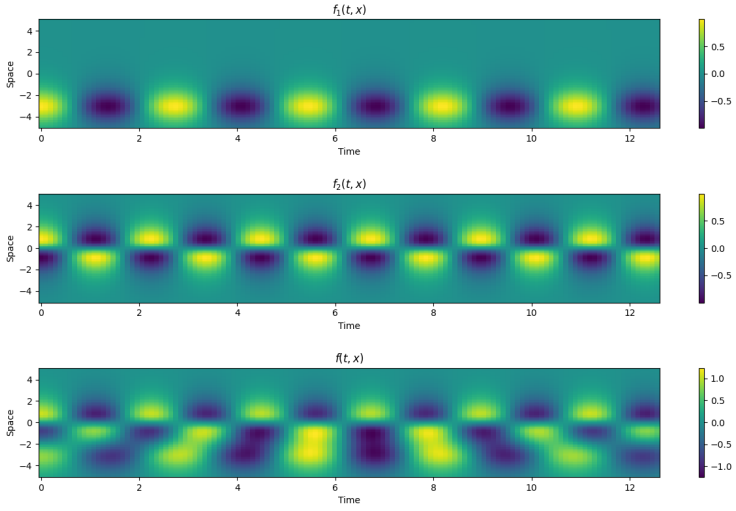

Figure 3.1 The real part of the functions f_1, f_2 and f plotted in the temporal interval $[0, 4\pi]$ and spatial interval $[-5, 5]$

[1] url: https://pydmd.github.io/PyDMD/tutorial1dmd.html

Figure 3.1 represents nothing else than a sequence of such snapshots and highlights how the temporal periodicity of the terms $e^{2.3i\Delta t}$ and $e^{2.8i\Delta t}$ is still fully recognizable in the function $f(x,t) = f_1(x,t)+f_2(x,t)$. The matrix Ψ_0^m used to generate this figure has $n = 65$ equidistant spatial points in the interval $[-5,5]$ and $m+1 = 129$ snapshots in the temporal interval $[0, 4\pi]$.

We now apply the DMD algorithm using the DMD class of the PyDMD package[2] on this data matrix $\Psi_0^m \in \mathbb{C}$, selecting the SVD truncation rank to be 2 (since we know that there are only two frequencies acting on the temporal behaviour of the snapshots), we see in Figure 3.2 that the discrete eigenvalues are exactly $\lambda_1 = e^{i2.8\Delta t}$ and $\lambda_2 = e^{i2.3\Delta t}$, resulting in the continuous eigenvalues $\mu_1 = 2.8i$ and $\mu_2 = 2.3i$ matching exactly the frequencies of the data sequence.

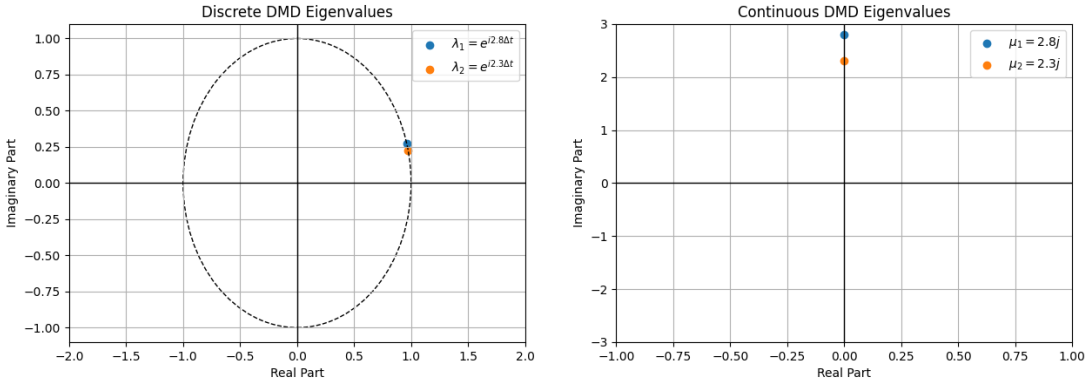

Figure 3.2 Eigenvalues extracted by the Standard DMD algorithm from the data sequence Ψ_0^{m-1}

Visualizing the modes in figure 3.3, we see that they are nothing but a scaled version of the Ritz modes $F_1(x_0, 0)$ and $F_2(x_0, 0)$, which are the terms defining the spatial ground truth of the data matrix Ψ_0^{m-1}.

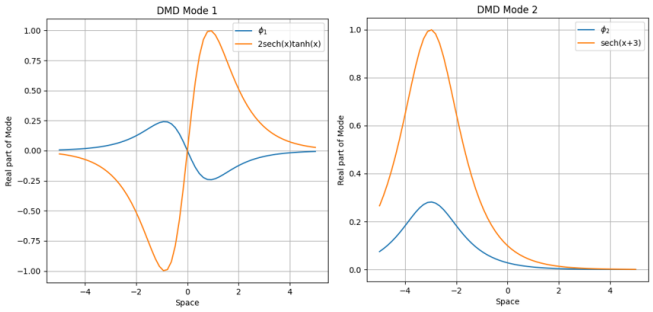

Figure 3.3 The 2 modes extracted by the DMD algorithm compared with the Ritz modes

It is hence clear that, with the 2 discrete eigenvalues λ_1 and λ_2, the 2 DMD modes ϕ_1 and ϕ_2 and the right amplitudes, we are able to fully reconstruct the data matrix Ψ_0^{m-1}, in Figure 3.4, we visualize the dynamics of those modes and the reconstructed data and underline the similarity with Figure 3.1.

While this result reassures us that the algorithm correctly recognizes the correct eigenvalues and modes needed to reconstruct the data, this method presents a very big flaw: as soon as the data becomes noisy (as all experimental measurements inevitably are) the method is not even capable of reconstructing the training data. Adding a randomized matrix $A_{rand} \in \mathbb{R}^{n\times m+1}$ generated by a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 0.2$, to the data matrix to obtain the noised measurements $\Psi_{0,noisy}^m = \Psi_0^m + A_{rand}$, in Figure 3.5 we visualize the noisy data.

Analyzing this data with the DMD algorithm, we notice in figure 3.6 that, while the modes are still identified quite well, the continuous eigenvalues are now on the left half of the real-imaginary-plane, as we can see in Figure 3.7. This results in the dynamic behaviour of the modes (and hence of the reconstructed data) to converge to zero as time goes on, making the predictions of DMD completely unreliable.

[2] url: https://pydmd.github.io/PyDMD/dmdbase.html

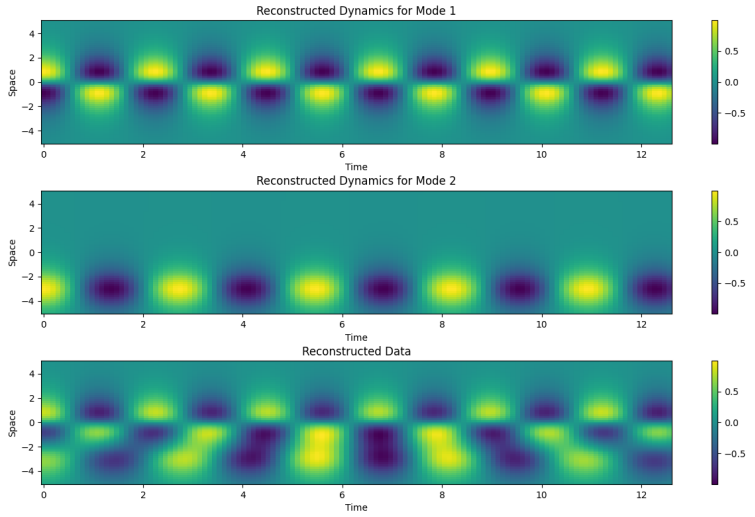

Figure 3.4 The real part of the dynamic behaviour of the 2 modes and the real part of the data matrix reconstructed by the DMD algorithm

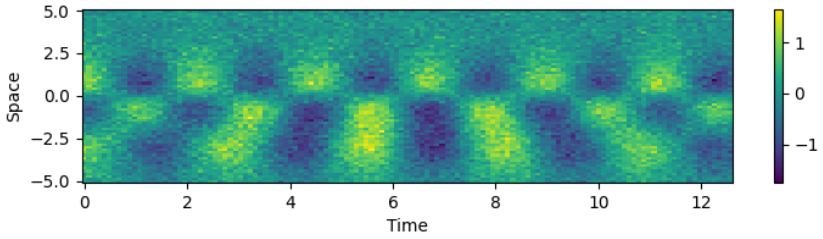

Figure 3.5 The real part of the noisy data

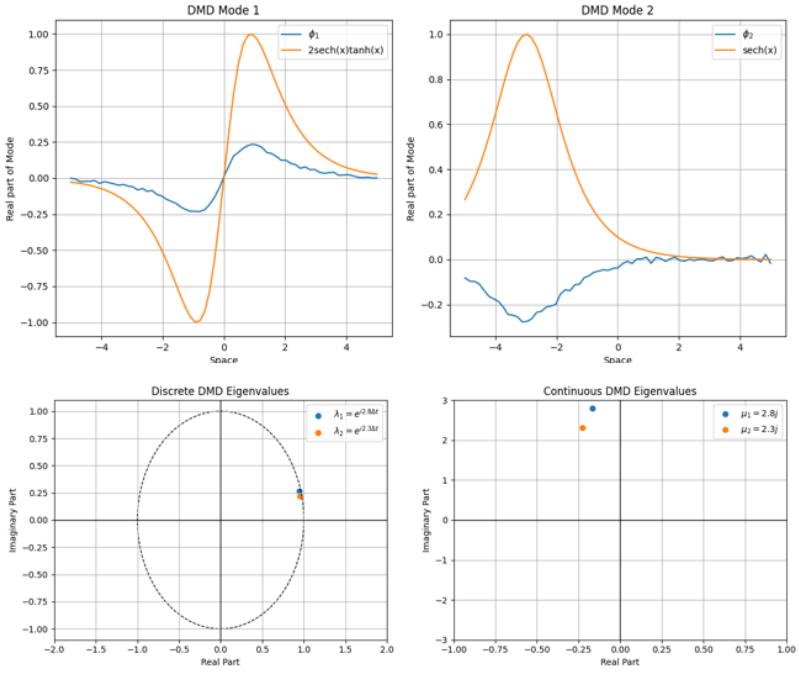

Figure 3.6 The modes and eigenvalues provided by the DMD algorithm fed with noisy data

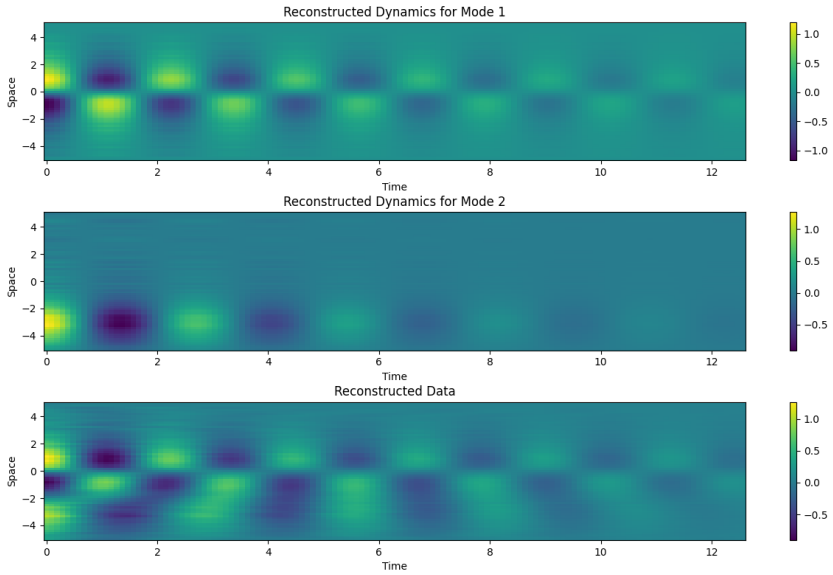

Figure 3.7 The data reconstructed by the DMD algorithm fed with noisy data, as we can see the negative real part of the continuous eigenvalues leads to the disappearing of the influence of the modes as time goes on

3.4.2 The Lorenz system

In this example, we will look at some data generated by the Lorenz system, the chaotic model discovered by E. N. Lorenz in [27], governed by the following ODE:

$$\begin{aligned} \frac{dx_1}{dt} &= \sigma(x_2 - x_1) \\ \frac{dx_2}{dt} &= x_1(\rho - x_3) - x_2 \\ \frac{dx_3}{dt} &= x_1 x_2 - \beta x_3 \end{aligned} \tag{3.35}$$

where $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$. 3.8.

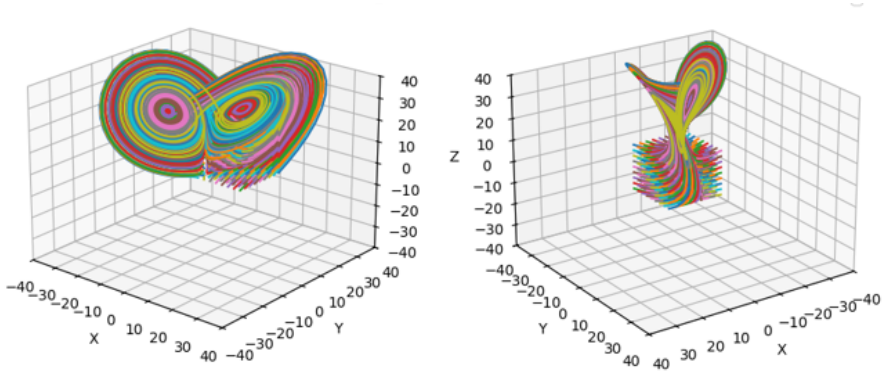

Figure 3.8 Two perspectives of the 729 trajectories generated by the fourth order Runge-Kutta integrator, the structure of the strange attractor is very easy to recognize, in the right plot, we can see the cube of initial conditions

This is an interesting model to analyze since, due to its chaoticity, the trajectories generated by the Lorenz system with two initial conditions being close to each other, will diverge as time progresses, however, it is shown that all trajectories converge to a 2D manifold, called the Lorenz attractor, because of this the trajectories will behave as almost periodic for longer periods of times and the eigenvalues will tend to the unit circle. Using a fourth order Runge-Kutta integrator, we generate $9^3 = 729$ 3-dimensional trajectories in the temporal interval $[0, 10]$ with the time step $\Delta t = 0.01$, the initial states of these trajectories

are uniformly distributed in the cube with its center located at the coordinate $[7, 7, 7]$ and its side being 20 units long, the 729 trajectories are represented in Figure 3.8.

Every kth time step of these 729 trajectories is stored in $9^3 3 = 2187$-dimensional snapshot vector $\Psi_k = [x_1(k\Delta t), y_1(k\Delta t), z_1(k\Delta t), x_2(k\Delta t), y_2(k\Delta t), z_2(k\Delta t), ..., x_{729}(k\Delta t), y_{729}(k\Delta t), z_{729}(k\Delta t)] \in \mathbb{R}^{2187}$, which is then stacked in the data matrix $\Psi_0^{1000} \in \mathbb{R}^{2187\times 1000}$. We feed this matrix Ψ_0^{1000} to the DMD class of the py-DMD python package and set the SVD rank to be optimal in the sense of the energy-thresholding method: for the economy-sized SVD of $\Psi_0^{1000} = U\Sigma V^*$ and $\Sigma = diag(\sigma_1, ..., \sigma_{1000})$, the method truncates the SVD at the lowest rank r such that $\frac{\sum_{i=1}^{r} |\sigma_i|^2}{\sum_{i=1}^{1000} |\sigma_i|^2} \geq 99.9\%$ holds.

The Standard DMD class recognizes this optimal SVD truncation rank to be $r = 483$, plotting the eigenvalues in Figure 3.10, we see that most of the eigenvalues lie very next to the unit circle, matching the stable property of Lorenz system's strange attractor.

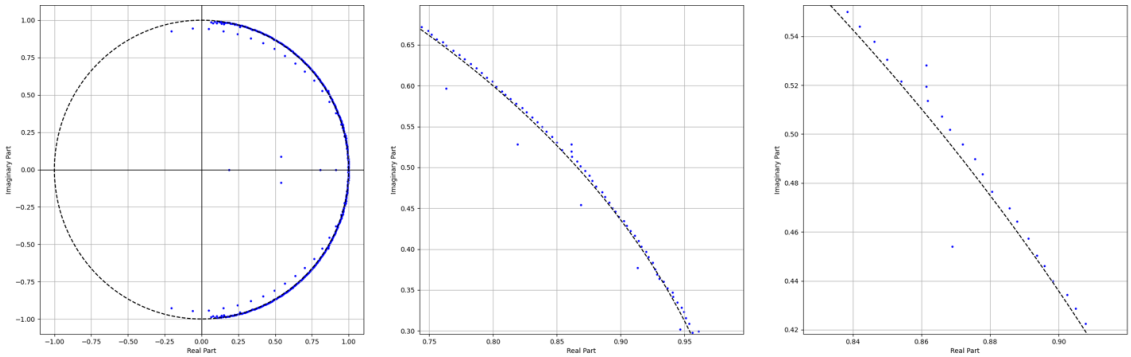

Figure 3.9 The 483 eigenvalues provided by the Standard DMD algorithm, in the second and third plot we see a zoom on the unit circle

We also visualize the first 10 dynamic modes, these also match the oscillatory behaviour of the 3 components of the trajectories in the Lorenz attractor, the modes are pairwise identical since they correspond to the couples of eigenvalues being complex conjugate to each other, having hence the same magnitude and opposite frequency. Even between those couples some similarities can be observed (see Mode 3 and Mode 5), this is due to many trajectories being very close to each other at the beginning, before the chaotic nature of the Lorenz system separates them.

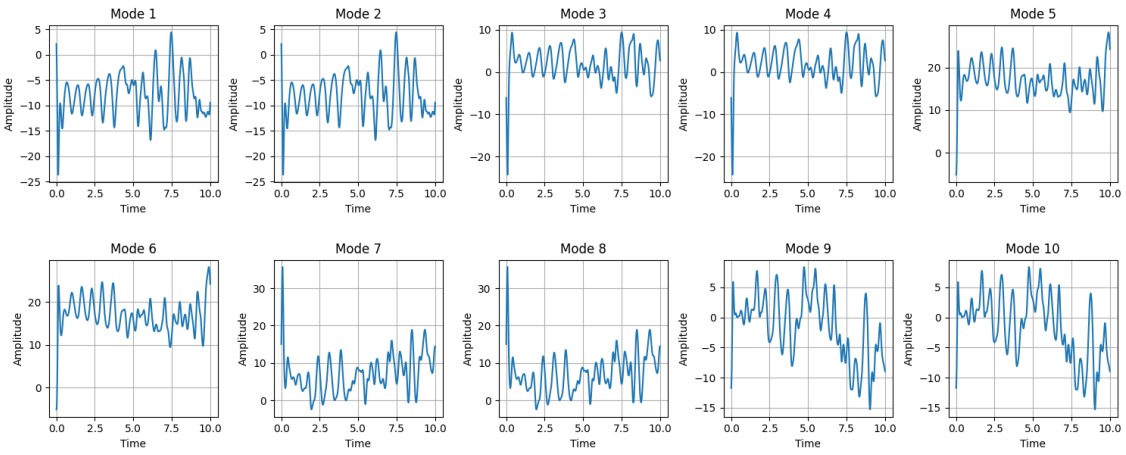

Figure 3.10 First 10 dynamic modes

Lastly, we plot the 3-dimensional reconstructed trajectories in Figure 3.11, they also match the behaviour of the trajectories plotted in Figure 3.8, such that the form of the Lorenz attractor is clearly recognizable.

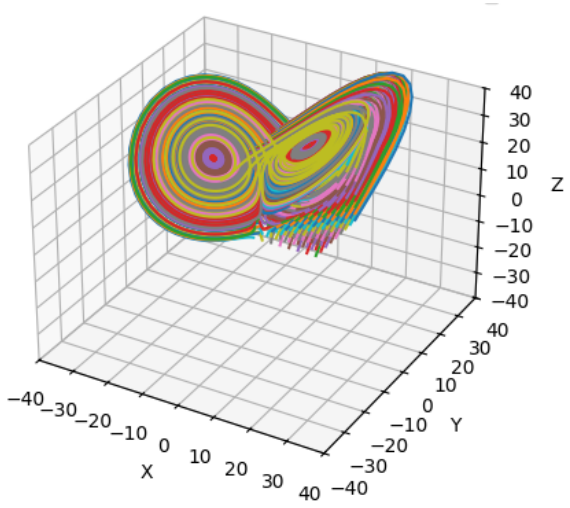

Figure 3.11 Trajectories reconstructed by the DMD algorithm

4 Low-rank and Sparse DMD

The standard DMD algorithm provides the best PyDMDm approximation A_{dmd} from equation (3.15) to the intersnapshot mapping A from (3.2), however it is very often the case that the number of significant dynamic modes r (which has to be known a-priori with insight knowledge) is lower than the number of snapshots m, hence $r << m$. While the SVD approach allows a truncation of the SVD of the matrix $\Psi_0^{m-1} = U_r \Sigma_r V_r^*$ to lower the number of modes, this method doesn't provide us the optimal PyDMDr approximation, nor does it always provide the subset of the most impactful dynamical modes. In this section we will follow the work of M. R. Jovanović et al. in [20] to derive the low-rank DMD variant, which provides the optimal rank-r approximation of A and sparse DMD, which provides the subset of modes of A having the most profound influence in the reconstruction of the snapshot sequence Ψ_0^{m-1}.

4.1 Low-rank DMD

4.1.1 Derivation of the algorithm

As we have seen in the previous chapter, the SVD approach of standard DMD provides the optimal rank-m solution to optimization problem (3.19), the low-rank DMD algorithm searches for the optimal rank-r solution, we hence obtain the minimization problem

$$\min_{\substack{F \in \mathbb{C}^{m \times m}, \\ rank(F)=r}} \|(U^* \Psi_1^m V \Sigma^{-1} - F)\Sigma\|_F^2, \tag{4.1}$$

a simple change of variables $H = U^* \Psi_1^m V$ and $F_\Sigma = F\Sigma$ transforms the optimization problem into

$$\min_{\substack{F \in \mathbb{C}^{m \times m}, \\ rank(F)=r}} \|H - F_\Sigma\|_F^2. \tag{4.2}$$

by the Eckart-Young-Mirsky theorem, we know that the rank-r truncation of the SVD of the matrix H

$$H = U_h \Sigma_h V_h^* = \sum_{i=1}^{m} u_{h_i} \sigma_{h_i} v_{h_i}^*,$$

obtained by setting the last singular values $\{\sigma_{h_{r+1}}, \sigma_{h_{r+2}}, ..., \sigma_{h_m}\}$ equal to zero, provides the rank-r solution F_Σ^r to (4.2)

$$F_\Sigma^r = \sum_{i=1}^{r} u_{h_i} \sigma_{h_i} v_{h_i}^*,$$

recalling the change of variables $F_\Sigma = F\Sigma$, the rank-r solution A_{dmd}^r to the optimization problem (4.1) is then given by

$$A_{dmd}^r = F_\Sigma^r \Sigma^{-1} = \left(\sum_{i=1}^{r} u_{h_i} \sigma_{h_i} v_{h_i}^* \right) \Sigma^{-1}.$$

This matrix A_{dmd}^r is the companion matrix provided by the low-rank DMD algorithm, we can hence proceed in the same manner as in the Standard DMD algorithm, so the next step is to compute its eigen-decomposition

$$A_{dmd}^r Y_r = Y_r \Lambda_r$$

and, analogously to Standard DMD, we compute the matrix Θ of *low-rank DMD modes*

$$\Theta = [\theta_1, \theta_2, ..., \theta_m] = U Y_r \in \mathbb{C}^{n \times m}$$

Remark: The r non-zero eigenvalues $\{\lambda_1, \lambda_2, ..., \lambda_r\}$ are not a subset of the DMD eigenvalues $\{\lambda_1, ..., \lambda_m\}$ provided by the Standard DMD algorithm: they have indeed a weaker connection to Koopman analysis, however they are chosen to to reconstruct the snapshot data Ψ_0^{m-1} with exactly r modes at best.

4.1.2 Modal reconstruction and optimal amplitudes

We follow the procedure of Section 3.2.3 and approximate the snapshots

$$x_0 = \sum_{i=1}^{m} \beta_i \theta_i,$$
$$x_k = \sum_{i=1}^{m} \beta_i \tilde{\lambda}_i^k \theta_i = \sum_{i=1}^{r} \beta_i \tilde{\lambda}_i^k \tilde{\theta}_i \ \forall k \in \{1, ..., m-1\}$$

Since only the first r eigenvalues $\{\tilde{\lambda}_1, ..., \tilde{\lambda}_r\}$ are non-zero, the set of amplitudes $\{\beta_{r+1}, ..., \beta_m\}$ are only used to approximate the initial condition x_0, but do not influence the reconstruction of the following iterations. Still following section 3.2.3, we reconstruct the matrix of stacked snapshots:

$$\Psi_0^{m-1} \approx \Theta D_\beta V_{and}^{\tilde{\lambda}} = [\theta_0, \theta_1, ..., \theta_{m-1}] \begin{bmatrix} \beta_1 & 0 & 0 & \dots & 0 \\ 0 & \beta_2 & 0 & \dots & 0 \\ 0 & 0 & \beta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \beta_m \end{bmatrix} \begin{bmatrix} 1 & \tilde{\lambda}_1 & \tilde{\lambda}_1^2 & \dots & \tilde{\lambda}_1^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \tilde{\lambda}_r & \tilde{\lambda}_r^2 & \dots & \tilde{\lambda}_r^{m-1} \\ 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix},$$

to find out the vector of optimal low-rank amplitudes $\tilde{\beta} = [\tilde{\beta}_1, ..., \tilde{\beta}_m]^\intercal \in \mathbb{C}^m$, we have hence to solve the optimization problem

$$\min_{\beta \in \mathbb{C}^m} \|\Psi_0^{m-1} - \Theta D_\beta V_{and}^{\tilde{\lambda}}\|_F^2 = \min_{\beta \in \mathbb{C}^m} \|U\Sigma V^* - U Y_r D_\beta V_{and}^{\tilde{\lambda}}\|_F^2$$
$$= \min_{\beta \in \mathbb{C}^m} \|\Sigma V^* - Y_r D_\beta V_{and}^{\tilde{\lambda}}\|_F^2$$

so, analogously to (3.27), the vector $\tilde{\beta}$ is given by

$$\tilde{\beta} = ((Y_r^* Y_r) \circ \overline{(V_{and}^{\tilde{\lambda}} V_{and}^{\tilde{\lambda}*})})^{-1} \overline{(diag\{V_{and}^{\tilde{\lambda}} V \Sigma^* Y_r\})}$$

4.1.3 Low-Rank DMD algorithm

We provide here the algorithm of the low-rank SVD approach to reconstruct the data matrix Ψ_0^{m-1} from a sequence of snapshots $\{x_0, x_1, ..., x_m\}$, for a previously chosen number of dynamic modes r:

Step 1: Stack the snapshots in the matrices $\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}]$ and $\Psi_1^m = [x_1, x_2, ..., x_m]$

Step 2: Compute the economy-sized SVD of the matrix Ψ_0^{m-1}

$$\Psi_0^{m-1} = U\Sigma V^*$$

Step 3: Compute the truncated SVD of the matrix $H = U^* \Psi_1^m V = \sum_{i=1}$:

$$H = U_h \Sigma_h^r V_h^*$$

where $\Sigma_h^r = diag(\{\sigma_{h_1}, \sigma_{h_2}, ..., \sigma_{h_r}, 0, ..., 0\}$

Step 4: Compute the eigendecomposition of the rank-r companion matrix $A^r_{dmd} = U_h \Sigma^r_h V^*_h \Sigma^{-1}$:

$$A^r_{dmd} Y_r = Y_r \Lambda,$$

then compute the matrix of low-rank DMD modes $\Theta = UY_r$.

Step 5: Compute the Vandermonde matrix $V^{\tilde{\lambda}}_{and}$:

$$V^{\tilde{\lambda}}_{and} = \begin{bmatrix} 1 & \tilde{\lambda}_1 & \tilde{\lambda}_1^2 & \dots & \tilde{\lambda}_1^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \tilde{\lambda}_r & \tilde{\lambda}_r^2 & \dots & \tilde{\lambda}_r^{m-1} \\ 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}.$$

Step 6: Compute the optimal amplitudes:

$$\tilde{\beta} = ((Y_r^* Y_r) \circ \overline{(V^{\tilde{\lambda}}_{and} V^{\tilde{\lambda}*}_{and})})^{-1} \overline{(diag\{V^{\tilde{\lambda}}_{and} V \Sigma^* Y_r\})}$$

.

Step 7: Reconstruct the stacked snapshots:

$$\Psi_0^{m-1} \approx \Theta D_{\tilde{\beta}} V^{\tilde{\lambda}}_{and}$$

4.2 Sparse DMD

We now follow the work of M. R. Jovanović in [21] to derive the Sparsity-promoting DMD algorithm, which is able the subset of dynamic modes having the most profound influence on the quality of approximation. This approach is carried out in 2 steps: in the first one, the sparsity structure of the dmd-amplitudes $\{\alpha_1, \dots \alpha_m\}$ is sought; in the second one the sparsity structure is fixed and the optimal amplitudes are determined.

4.2.1 First Step: Determination of the sparsity structure

In order to induce sparsity of the vector of amplitudes $\alpha = [\alpha_1, \dots \alpha_m]$, we could augment the objective function $J(\alpha)$ in (3.24) with a cardinality term $card(\alpha)$, which penalizes the number of non-zero components of α:

$$\min_{\alpha} \|\Sigma V^* - Y D_\alpha V_{and}\|_F^2 + \gamma card(\alpha), \tag{4.3}$$

where the scalar $\gamma > 0$ emphasizes the number of zero components of α: the higher the parameter γ is set, the sparser the resulting optimal vector will be. While this penalty term seems a plausible choice in the theory, problem (4.3) is only solvable with a combinatorial search, which becomes quickly intractable for all real-world problems.

Instead of taking the penalty term $card(\alpha)$, we relax problem (4.3) with an ℓ_1-term of α

$$\min_{\alpha} \|\Sigma V^* - Y D_\alpha V_{and}\|_F^2 + \gamma \|\alpha\|_1 = \min_{\alpha} \|\Sigma V^* - Y D_\alpha V_{and}\|_F^2 + \gamma \sum_{i=1}^{m} |\alpha_i|, \tag{4.4}$$

We will now show that this optimization is convex and illustrate an algorithm capable to solve the problem.

Alternating direction method of multipliers (ADMM)

We transform the optimization problem (4.4), replacing the vector α with the vector β in the ℓ_1-term

$$\min_{\alpha,\beta} J(\alpha) + \gamma\|\beta\|_1,$$
$$\text{subject to } \; c(\alpha,\beta) = \alpha - \beta = 0,$$

We then compute the Lagrangian function $\mathcal{L}(\alpha,\beta)$ to get rid of the equality constraint

$$\begin{aligned}\mathcal{L}(\alpha,\beta,\lambda) &= J(\alpha) + \gamma\|\beta\|_1 + \langle\lambda, \alpha-\beta\rangle \\ &= J(\alpha) + \gamma\|\beta\|_1 + \frac{1}{2}(\langle\lambda,\alpha-\beta\rangle + \langle(\alpha-\beta)^*,\lambda^*\rangle) \\ &= J(\alpha) + \gamma\|\beta\|_1 + \frac{1}{2}(\lambda^*(\alpha-\beta) + (\alpha-\beta)^*\lambda)\end{aligned}$$

where λ is the vector of Lagrange multipliers. We add the ℓ_2-squared term of the constraint to obtain the augmented Lagrangian function $\mathcal{L}_a(\alpha,\beta,\lambda)$

$$\mathcal{L}_a(\alpha,\beta,\lambda) = J(\alpha) + \gamma\|\beta\|_1 + \frac{1}{2}(\lambda^*(\alpha-\beta) + (\alpha-\beta)^*\lambda + \rho\|\alpha-\beta\|_2^2).$$

where the scalar $\rho > 0$ introduces the augmented penalty term (In case $\rho = 0$, the augmented Lagrangian is equal to the Lagrangian function)

The alternating direction method of multipliers (ADMM) consists in finding the optimized parameters $(\alpha_{opt}, \beta_{opt}, \lambda_{opt})$ through several iterative steps of the parameters, starting with some user-defined parameters (β_0, λ_0), we first compute the α-minimization step, then the β-minimization step and lastly the Lagrange multiplier update step:

$$\begin{aligned}\alpha_{k+1} &= \arg\min_{\alpha} \mathcal{L}_a(\alpha, \beta_k, \lambda_k) \\ \beta_{k+1} &= \arg\min_{\beta} \mathcal{L}_a(\alpha_{k+1}, \beta, \lambda_k) \\ \lambda_{k+1} &= \lambda_k + \rho(\alpha_{k+1} - \beta_{k+1})\end{aligned}$$

The iterations stop once the feasibility tolerances ϵ_{prim} and ϵ_{dual} are met:

$$\|\alpha_{k+1} - \beta_{k+1}\|_2 \leq \epsilon_{prim}, \quad \|\beta_{k+1} - \beta_k\|_2 \leq \epsilon_{dual}$$

α-minimization step: For given β_k, λ_k and recalling $\|p\|_2^2 = p^*p$, we see that the following quadratic equation holds:

$$\begin{aligned}\rho\left\|\alpha - \beta_k + \frac{\lambda_k}{\rho}\right\|_2^2 &= \rho\left(\left(\alpha - \beta_k - \frac{\lambda_k}{\rho}\right)^*\left(\alpha - \beta_k - \frac{\lambda_k}{\rho}\right)\right) \\ &= \rho\left(\left(\alpha^* - \beta_k^* - \frac{\lambda_k^*}{\rho}\right)\left(\alpha - \beta_k - \frac{\lambda_k}{\rho}\right)\right) \\ &= \rho\left((\alpha^* - \beta_k^*)(\alpha - \beta_k) - \frac{\lambda_k^*}{\rho}(\alpha - \beta_k) - (\alpha - \beta_k)^*\frac{\lambda_k}{\rho} + \frac{\|\lambda_k\|_2^2}{\rho^2}\right) \\ &= \rho\|\alpha - \beta_k\|_2^2 - \lambda_k^*(\alpha - \beta_k) - (\alpha - \beta_k)^*\lambda_k + \frac{\|\lambda_k\|_2^2}{\rho}\end{aligned}$$

so, with the change of variables $u_k = \beta_k - \frac{\lambda_k}{\rho}$, it holds

$$J(\alpha) + \gamma\|\beta_k\|_1 + \frac{\rho}{2}\|\alpha - u_k\|_2^2 = \mathcal{L}_a(\alpha,\beta_k,\lambda_k) + \frac{\|\lambda_k\|_2^2}{\rho}$$

and, because the terms $\frac{\|\lambda_k\|_2^2}{\rho}$ and $\gamma\|\beta_k\|_1$ are independent of α,

$$\arg\min_{\alpha} \mathcal{L}_a(\alpha, \beta_k, \lambda_k) = \arg\min_{\alpha} \left(\mathcal{L}_a(\alpha, \beta_k, \lambda_k) + \frac{\|\lambda_k\|_2^2}{\rho} \right) = \arg\min_{\alpha} \left(J(\alpha) + \frac{\rho}{2} \|\alpha - u_k\|_2^2 \right).$$

So, instead of minimizing the augmented Lagrangian function $\mathcal{L}_a(\alpha, \beta_k, \lambda_k)$, we can minimize the function $J(\alpha) + \frac{\rho}{2} \|\alpha - u_k\|_2^2$, recalling the quadratic form of $J(\alpha)$ from equation (3.26),

$$\begin{aligned}
J(\alpha) + \frac{\rho}{2} \|\alpha - u_k\|_2^2 =& J(\alpha) + \frac{\rho}{2} (\alpha - u_k)^* (\alpha - u_k) \\
=& J(\alpha) + \frac{\rho}{2} \left(\alpha^* I \alpha^* - u_k^* \alpha + \alpha * u_k + 2 \|u_k\|_2^2 \right) \\
=& \alpha^* P \alpha - q^* \alpha - \alpha^* q + s \\
&+ \frac{\rho}{2} \left(\alpha^* I \alpha^* - u_k^* \alpha + \alpha * u_k + 2 \|u_k\|_2^2 \right) \\
=& \alpha^* \left(P + \frac{\rho}{2} I \right) \alpha - \left(q + \frac{\rho}{2} u_k \right)^* \alpha - \alpha^* \left(q + \frac{\rho}{2} u_k \right) + s + \rho \|u_k\|_2^2
\end{aligned}$$

The function is quadratic and hence convex, taking the gradient and setting it to zero provides us the next step α_{k+1}:

$$\begin{aligned}
\nabla J(\alpha) = 2 \left(P + \frac{\rho}{2} I \right) \alpha_{k+1} - 2 \left(q + \frac{\rho}{2} u_k \right) &= 0 \\
\implies \left(P + \frac{\rho}{2} I \right) \alpha_{k+1} - \left(q + \frac{\rho}{2} u_k \right) &= 0 \\
\implies \alpha_{k+1} = \left(P + \frac{\rho}{2} I \right)^{-1} \left(q + \frac{\rho}{2} u_k \right)&
\end{aligned}$$

β-minimization step: With the new iteration α_{k+1} and the Lagrange multiplier λ_k, we focus on the minimization problem

$$\min_{\beta} \mathcal{L}_a(\alpha_{k+1}, \beta, \lambda_k),$$

again we underline the quadratic relation

$$\begin{aligned}
\rho \left\| \beta - \alpha_{k+1} - \frac{\lambda_k}{\rho} \right\|_2^2 &= \rho \left\| \alpha_{k+1} + \frac{\lambda_k}{\rho} - \beta \right\|_2^2 \\
&= \rho \left(\left(\alpha_{k+1} - \beta - \frac{\lambda_k}{\rho} \right)^* \left(\alpha_{k+1} - \beta_k - \frac{\lambda_k}{\rho} \right) \right) \\
&= \rho \left(\left(\alpha_{k+1}^* - \beta^* - \frac{\lambda_k^*}{\rho} \right) \left(\alpha_{k+1} - \beta - \frac{\lambda_k}{\rho} \right) \right) \\
&= \rho \left((\alpha_{k+1}^* - \beta^*) (\alpha_{k+1} - \beta) - \frac{\lambda_k^*}{\rho} (\alpha_{k+1} - \beta) - (\alpha_{k+1} - \beta)^* \frac{\lambda_k}{\rho} + \frac{\|\lambda_k\|_2^2}{\rho^2} \right) \\
&= \rho \|\alpha_{k+1} - \beta\|_2^2 - \lambda_k^* (\alpha_{k+1} - \beta) - (\alpha_{k+1} - \beta)^* \lambda_k + \frac{\|\lambda_k\|_2^2}{\rho}
\end{aligned}$$

so, with the change of variables $v_k = \alpha_{k+1} + \frac{\lambda_k}{\rho}$, it holds

$$J(\alpha_{k+1}) + \gamma\|\beta\|_1 + \frac{\rho}{2} \|\beta - v_k\|_2^2 = \mathcal{L}_a(\alpha_{k+1}, \beta, \lambda_k) + \frac{\|\lambda_k\|_2^2}{\rho}.$$

and, because the terms $\frac{\|\lambda_k\|_2^2}{\rho}$ and $J(\alpha_{k+1})$ are independent of β,

$$\arg\min_{\beta} \mathcal{L}_a(\alpha_{k+1}, \beta, \lambda_k) = \arg\min_{\beta} \left(\mathcal{L}_a(\alpha_{k+1}, \beta_k, \lambda_k) + \frac{\|\lambda_k\|_2^2}{\rho} \right) = \arg\min_{\beta} \left(\gamma\|\beta\|_1 + \frac{\rho}{2} \|\beta - v_k\|_2^2 \right).$$

So, instead of minimizing the augmented Lagrangian function $\mathcal{L}_a(\alpha_{k+1}, \beta, \lambda_k)$, we can minimize the function $c(\beta) = \gamma\|\beta\|_1 + \frac{\rho}{2}\|\beta - v_k\|_2^2$, we hence focus on the optimization problem

$$\min_{\beta} c(\beta) = \min_{\beta} \gamma\|\beta\|_1 + \frac{\rho}{2}\|\beta - v_k\|_2^2,$$

which has the optimality condition

$$0 \in \nabla c(\beta) \implies 0 \in \rho(\beta - v_k) + \gamma\partial\|\beta\|_1 \tag{4.5}$$

to do this we write out the sum forms of the ℓ_1-term and of the ℓ_2-squared term, such that we can then calculate the partial differentials:

$$\begin{aligned} c(\beta) = \gamma\|\beta\|_1 + \frac{\rho}{2}\|\beta - v_k\|_2^2 &= \gamma\sum_{i=1}^{m}|\beta_i| + \frac{\rho}{2}\sum_{i=1}^{m}(\beta_i - v_{k_i})^2 \\ \implies \frac{\partial}{\partial\beta_i}c(\beta) = \rho(\beta_i - v_{k_i}) + \gamma\frac{\partial}{\partial\beta_i}|\beta_i| &= 0 \end{aligned} \tag{4.6}$$

Now, because the function $h(x) = |x|$ is separable, we consider the two following cases:

$\beta_i \neq 0$: In this case, it holds $\frac{\partial}{\partial\beta_i}|\beta_i| = \text{sign}(\beta_i)$, it follows from optimality condition (4.6)

$$\beta_i = v_{k_i} - \frac{\gamma}{\rho}\text{sign}(\beta_i) \tag{4.7}$$

Note that if $\beta_i < 0$, then $v_{k_i} \leq -\frac{\gamma}{\rho}$ and equivalently if $\beta_i > 0$ then $v_{k_i} \geq \frac{\gamma}{\rho}$. Thus, $|v_{k_i}| > \frac{\gamma}{\rho}$ and $\text{sign}(\beta_i) = \text{sign}(v_{k_i})$

$\beta_i = 0$: The subdifferential of the ℓ_1-norm is the interval $[-1, 1]$, with optimality condition (4.5), it holds

$$\begin{aligned} 0 &\in -\rho v_{k_i} + \gamma[-1, 1] \\ \implies \rho v_{k_i} &\in [-\gamma, \gamma] \\ \implies v_{k_i} &\in \left[-\frac{\gamma}{\rho}, \frac{\gamma}{\rho}\right] \end{aligned} \tag{4.8}$$

Putting (4.7) and (4.8) together, the components of the optimal vector β_{k+1} are given by

$$\beta_{k+1_i} = \begin{cases} v_{k_i} - \frac{\gamma}{\rho} & \text{for } v_{k_i} > \frac{\gamma}{\rho} \\ 0 & \text{for } v_{k_i} \in \left[-\frac{\gamma}{\rho}, \frac{\gamma}{\rho}\right] \\ v_{k_i} + \frac{\gamma}{\rho} & \text{for } v_{k_i} < \frac{\gamma}{\rho} \end{cases}$$

which, for $\kappa = \frac{\gamma}{\rho}$, is the *soft thresholding operator* $S_\kappa(v_{k_i})$ given in Appendix B of [21].

4.2.2 Second Step: Determination of the amplitudes

We have now fully derived the algorithm to find the solution $\tilde{\alpha}$ of the optimization (4.4), now it's time to find out the optimal values of the sparse amplitudes. First, we fix the sparse structure of the vector $\tilde{\alpha}$ in a matrix E, then we determine the value of its non-zero components solving the optimization problem

$$\begin{aligned} &\min_{\alpha,\beta} J(\alpha), \\ &\text{subject to } \; E^\top\alpha = 0. \end{aligned} \tag{4.9}$$

We can now derive the optimality conditions of the problem by taking the gradient of the Lagrangian function $\mathcal{L}(\alpha, \nu)$

$$\begin{aligned}\mathcal{L}(\alpha, \nu) &= J(\alpha) + \nu^* E^\top \alpha + (E^\top \alpha)^* \nu \\ &= \alpha^* P \alpha - q^* \alpha - \alpha^* q + \nu^* E^\top \alpha + (E^\top \alpha)^* \nu,\end{aligned}$$

with respect to α and the vector of the Lagrange multipliers ν and setting them equal to zero

$$\begin{aligned}\nabla_\alpha \mathcal{L}(\alpha, \nu) = 2P\alpha - 2q - 2E\nu = 0 \\ \implies P\alpha - E\nu = q, \\ \nabla_\nu \mathcal{L}(\alpha, \nu) = 2E^\top \alpha = 0 \\ \implies E^\top \alpha = 0.\end{aligned}$$

The optimality conditions of the optimality problem (4.9) are hence given by

$$\begin{bmatrix} P & E \\ E^\top & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \nu \end{bmatrix} = \begin{bmatrix} q \\ 0 \end{bmatrix} \tag{4.10}$$

and so the optimal amplitudes of the sparsity-promoting DMD algorithm are given by

$$\alpha_{sp} = [I, 0] \begin{bmatrix} P & E \\ E^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} q \\ 0 \end{bmatrix} \tag{4.11}$$

Example (Encoding sparsity structure in matrix E): Let the optimal vector $\tilde{\alpha} \in \mathbb{C}^4$ have the structure

$$\tilde{\alpha} = [0, \tilde{\alpha}_2, 0, \tilde{\alpha}_4]^\top$$

since the first and third components are zero, the matrix E encoding the sparsity structure is given by

$$E = [e_1, e_3] = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

4.2.3 Sparse DMD algorithm

In this subsection we provide the Sparse DMD algorithm, since the only difference with Standard DMD is the determination of the amplitudes, the first 4 steps are identical to the ones provided by algorithm 3.2.4

Step 1: Stack the snapshots in the matrices $\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}]$ and $\Psi_1^m = [x_1, x_2, ..., x_m]$

Step 2: Compute the economy-sized SVD of the matrix Ψ_0^{m-1}

$$\Psi_0^{m-1} = U\Sigma V^*$$

Step 3: Compute the companion matrix $A_{dmd} = U^* A U = U^* \Psi_1^m V \Sigma^{-1}$ and its eigendecomposition:

$$\begin{aligned}A_{dmd} Y &= Y D_\lambda \\ A_{dmd} &= Y D_\lambda Y^{-1}\end{aligned}$$

then compute the matrix of DMD modes $\Psi = UY$.

Step 4: Compute the Vandermonde matrix:

$$V_{and}^\lambda = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \dots & \lambda_1^{m-1} \\ 1 & \lambda_2 & \lambda_2^2 & \dots & \lambda_2^{m-1} \\ 1 & \lambda_3 & \lambda_3^2 & \dots & \lambda_3^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_m & \lambda_m^2 & \dots & \lambda_m^{m-1} \end{bmatrix}$$

Step 5: For a chosen value $\gamma > 0$, determine the sparsity structure of the optimal sparse amplitudes α_{sp} by solving the optimization problem

$$\min_{\alpha} \|\Sigma V^* - Y D_\alpha V_{and}\|_F^2 + \gamma \|\alpha\|_1 \tag{4.12}$$

with the ADMM algorithm (for further information, see subsection 4.2.1).

Step 6: Encode the sparsity structure of the vector $\tilde{\alpha} \in \mathbb{C}^m$ solving (4.12) in a matrix E: collect the indeces of the components $\tilde{\alpha}$ which are equal to zero in a sequence $\{i_1, ..., i_s\}$ $(s \leq m)$, then $E = [e_{i_1}, ..., e_{i_s}]$, where $e_{i_j} \in \mathbb{C}^m$ is the i_j-th unit vector.

Step 7: Determine the vector of optimal sparse amplitudes α_{sp} by solving

$$\min_{\alpha,\beta} J(\alpha),$$
$$\text{subject to } \; E^\intercal \alpha = 0.$$

The solution is then given by

$$\alpha_{sp} = [I, 0] \begin{bmatrix} P & E \\ E^\intercal & 0 \end{bmatrix}^{-1} \begin{bmatrix} q \\ 0 \end{bmatrix},$$

where $P = (Y^*Y) \circ (\overline{V_{and}^{\lambda} V_{and}^{\lambda *}})$.

Step 8: Reconstruct the stacked snapshots:

$$\Psi_0^{m-1} \approx \Phi D_{\alpha_{sp}} V_{and}^{\lambda}$$

4.3 Examples

4.3.1 The Lorenz system

As we have seen in Example 3.4.2, the data set of the 729 3-dimensional trajectories, generated as iterates of the Lorenz system, has a quite high amount of DMD eigenvalues, this makes it an interesting model to compare the reconstruction error of the standard, low-rank and sparse DMD algorithms.

Comparison between Sparse DMD and standard DMD

In order to highlight the choice of amplitudes of the Sparse DMD algorithm, we first illustrate in Figure 4.1 the dependence of the absolute value of the amplitudes $|\alpha_i|$ of the DMD modes obtained by solving optimization problem (3.24) on the frequency of the corresponding eigenvalue λ_i.

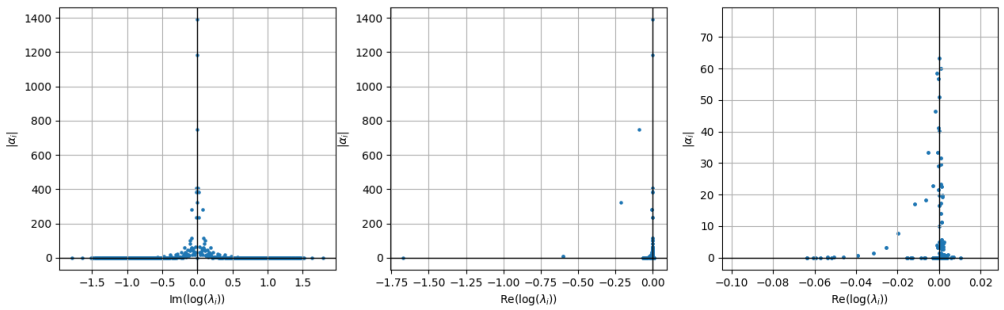

Figure 4.1 Dependence of the absolute value of the DMD amplitudes α_i on the frequency (left) and on the real part of the corresponding DMD eigenvalues λ_i (center and right). The right subplot focuses on the amplitudes that correspond to lightly damped and lightly unstable eigenvalues; it represents a zoomed version of the second subplot

As we can see in the first subplot, the most prominent amplitudes appear at the low-frequency terms; from the second and third subplot we recognize that the most prominent amplitudes are also highly concentrated at almost stable eigenvalues, such that $Re(\log(\alpha_i)) \approx 0$, the choice of important amplitudes should also point out to low-frequency, almost stable eigenvalues.

We now apply the SpDMD class from the PyDMD python package[1] on the training data on the data matrix $\Psi_0^{1000} \in \mathbb{R}^{2187\times 1000}$ from Example 3.4.2, in Figure 4.2, we show that raising the user-specified parameter γ, the number of nonzero amplitudes $card(\alpha)$ decreases, while the normalized residual $\frac{\|\Psi_0^{m-1} - \Phi D_\alpha V_{and}\|_F}{\|\Psi_0^{m-1}\|_F}$ increases.

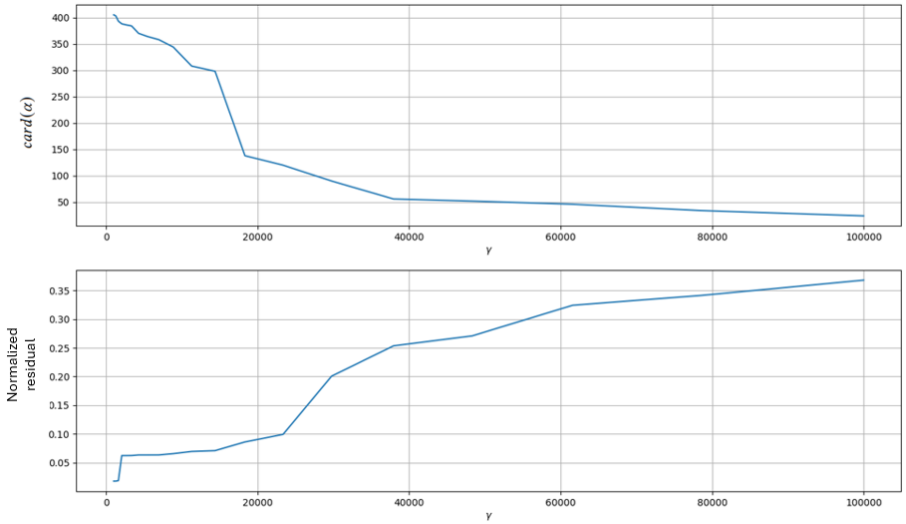

Figure 4.2 Cardinality term (upper plot) and normalized residual term (lower plot) evolving as the parameter γ grows

In Figure 4.3, we visualize the standard DMD eigenvalues from Figure 3.10 together with the selection of sparse DMD eigenvalues for several sparsity structures chosen by the user: as we can see, the eigenvalues with the most profound influence indeed tend to be the low-frequency almost-stable ones. The choice of

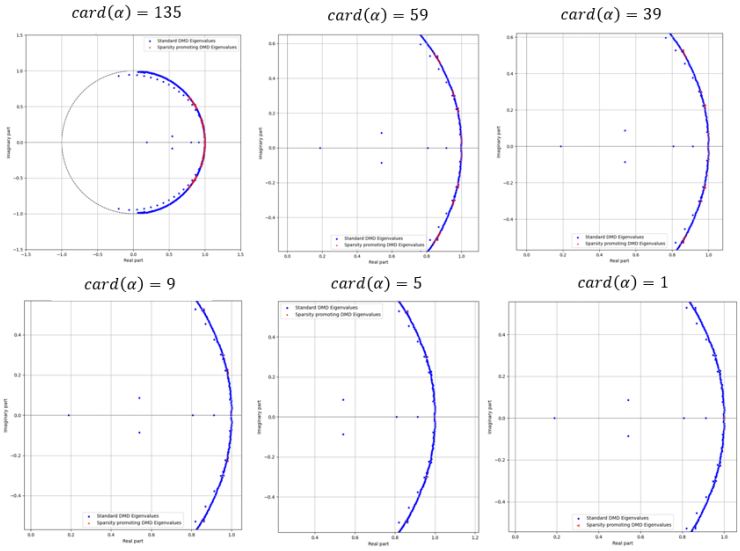

Figure 4.3 Eigenvalues selected by the Sparse DMD algorithm for different sparsity structures

amplitudes is however not trivial, in Figure 4.4 the absolute values of the sparse amplitudes are plotted:

[1] url: https://pydmd.github.io/PyDMD/spdmd.html

although the amplitudes do tend to correspond to low-frequency eigenvalues, these have very low absolute values and are more distributed on the frequencies of eigenvalues than expected.

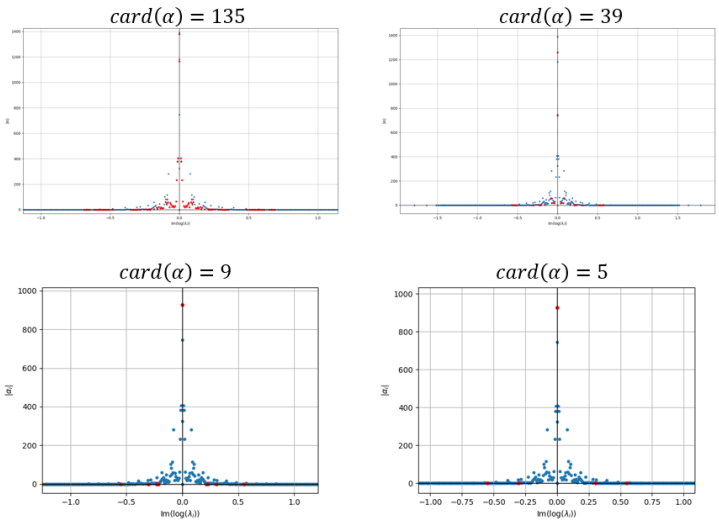

Figure 4.4 Absolute value of the sparse amplitudes (the red ones) next to the standard DMD amplitudes for several sparsity structure

Comparison between low-rank DMD and standard DMD

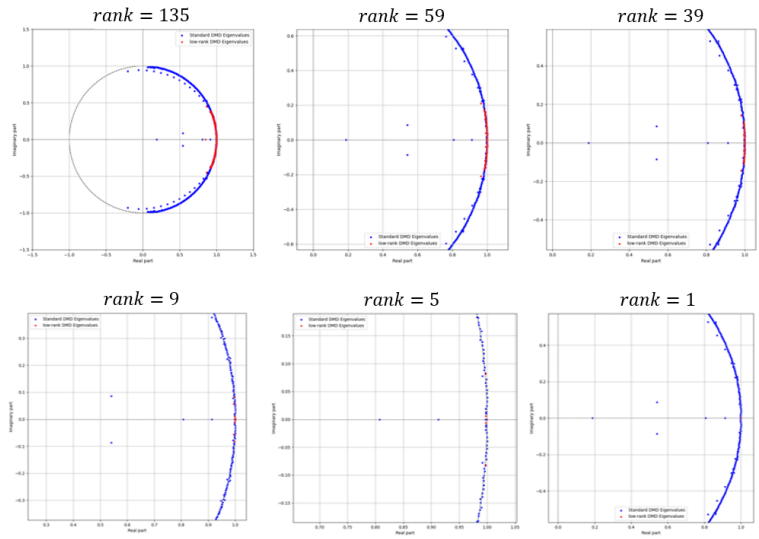

Figure 4.5 Absolute value of the low-rank eigenvalues (the red ones) next to the standard DMD amplitudes for several sparsity degrees of approximation

A whole different picture emerges when we apply a self-implemented low-rank DMD python class (for further informations see Appendix A.1.1) on the dataset and look at the visualization of the eigenvalues in Figure 4.5: as already mentioned, the eigenvalues identified by low-rank DMD differ from the ones determined by the standard DMD; in addition, the low-rank DMD eigenvalues show low-frequency, almost-stable behaviour, matching the behaviour shown in Figure 4.1 and a similar concentration of the sparse eigenvalues shown in Figure 4.3.

Comparison between Sparse DMD and low-rank DMD

Finally in Figure 4.6 we visualize the normalized reconstruction error of the sparse, low-rank and standard DMD for different ranks, as we can observe, the error of low-rank and sparse DMD is very competitive, with Sparse DMD being slightly more precise better than low-rank DMD for lower ranks and low-rank DMD being slightly more precise for higher ranks. It has to be noted that although sparse DMD has a more profound connection to the Koopman operator (since the eigenvalues identified are a subset of the eigenvalues of standard DMD), its iterative nature leads to higher computational times than low-rank DMD.

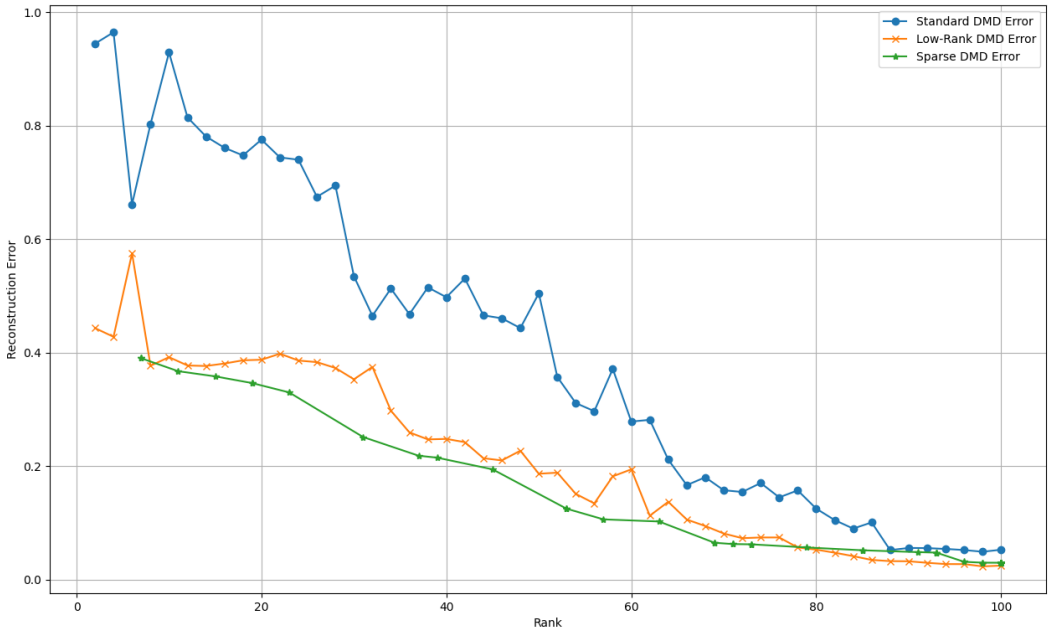

Figure 4.6 Reconstruction error of law-rank, sparse and standard DMD methods for several ranks, for the standard DMD plot the rank is given by the truncation of the SVD of Ψ_0^{m-1}, while for sparse DMD the amount of active amplitudes is taken as rank, since the inactive modes do not contribute to the reconstruction of the dynamics

5 Optimized DMD (OptDMD)

While the standard DMD algorithm treats the snapshots $x_k = x(t_k) = x(k\Delta t),\ k \in \{0, ..., m\}$ pairwise, aiming to decompose a similarity transform of the matrix A ruling the linear dynamical system (3.2); the optimized DMD algorithm (optDMD), developed by T. Askham and J. N. Kutz in [3], treats all the snapshots of the data at once by solving a (potentially) large dimensional optimization problem. The optimized DMD algorithm presents two fundamental advantages compared to the standard DMD algorithm: the first one is that the time values t_i of the snapshots do not need to be separated by a fixed time step Δt, it is uniquely expected for them to satisfy $t1 < t2 < ... < t_m$; moreover, treating all the snapshots at once, the reconstruction error is distributed in all of the snapshots, instead of only in the last one, making the algorithm more robust for predictions of *noisy* (hence experimental) data.

5.1 Variable projection method

In this section, following the pipeline of [3], we illustrate the variable projection algorithm used to solve the optimization problem obtained via the inverse differential equations (see next section for further information).

5.1.1 Vector form

Let us consider the vector nonlinear least squares problem given by

$$\min_{\alpha \in \mathbb{C}^k, \beta \in \mathbb{C}^l} \|\eta - \Phi(\alpha)\beta\|_2, \tag{5.1}$$

where $\eta \in \mathbb{C}^m$, $\Phi(\alpha) \in \mathbb{C}^{m \times l}$ and $m > l$ holds.

A typical example for problems of this form is the reconstruction of a nonlinear function $\eta(t)$ as the linear combination of l nonlinear functions $\phi_j(\alpha, t)$ with the β_is as coefficients, in such a case the components of η are given by $\eta_i = \eta(t_i)$ and the components of the matrix $\Phi(\alpha)$ are given by $\Phi(\alpha)_{ij} = \phi_j(\alpha, t_i)$, such that the dependence of the matrix $\Phi(\alpha)$ on the times t_i is implicit.

To apply the variable projection method on this problem we notice that, fixing the vector α, the vector β minimizing the function $J(\beta) = \|\eta - \Phi(\alpha)\beta\|_2$ is given by $\beta = \Phi(\alpha)^\dagger \eta$, with this equation we can rewrite optimization problem (5.1) as an α-minimization problem only

$$\min_{\alpha \in \mathbb{C}^k} \frac{1}{2}\|\eta - \Phi(\alpha)\Phi(\alpha)^\dagger \eta\|_2^2, \tag{5.2}$$

where we have squared the ℓ_2-residual and multiplied with $\frac{1}{2}$ for notational convenience.

Levenberg-Marquardt algorithm and calculation of the Jacobi matrix

To solve problem (5.2), typically (see [17, 19, 24, 31]) the Levenberg-Marquardt algorithm [25, 29] is used, this method is an iterative method which should converge to nearby a (possibly local) minimizer.

Let us define the residual function

$$\begin{aligned} \rho(\alpha) &= \eta - \Phi(\alpha)\Phi(\alpha)^\dagger \eta \\ &= (I - \Phi(\alpha)\Phi(\alpha)^\dagger)\eta \end{aligned} \tag{5.3}$$

starting with an initial guess $\alpha_0 \in \mathbb{C}^k$, the Levenberg-Marquardt algorithm gives us the update vector δ_i, such that the next iteration is given by $\alpha_{i+1} = \alpha_i - \delta_i$, as the solution of the optimization problem

$$\min_{\delta_i \in \mathbb{C}^k} \left\| \begin{bmatrix} J(\alpha_i) \\ \nu_i M(\alpha_i) \end{bmatrix} \delta_i - \begin{bmatrix} \rho(\alpha_i) \\ 0 \end{bmatrix} \right\|_2^2, \tag{5.4}$$

where $J(\alpha_i) \in \mathbb{C}^{l \times k}$ is the Jacobian matrix of the residual function $\rho(\alpha)$ evaluated at the actual iteration α_i and $M \in (\alpha_i)\mathbb{C}^{k \times k}$ is a diagonal matrix, such that $M(\alpha_i)_{jj} = \|J(\alpha_i)(:, j)\|_2$, and $J(\alpha_i)(:, j)$ is the jth column of the matrix $J(\alpha_i)$. Typically the parameter ν_i is chosen as part of a trust-region method, i.e. ν_i is increased until a step δ_i is found so that the new α_{i+1} results in a smaller residual $\|\rho(\alpha_{i+1})\|_2$.

In order to apply this method, we must have an expression for the Jacobian of $\rho(\alpha)$. The derivatives of Φ with respect to α are known analytically, e.g. they are simple to obtain, in our case we will use the l functions $\phi_j(\alpha, t) = e^{\alpha_j t}$. We therefore assume that these derivatives are available. For simplicity of notation, we will leave out the dependence of the matrices on α in order to simplify the notation. Let $\mathbb{P}_\Phi$ denote the orthogonal projection onto the columns of Φ, i.e. $\mathbb{P}_\Phi = \Phi\Phi^\dagger$, and $\mathbb{P}_\Phi^\perp$ denote the projection onto the complement of the column space of Φ, i.e. $\mathbb{P}_\Phi^\perp = I - \Phi\Phi^\dagger$, which is precisely equation (5.3), hence $\rho = \mathbb{P}_\Phi^\perp \eta$.

From Lemma 4.1 of [18], we have

$$J(:, j) = \frac{\partial \rho}{\partial \alpha_j} = -\left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger + \left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right)^* \right) \eta, \tag{5.5}$$

we now compute the two terms $\left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right) \eta$ and $\left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right)^* \eta$, computing a rank-$r$ reduced SVD $\Phi = U\Sigma V^*$ with $U \in \mathbb{C}^{m \times r}$, $V \in \mathbb{C}^{l \times r}$ and $\Sigma \in \mathbb{R}^{r \times r}$ and, recalling that $\Phi^\dagger = V\Sigma^{-1}U^*$, we notice that

$$\begin{aligned} \mathbb{P}_\Phi^\perp &= I - \Phi\Phi^\dagger \\ &= I - U\Sigma V^* V \Sigma^{-1} U^* \\ &= I - UU^* \end{aligned}$$

and hence that

$$\mathbb{P}_\Phi^\perp \eta = (I - UU^*)\eta = (I - UU^*)^* \eta = (\mathbb{P}_\Phi^\perp)^* \eta. \tag{5.6}$$

Thus, the first term is given by

$$\begin{aligned} \left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right) \eta &= \left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right) \eta \\ &= (I - UU^*) \frac{\partial \Phi}{\partial \alpha_j} V \Sigma U^* \eta \\ &= (I - UU^*) \frac{\partial \Phi}{\partial \alpha_j} \beta, \end{aligned}$$

where we have solved for β recalling $\beta = \Phi^\dagger \eta = V\Sigma U^* \eta$; the second term is given by

$$\begin{aligned} \left(\mathbb{P}_\Phi^\perp \frac{\partial \Phi}{\partial \alpha_j} \Phi^\dagger \right)^* \eta &= \Phi^{\dagger *} \frac{\partial \Phi}{\partial \alpha_j}^* (\mathbb{P}_\Phi^\perp)^* \eta \\ &= (V\Sigma^{-1}U^*)^* \frac{\partial \Phi}{\partial \alpha_j}^* \mathbb{P}_\Phi^\perp \eta \\ &= U\Sigma^{-1}V^* \frac{\partial \Phi}{\partial \alpha_j}^* \rho, \end{aligned}$$

where we used the fact that the residual is given by $\rho = \mathbb{P}_\Phi^\perp \eta$.

We have now a full expression for $J(:, j)$ for $j \in \{1, ..., k\}$ and are hence able to compute the Levenberg-Marquardt update step δ_i solving the optimization problem(5.4), to find the next iterate α_{i+i}.

5.1.2 Matrix form

One of the pivotal results introduced in [17] was the extension of the variable projection method for multiple right hand sides $H \in \mathbb{C}^{m\times n}$, i.e. to the matrix norm of problem (5.1)

$$\min_{\alpha\in\mathbb{C}^k, B\in\mathbb{C}^{l\times n}} \|H - \Phi(\alpha)B\|_F \tag{5.7}$$

where $\Phi(\alpha) \in \mathbb{C}^{m\times l}$ and $m > l$.

The example equivalent to the one from the subsection above would translate in the approximation of n functions $\eta_i(t)$ as linear approximations of l nonlinear functions $\phi_j(\alpha, t)$ with the coefficients given B_{ij}, in this case the components of H are to be interpreted as $H_{ij} = \eta_j(t_i)$.

Let $\eta \in \mathbb{C}^{mn}$ be the vector given by the stacked columns of the matrix H and $\beta \in \mathbb{C}^{ln}$, then problem (5.7) is equivalent to

$$\min_{\alpha\in\mathbb{C}^k, \beta\in\mathbb{C}^{ln}} \|\eta - I_n \otimes \Phi(\alpha)\beta\|_2, \tag{5.8}$$

where $A \otimes B$ is the Kronecker product of two matrices A and B.

Again, similarly to the subsection above, for a fixed vector α, the optimal matrix B is given by $B = \Phi^\dagger H$, such that β can be computed in blocked form. In the same way, we can compute the residual ρ in blocked form, taking the function $P = H - \Phi B = H - \Phi\Phi^\dagger H = (I - \Phi\Phi^\dagger)H$, then ρ is given as the vector of stacked columns of the matrix P. Most importantly, the Jacobian matrix can be put in blocked form too, setting

$$J_j^{mat} = -\left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger + \left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger\right)^*\right) H \tag{5.9}$$

then the jth column of the Jacobian matrix $J(:, j)$ is given by the stacked columns of the matrix J_j^{mat}.

We can then compute the two terms $\left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger\right) H$ and $\left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger\right)^* H$, taking the SVD $\Phi = U\Sigma V^*$ we can see that

$$\begin{aligned}\left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger\right) H &= (I - UU^*)\frac{\partial\Phi}{\partial\alpha_j}V\Sigma^{-1}U^*H \\ &= (I - UU^*)\frac{\partial\Phi}{\partial\alpha_j}B\end{aligned} \tag{5.10}$$

where we have solved for the matrix B using $B = \Phi^\dagger H = V\Sigma^{-1}U^*H$, and

$$\begin{aligned}\left(\mathbb{P}_\Phi^\perp \frac{\partial\Phi}{\partial\alpha_j}\Phi^\dagger\right)^* H &= (V\Sigma^{-1}U^*)^* \frac{\partial\Phi}{\partial\alpha_j}^* (\mathbb{P}_\Phi^\perp)^* H \\ &= U\Sigma^{-1}V^* \frac{\partial\Phi}{\partial\alpha_j}^* \mathbb{P}_\Phi^\perp H \\ &= U\Sigma^{-1}V^* \frac{\partial\Phi}{\partial\alpha_j}^* P\end{aligned} \tag{5.11}$$

where we have used $P = \mathbb{P}_\Phi^\perp H$.

With these terms, for some iteration α_i, we are able to compute the columns of the Jacobian matrix $J(\alpha_i)$ (and hence assemble it) to solve the optimization problem (5.4) to obtain the Marquardt-Levenberg update step δ_i, such that the next iteration is given by $\alpha_{i+1} = \alpha_i - \delta_i$. such that the computational cost of all k columns of the Jacobian matrix J becomes $O(k^2m + kmn)$ instead of $O(k^2m + kmn + k^2n + k^2mn)$.

5.1.3 Inverse differential equations

Suppose that $z(t) \in \mathbb{C}^n$ is the solution of

$$\dot{z}(t) = Az(t) \tag{5.12}$$

for a matrix $A \in \mathbb{C}^{n\times n}$.

For some initial condition $z_0 = z(t_1 = 0)$, it is well known that the analytical solution of this ODE is given by

$$z(t) = e^{At} z_0 \tag{5.13}$$

Assuming that the matrix A is diagonalizable (i.e. $A = S\Lambda S^{-1}$ for $\Lambda = diag(\alpha_i)$, with the eigenvalues $\alpha_1, ..., \alpha_n$ being distinct), we can rewrite the solution

$$\begin{aligned}
z(t) &= e^{S\Lambda S^{-1}t} z_0 \\
&= e^{S\Lambda t S^{-1}} z_0 \\
&= S e^{\Lambda t} S^{-1} z_0 \\
&= [S_1, S_2, ..., S_n] e^{\Lambda t} S^{-1} z_0 \\
&= [e^{\alpha_1 t} S_1, e^{\alpha_2 t} S_2, ..., e^{\alpha_n t} S_n] S^{-1} z_0, \\
&= [e^{\alpha_1 t} S_1, e^{\alpha_2 t} S_2, ..., e^{\alpha_n t} S_n] \begin{bmatrix} (S^{-1}z_0)_1 \\ (S^{-1}z_0)_2 \\ \vdots \\ (S^{-1}z_0)_n \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^n e^{\alpha_i t} S_{1i} (S^{-1}z_0)_i \\ \sum_{i=1}^n e^{\alpha_i t} S_{2i} (S^{-1}z_0)_i \\ \vdots \\ \sum_{i=1}^n e^{\alpha_i t} S_{ni} (S^{-1}z_0)_i \end{bmatrix} \\
\implies z(t)^\intercal &= \begin{bmatrix} \sum_{i=1}^n e^{\alpha_i t} S_{1i} (S^{-1}z_0)_i & \sum_{i=1}^n e^{\alpha_i t} S_{2i} (S^{-1}z_0)_i & \dots & \sum_{i=1}^n e^{\alpha_i t} S_{ni} (S^{-1}z_0)_i \end{bmatrix} \\
&= [e^{\alpha_1 t}, e^{\alpha_2 t}, ..., e^{\alpha_n t}] \begin{bmatrix} S_{11}(S^{-1}z_0)_1 & S_{21}(S^{-1}z_0)_1 & \dots & S_{n1}(S^{-1}z_0)_1 \\ S_{12}(S^{-1}z_0)_2 & S_{22}(S^{-1}z_0)_2 & \dots & S_{n2}(S^{-1}z_0)_2 \\ \vdots & \vdots & \ddots & \vdots \\ S_{1n}(S^{-1}z_0)_n & S_{2n}(S^{-1}z_0)_n & \dots & S_{nn}(S^{-1}z_0)_n \end{bmatrix}
\end{aligned}$$

Then, defining the n functions $\phi_j(t) = e^{\alpha_j t}$ and the matrices $\Phi(\alpha)$ and H as in the subsections above, i.e. $\Phi(\alpha)_{ij} = \phi_j(\alpha, t_i)$ and $H_{ij} = z(t_i)_j$ we can see that it holds

$$\begin{aligned}
H &= \begin{bmatrix} z(t_1)^\intercal \\ \vdots \\ z(t_m)^\intercal \end{bmatrix} \\
&= \begin{bmatrix} e^{\alpha_1 t_1} & e^{\alpha_2 t_1} & \dots & e^{\alpha_n t_1} \\ e^{\alpha_1 t_2} & e^{\alpha_2 t_2} & \dots & e^{\alpha_n t_2} \\ \vdots & \vdots & \ddots & \vdots \\ e^{\alpha_1 t_m} & e^{\alpha_2 t_m} & \dots & e^{\alpha_n t_m} \end{bmatrix} \begin{bmatrix} S_{11}(S^{-1}z_0)_1 & S_{21}(S^{-1}z_0)_1 & \dots & S_{n1}(S^{-1}z_0)_1 \\ S_{12}(S^{-1}z_0)_2 & S_{22}(S^{-1}z_0)_2 & \dots & S_{n2}(S^{-1}z_0)_2 \\ \vdots & \vdots & \ddots & \vdots \\ S_{1n}(S^{-1}z_0)_n & S_{2n}(S^{-1}z_0)_n & \dots & S_{nn}(S^{-1}z_0)_n \end{bmatrix} = \Phi(\alpha) B,
\end{aligned} \tag{5.14}$$

where the entries of the matrix $B \in \mathbb{C}^{n \times n}$ is given by $B_{ij} = S_{ji}(S^{-1}z_0)_i$.

The inverse differential equations problem can be then solved by first solving the problem

$$\min_{\alpha \in \mathbb{C}^n, B \in \in \mathbb{C}^{n \times n}} \|H - \Phi(\alpha) B\|_F$$

notice that this is the version of problem (5.7) with $n = l = k$.

We now notice that

$$B^\intercal = \begin{bmatrix} S_{11}(S^{-1}z_0)_1 & \dots & S_{1n}(S^{-1}z_0)_n \\ \vdots & \ddots & \vdots \\ S_{n1}(S^{-1}z_0)_1 & \dots & S_{nn}(S^{-1}z_0)_n \end{bmatrix} = \begin{bmatrix} S(:,1)(S^{-1}z_0)_1 & \dots & S(:,n)(S^{-1}z_0)_n \end{bmatrix}, \tag{5.15}$$

the ith column of $B^\intercal$ is hence the ith column of S scaled by the factor $(S^{-1}z_0)_i$, since by definition the columns of S are eigenvectors of A, the same holds for the columns of $B^\intercal$. We are hence able to reconstruct the matrix A from the eigenvectors given by the columns of $B^\intercal$ and the eigenvalues given by the components of the vector α.

Remark: We note that the variable projection framework also applies immediately to the case that $n > l$, i.e. to the case of fitting an l nonlinear functions $\phi_j(\alpha, t)$ to a a higher dimensional set of trajectories $\eta_i(t)$. In such a case, we can just diagonalize the matrix A just truncating at rank l

$$A = S\Lambda S^\dagger$$

where

$$S \in \mathbb{C}^{n\times l}$$
$$\Lambda \in \mathbb{C}^{l\times l}$$

Then the derivation of equation (5.14) follows in the same manner as for the full rank case, but with $\Phi(\alpha) \in \mathbb{C}^{m\times l}$. The optimized DMD algorithm provided in the next section is the direct result of this observation.

5.2 Optimized DMD

5.2.1 Derivation of the algorithm

Let $\Psi_0^m = [x_0, x_1, ..., x_m] \in \mathbb{C}^{n\times(m+1)}$ be a matrix of snapshots, with $x_i = x(t_i) \in \mathbb{C}^n$ and let us, as always in DMD algorithms, assume these snapshots to be the solution of a linear dynamical system as in equation (5.12), restricted to a subspace of dimension r. We are hence assuming that the solution is given by the function

$$x(t) \approx Se^{\Lambda t}S^\dagger z_0 \tag{5.16}$$
$$\implies x_k \approx Se^{\Lambda t_k}S^\dagger z_0 \tag{5.17}$$

where $S \in \mathbb{C}^{n\times r}$ and $\Lambda = diag(\{\alpha_i\}) \in \mathbb{C}^{r\times r}$.

As in Subsection 5.1.3, Equation (5.14), we may rewrite the matrix $\Psi_0^{m\intercal}$ as

$$\Psi_0^{m\intercal} = \begin{bmatrix} x_0^\intercal \\ \vdots \\ x_m^\intercal \end{bmatrix} \approx \Phi(\alpha)B, \tag{5.18}$$

where

$$\Phi(\alpha)_{ij} = \phi_j(\alpha, t_i) = e^{\alpha_j t_i} \text{ for } 1 \le i \le m+1,\ 1 \le j \le r, \tag{5.19}$$
$$B_{ij} = S_{ji}(S^{-1}z_0)_i \qquad \text{for } 1 \le i \le r,\ 1 \le j \le n. \tag{5.20}$$

We can then identify the optimal α_{opt} and B_{opt} as the solution of the optimization problem

$$\min_{\alpha\in\mathbb{C}^r, B\in\mathbb{C}^{r\times n}} \|\Psi_0^{m\intercal} - \Phi(\alpha)B\|_F \tag{5.21}$$

using the variable projection method displayed in subsection 5.1.2.

The optimized *DMD eigenvalues* are then determined by $\lambda_i = \alpha_{opt_i}$ with their associated *optimized DMD modes*

$$\varphi_i = \frac{1}{\|B_{opt}^\intercal(:, i)\|_2} B_{opt}^\intercal(:, i) \tag{5.22}$$

where $B_{opt}^{\intercal}(:, i)$ is the ith column of the matrix $B_{opt}^{\intercal}$.

Setting the amplitudes $\beta_i = \|B_{opt}^{\intercal}(:, i)\|_2$, the reconstructed snapshots are given by

$$\overline{z}_k = \sum_{i=1}^{r} \beta_i e^{\alpha_i t_k} \varphi_i, \tag{5.23}$$

these vectors $\overline{z}_k$ are approximations of the snapshots z_k for each $k \in \{0, 1, ..., m\}$. Hence, identifying the solution α_{opt} and B_{opt} of problem (5.21), the reconstruction of the Optimized DMD modes, values and amplitudes is trivial.

5.2.2 Initialization

The Levenberg-Marquardt algorithm, which is at the heart of the variable projection method used to solve (5.21), requires a good initial guess α_0 in order to find a suitable minimizer. We can hence view the Optimized DMD algorithm as a post-processor for an already good guess, in this subsection, we are going to illustrate how to find out a good initial guess with the trapezoidal rule, as Askham and Kutz suggest in [3]. Assuming the snapshots $\{x_0, x_1, ..., x_m\}$ with the sample times $t_0 < t_1 < ... < t_m$ to be iterates of the trapezoidal rule applied to the linear dynamical system (5.12)

$$\dot{x}(t) = Ax(t),$$

then

$$\begin{aligned} x_{j+1} = x(t_{j+1}) &= x(t_j) + \int_{t_j}^{t_{j+1}} \dot{x}(t)dt \\ &= x_j + \int_{t_j}^{t_{j+1}} Ax(t)dt \\ &= x_j + (t_{j+1} - t_j)\frac{1}{2}(Ax(t_{j+1}) - Ax(t_j)) \\ &= x_j + (t_{j+1} - t_j)\frac{1}{2}A(x(t_{j+1}) - x(t_j)), \end{aligned}$$

and thus

$$\frac{x_{j+1} - x_j}{t_{j+1} - t_j} = \frac{1}{2}A(x(t_{j+1}) - x(t_j)). \tag{5.24}$$

holds for $j \in \{0, 1, ..., m-1\}$.

Stacking the snapshots in the two matrices $\Psi_0^{m-1} = [x_0, ..., x_{m-1}]$ and $\Psi_1^m = [x_1, ..., x_m]$ and defining the matrix $T = diag(t_1 - t_0, ..., t_m - t_{m-1})$, we see from the equation above that the matrix A satisfies

$$(\Psi_1^m - \Psi_0^{m-1})T^{-1} = A\frac{\Psi_1^m - \Psi_0^{m-1}}{2}.$$

Defining the matrices $Y = \frac{\Psi_1^m - \Psi_0^{m-1}}{2}$ and $Z = (\Psi_1^m - \Psi_0^{m-1})T^{-1}$ we can proceed in a similar manner as in Exact DMD: we take the (reduced) SVD of the matrix $Y = U\Sigma V^*$ then from relation (5.2.2), we can define a similarity transform $\tilde{A} \in \mathbb{C}^{r \times r}$ of the matrix A

$$\begin{aligned} Z &= AY && (5.25)\\ &= AU\Sigma V^* && (5.26)\\ \implies ZV\Sigma^{-1} &= AU && (5.27)\\ \implies U^*ZV\Sigma^{-1} &= U^*AU = \tilde{A} && (5.28) \end{aligned}$$

we can then extract the r eigenvalues of the matrix $\tilde{A}$ and use them as initial guess α_0 for the optimized DMD algorithm.

Initialization routine

We now provide the algorithm which gives the initial guess α_0 for the trapezoidal rule starting with a snapshot sequence $\{x_0, x_1, ..., x_m\}$ with sample times $t_0 < t_1 < ... < t_m$.

Step 1: Stack the snapshots in the matrices $\Psi_0^{m-1} = [x_0, x_1, ..., x_{m-1}] \in \mathbb{C}^{n\times m}$, $\Psi_1^m = [x_0, x_2, ..., x_m] \in \mathbb{C}^{n\times m}$, define the matrix $T = diag(t_1 - t_0, ..., t_m - t_{m-1})$ and compute the two matrices

$$Y = \frac{\Psi_1^m - \Psi_0^{m-1}}{2}, \quad Z = (\Psi_1^m - \Psi_0^{m-1})T^{-1}$$

Step 2: Take the (reduced) rank SVD of the matrix Y

$$Y = U\Sigma V^*,$$

where $U \in \mathbb{C}^{n\times r}$, $\Sigma \in \mathbb{C}^{r\times r}$ and $V \in \mathbb{C}^{m\times r}$ and r is the rank of the matrix Y.

Step 3: Define the companion matrix

$$\tilde{A} = U^*AU = U^*ZV\Sigma^{-1}.$$

Step 4: Compute the eigendecomposition of $\tilde{A}$ and return the vector of the r nonzero eigenvalues $\tilde{\alpha} = [\tilde{\alpha}_1, ..., \tilde{\alpha}_r] \in \mathbb{C}^r$.

5.2.3 Optimized DMD algorithm

In this subsection we provide the Optimized DMD algorithm starting with a sequence of snapshots $\{x_0, ..., x_m\}$

Step 1: Use the initialization routine from subsection 5.2.2 to compute an initial guess α_0.

Step 2: Stack the snapshots in the matrix $\Psi_0^m = [x_0, x_1, ..., x_m] \in \mathbb{C}^{n\times(m+1)}$.

Step 3: Solve the optimization problem

$$\min_{\alpha\in\mathbb{C}^r, B\in\mathbb{C}^{r\times n}} \|\Psi_0^{m\intercal} - \Phi(\alpha)B\|_F,$$

where $\Phi(\alpha) \in \mathbb{C}^{(m+1)\times r}$ with $\Phi(\alpha)_{ij} = e^{\alpha_j t_i}$, using the variable projection method (for further information, see subsection 5.1.2).

Step 4: For (α_{opt}, B_{opt}) solving the optimization problem, set the optimized DMD eigenvalues as $\lambda_i = \alpha_{opt_i}$ and the optimized DMD modes as

$$\varphi_i = \frac{1}{\|B_{opt}^\intercal(:, i)\|_2} B_{opt}^\intercal(:, i), \tag{5.29}$$

saving the amplitudes $\beta_i = \|B_{opt}^\intercal(:, i)\|_2$.

5.3 Examples

5.3.1 Periodic sequences

We will now analyze the data set generated in Example 3.4.1, i.e. the data generated by the sum of the two functions

$$f_1(x, t) = \text{sech}(x + 3)e^{2.3it},$$
$$f_2(x, t) = 2\text{sech}(x)\tanh(x)e^{2.8it},$$

with the BOPDMD class from the PyDMD package[1].

First we analyze the data without noise, to assure that it correctly recognizes the modes and eigenvalues. From Figure 5.1 we can recognize that the eigenvalues and modes are correctly identified: the modes are a scaled version of the Ritz values and the continuous eigenvalues are given by $\mu_1 = 2.8i$ and $\mu_2 = 2.3i$.

[1] url: https://pydmd.github.io/PyDMD/bopdmd.html

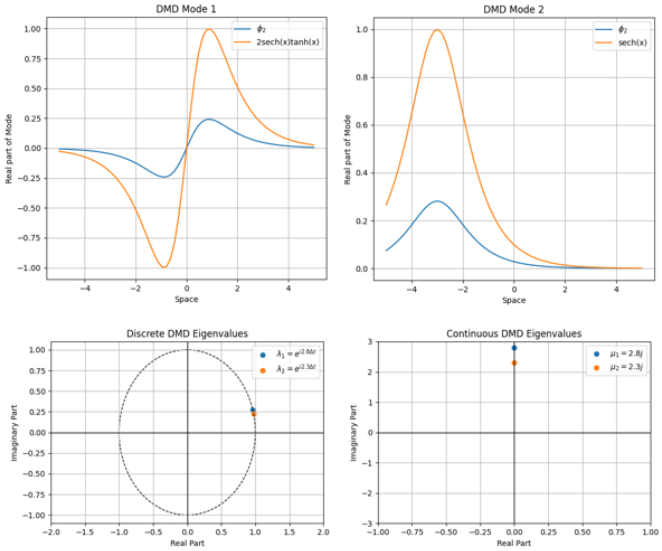

Figure 5.1 The modes and eigenvalues provided by the DMD algorithm fed with perfect data

We now apply the optimized DMD algorithm on the noisy data from Figure 3.5, as we can see in Figure 5.2, the eigenvalues determined are very close to the imaginary axis, hence being way more stable than the ones found by standard DMD in Figure 3.6, this results in the reconstructed dynamics being almost identical to the ground truth data in comparison with standard DMD (see Figure 5.3).

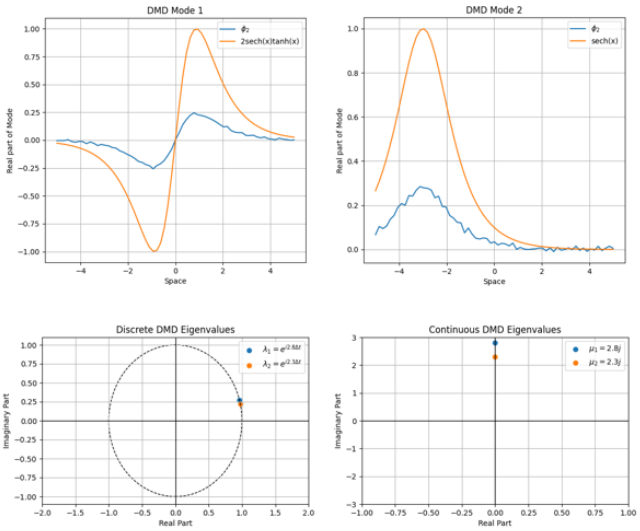

Figure 5.2 The modes and eigenvalues provided by the DMD algorithm fed with noisy data

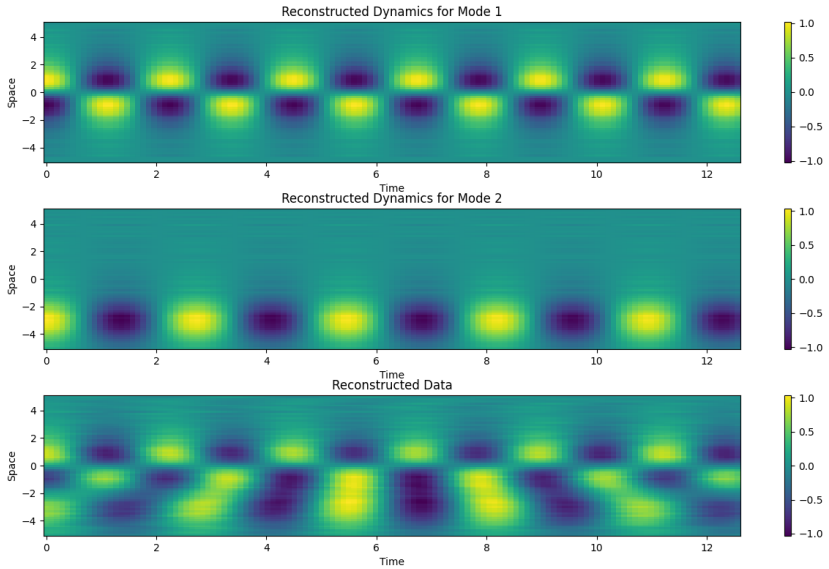

Figure 5.3 The data reconstructed by the DMD algorithm fed with noisy data

6 Extended Dynamic Mode Decomposition (EDMD)

In this section, we are going to outline the extended dynamic mode decomposition method and its kernelized variant developed by Williams, Rowley and Kevrekidis in [43] and [44]. This variant of DMD consists in extending the set of observables fed to the algorithm to a higher basis than the one used to obtain the full states (i.e. the projection maps of the full state components). This is done by choosing a dictionary of observables $\mathcal{D}$ which, if chosen correctly, can span a Koopman almost-invariant subspace, allowing us to better capture the dynamics of the system.

6.1 Extended Dynamic Mode Decomposition

6.1.1 Derivation of the algorithm

The EDMD method requires, in the same manner as Exact DMD from section 3.3, a set of snapshot pairs $\{x_k, y_k\}_{k=1}^m$ that we will organize in the data matrices

$$X = [x_1, x_2, ..., x_m], \quad Y = [y_1, y_2, ..., y_m]$$

where $x_k \in \mathcal{M}$ and $y_k \in \mathcal{M}$ are snapshots of the system state being related by the dynamical system 2.1, i.e. $y_k = f(x_k)$. Again, if the snapshots are given as a single time series, then they are related by the following equation $y_k = f(x_k) = x_{k+1}$. Moreover, EDMD requires a dictionary of observables $\mathcal{D}_N = \{\psi_1, \psi_2, ..., \psi_N\}$, whose span is defined as $\mathcal{F}_{\mathcal{D}_N} \subset \mathcal{F}$, we also define the vector-valued function $\Psi : \mathcal{M} \to \mathbb{R}^{1\times N}$ as

$$\Psi(x) = [\psi_1(x), ..., \psi_N(x)]. \tag{6.1}$$

from now on, this function Ψ the dictionary function. In order to establish a relation with the results of Klus et al. in [38], we are also going to determine the matrices of observables

$$\Psi_X = \begin{bmatrix} \Psi(x_1) \\ \Psi(x_2) \\ \vdots \\ \Psi(x_m) \end{bmatrix}, \quad \Psi_Y = \begin{bmatrix} \Psi(y_1) \\ \Psi(y_2) \\ \vdots \\ \Psi(y_m) \end{bmatrix}. \tag{6.2}$$

Approximating Koopman Operator

As for every other DMD variant, the goal of EDMD is to generate a matrix $K_{EDMD} \in \mathbb{C}^{N\times N}$, which is a finite-dimensional approximation of the Koopman operator $\mathcal{K}$. To this end, let $\psi_a \in \mathcal{F}_{\mathcal{D}_N}$, then by definition, we can rewrite

$$\psi_a(x) = \sum_{k=1}^{N} a_k \psi_k(x) = \Psi(x)a, \tag{6.3}$$

where $a \in \mathbb{R}^N$ is the vector of weights.

We want K_{EDMD} to satisfy the equation

$$\mathcal{K}\psi(x) = (\Psi \circ f)(x)a = \Psi(x)K_{EDMD}a + r(x) \tag{6.4}$$

where we included the residual term $r \in \mathcal{F}/\mathcal{F}_{\mathcal{D}_N}$, because the subspace $\mathcal{F}_{\mathcal{D}_N}$ is typically not Koopman invariant.

In order to determine K_{EDMD}, we minimize the term

$$\begin{aligned} J(K) &= \frac{1}{2}\sum_{k=1}^{M} |r(x_k)|^2 \\ &= \frac{1}{2}\sum_{k=1}^{M} |((\Psi \circ f)(x_k) - \Psi(x_k)K)a|^2 \\ &= \frac{1}{2}\sum_{k=1}^{M} |(\Psi(y_k) - \Psi(x_k)K)a|^2. \end{aligned} \tag{6.5}$$

Since K_{EDMD} has to minimize this term for all vectors $a \in \mathbb{R}^N$, we can rewrite the the minimization term J as

$$\min_{K\in\mathbb{C}^{N\times N}} J = \min_{K\in\mathbb{C}^{N\times N}} \frac{1}{2}\|\Psi_Y - \Psi_X K\|_F^2, \tag{6.6}$$

as we have already seen in the previous chapters, the solution to this minimization problem is given by

$$K_{EDMD} = \Psi_X^{\dagger}\Psi_Y,$$

which is exactly the result provided in equation (13) in [38].

Using the equality $\Psi_X^{\dagger} = (\Psi_X^*\Psi_X)^{\dagger}\Psi_X^*$, we can rewrite this matrix K_{EDMD} in the following way

$$K_{EDMD} = \Psi_X^{\dagger}\Psi_Y \tag{6.7}$$

$$= (\Psi_X^*\Psi_X)^{\dagger}\Psi_X^*\Psi_Y \tag{6.8}$$

$$= G^{\dagger}A \tag{6.9}$$

where

$$G = \frac{1}{m}\Psi_X^*\Psi_X = \frac{1}{m}\sum_{k=1}^{m} \Psi(x_k)^*\Psi(x_k), \tag{6.10}$$

$$A = \frac{1}{m}\Psi_X^*\Psi_Y = \frac{1}{m}\sum_{k=1}^{m} \Psi(x_k)^*\Psi(y_k). \tag{6.11}$$

For (μ_j, ξ_j) being an eigenpair of the matrix K_{EDMD}, i.e. $K\xi_j = \mu_j\xi_j$, the approximation of a Koopman eigenfunction is given by

$$\varphi_j(x) = \Psi(x)\xi_j, \tag{6.12}$$

since, by equation (6.4), the next iteration is given by

$$\mathcal{K}\varphi_j(x) = \Psi(x)K_{EDMD}\xi_j + r(x) = \mu_j\Psi(x)\xi_j + r(x) = \mu_j\varphi_j(x) + r(x)$$

Computing the Koopman modes

In the same way as we did in Example 2.2.1, we recall that we can generate the full-state vector-valued observable function $g : \mathcal{M} \to \mathbb{C}^n$, by stacking the projection maps $g_i : \mathcal{M} \to \mathbb{C}$

$$g(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_n(x) \end{bmatrix} = \begin{bmatrix} e_1^*x \\ e_2^*x \\ \vdots \\ e_n^*x \end{bmatrix},$$

where $e_i \in \mathbb{R}^n$ is the ith unit vector.

We now assume that $g_i \in \mathcal{F}_{\mathcal{D}_N}\ \forall i \in \{1, ..., n\}$, such that we can rewrite these observables as $g_i(x) = \sum_{k=1}^{N} \phi_k(x) b_{i_k} = \Psi(x) b_i = b_i^\intercal \Psi(x)^\intercal$ and hence the full-state function g can be reformulated as

$$g(x) = B^\intercal \Psi(x)^\intercal = (\Psi(x)B)^\intercal \tag{6.13}$$

where $B = [b_1, ..., b_n] \in \mathbb{C}^{N \times n}$.

We now define the the vector-valued eigenfunction $\Phi : \mathcal{M} \to \mathbb{C}^{1 \times N}$ as

$$\begin{aligned}
\Phi(x) &= [\varphi_1(x), ..., \varphi_N(x)] && (6.14)\\
&= [\Psi(x)\xi_1, \Psi(x)\xi_2, ..., \Psi(x)\xi_N] && (6.15)\\
&= \Psi(x)[\xi_1, \xi_2, ..., \xi_N] && (6.16)\\
&= \Psi(x)\Xi, && (6.17)
\end{aligned}$$

where $\Xi = [\xi_1, \xi_2, ..., \xi_N] \in \mathbb{C}^{N \times N}$. Because Ξ is the matrix of eigenvectors of K_{EDMD}, its inverse is given by

$$\Xi^{-1} = W^* = \begin{bmatrix} w_1^* \\ w_2^* \\ \vdots \\ w_N^*, \end{bmatrix} \tag{6.18}$$

where $w_i \in \mathbb{C}^{1 \times N}$ is the ith left eigenvector of the matrix K associated with the eigenvalue μ_i, i.e. satisfying $w_i^* K_{EDMD} = w_i^* \mu_i$, and appropriately scaled such that $w_i^* \xi_j = \delta_{ij}$, with δ_{ij} being the Kronecker delta. Combining equations (6.13) and (6.18), we rewrite the full-state function g as

$$\begin{aligned}
g(x) &= (\Psi(x)B)^\intercal = (\Phi(x)W^*B)^\intercal \\
&= (W^*B)^\intercal \Phi(x)^\intercal = V\Phi(x)^T \\
&= \sum_{k=1}^{N} v_k \varphi_k(x) && (6.19)
\end{aligned}$$

where $V = [v_1, ..., v_N] = (W^*B)^\intercal$ is the matrix of approximated Koopman modes and the kth mode is given by $v_k = (w_k^* B)^\intercal$.

6.1.2 Convergence to a Galerkin method

In this context, a Galerkin method is a weighted-residual method where the residual is orthogonal to the span of $\mathcal{D}_N$. The elements of X (the snapshots) are drawn independently from a distribution on $\mathcal{M}$ with the probability density ρ. We assume that the observable space is given by $\mathcal{F} = L^2(\mathcal{M}, \rho)$, such that the inner products in the Galerkin method converge, and that ρ is one of the natural measures associated with the underlying dynamical system.

If EDMD was a Galerkin method, the components of the matrices A and G would be then defined as

$$\hat{G}_{ij} = \int_{\mathcal{M}} \overline{\psi_i(x)} \psi_j(x) \rho(x) \mathrm{dx} = \langle \psi_i, \psi_j \rangle_\rho \tag{6.20}$$

$$\hat{A}_{ij} = \int_{\mathcal{M}} \overline{\psi_i(x)} \psi_j(f(x)) \rho(x) \mathrm{dx} = \langle \psi_i, \mathcal{K}\psi_j \rangle_\rho \tag{6.21}$$

where $\langle \cdot, \cdot \rangle_\rho$ is the inner product and the finite- dimensional Galerkin approximation of the Koopman operator would be $\hat{K} = \hat{G}^{-1}\hat{A}$. For a finite number of snapshots m, the components of G and A are given by their respective equations (6.10) and (6.11), so the ijth components of G and A are given by

$$G_{ij} = \frac{1}{m} \sum_{k=1}^{m} \overline{\psi_i(x_k)} \psi_j(x_k), \tag{6.22}$$

$$A_{ij} = \frac{1}{m} \sum_{k=1}^{m} \overline{\psi_i(x_k)} \psi_j(y_k), \tag{6.23}$$

which are, respectively, the sample mean of $\psi_i(x)^*\psi_j(x)$ and the sample mean of $\psi_i(x)^*\psi_j(f(x))$. By the law of large numbers, we know that the sample means will converge to the expected values as m becomes sufficiently large, therefore it holds

$$\lim_{m\to\infty} G_{ij} = \lim_{m\to\infty} \frac{1}{m}\sum_{k=1}^{m} \overline{\psi_i(x_k)}\psi_j(x_) = \int_{\mathcal{M}} \overline{\psi_i(x)}\psi_j(x)\rho(x)\mathrm{dx} = \hat{G}_{ij} \tag{6.24}$$

$$\lim_{m\to\infty} A_{ij} = \lim_{m\to\infty} \frac{1}{m}\sum_{k=1}^{m} \overline{\psi_i(x_k)}\psi_j(y_k) = \int_{\mathcal{M}} \overline{\psi_i(x)}\psi_j(f(x))\rho(x)\mathrm{dx} = \hat{A}_{ij} \tag{6.25}$$

Hence as $M \to \infty$, convergence means that $\mathbb{K}_{EDMD}$ approaches a matrix representation of $\mathbb{P}_{\mathcal{D}_N}\mathcal{K}\mathbb{P}_{\mathcal{D}_N}$, where $\mathbb{P}_{\mathcal{D}_N}$ is the orthogonal projection on $\mathcal{D}_N$.

Therefore, the output of the EDMD procedure will converge to the output of a Galerkin method. With randomly distributed snapshots $\{x_k\}_{k=1}^m$, the needed integrals are computed using Monte Carlo integration and the rate of convergence $O(m^{-\frac{1}{2}})$.

Using other quadrature rules

It is possible to achieve a better convergence rate by using other sampling method for the snapshots, such as grids, using quadrature rules with quadrature weights $\{w_k\}_{k=1}^m$ contained in the matrix $W = diag(\{w_k\}_{k=1}^m)$, such that the integrals can be approximated as

$$\hat{G}_{ij} = \int_{\mathcal{M}} \overline{\psi_i(x)}\psi_j(x)\rho(x)\mathrm{dx} \tag{6.26}$$

$$\approx \sum_{k=1}^{m} w_k\overline{\psi_i(x_k)}\psi_j(x_k) = [\overline{\psi_i(x_1)}, ..., \overline{\psi_i(x_m)}]W\begin{bmatrix}\psi_j(x_1)\\ \vdots \\ \psi_j(x_m)\end{bmatrix}, \tag{6.27}$$

$$\hat{A}_{ij} = \int_{\mathcal{M}} \overline{\psi_i(x)}\psi_j(f(x))\rho(x)\mathrm{dx} \tag{6.28}$$

$$\approx \sum_{k=1}^{m} w_k\overline{\psi_i(x_k)}\psi_j(y_k) = [\overline{\psi_i(x_1)}, ..., \overline{\psi_i(x_m)}]W\begin{bmatrix}\psi_j(y_1)\\ \vdots \\ \psi_j(y_m)\end{bmatrix}, \tag{6.29}$$

and the matrices G and A can be computed as $G = \Psi_X^* W \Psi_X$ and $A = \Psi_X^* W \Psi_Y$, if the quadrature rule converges, then, in the same way as equations (6.25) and (6.24) hold,

$$\lim_{m\to\infty} G_{ij} = \langle\psi_i, \psi_j\rangle_\rho, \lim_{m\to\infty} A_{ij} = \langle\psi_i, \mathcal{K}\psi_j\rangle_\rho \tag{6.30}$$

holds too. Notice that in the case discussed above, the quadrature weights resulted simply in being $\frac{1}{m}$.

6.1.3 Relation with DMD

We will now show that EDMD is equivalent to DMD for a specific choice of the dictionary $\mathcal{D}_N$. We recall from Section 3.3 (and [41]) that the exact DMD modes are given as the eigenvectors of the matrix

$$K_{DMD} = YX^\dagger \tag{6.31}$$

where the jth mode is associated with the jth eigenvalue μ_j of the matrix K_{DMD}. Let the dictionary be given by $\mathcal{D}_N = \{\psi_1(x), ..., \psi_n(x)\}$, defining the functions $\psi_k(x) = e_i^* x \ \forall k \in \{1, ..., n\}$.

Because the elements of the full-state observable are the dictionary elements ($g_i(x) = \psi_i(x)$), the matrix B from equation (6.13) is the identity matrix, i.e. $B = I$, the kth approximated Koopman mode is hence given by $s_k^\intercal = w_k^*$, with w_k^* satisfying the equality $w_k^* K = w_k^* \mu_k$. Furthermore $G^\intercal = \frac{1}{m}XX^*$ and $A^\intercal = \frac{1}{m}YX^*$, then it holds

$$K_{EDMD}^\intercal = A^\intercal(G^\intercal)^\dagger = YX^*(XX^*)^\dagger = YX^\dagger = K_{DMD}. \tag{6.32}$$

Thus, for the approximated Koopman mode v_k, it holds $K_{DMD}v_k = (v_k^\intercal K_{DMD}^\intercal)^\intercal = (w_k^* K_{EDMD})^\intercal = (w_k^* \mu_k)^\intercal = \mu_k v_k$, the Koopman modes approximated by the EDMD method are thus the exact DMD modes.

6.1.4 Choice of the dictionary

Without knowledge of the underlying dynamical system f and, possibly, of the manifold $\mathcal{M}$, the choice of optimal dictionary is a very difficult task. Some possible choices of the dictionary $\mathcal{D}_N$ by Williams et al. in [43] are: polynomials ([22]), Fourier modes ([40]), radial basis functions ([42]), and spectral elements ([23]). If the manifold $\mathcal{M}$ is not known, we can typically define the domain Ω of the observables $\psi_k : \Omega \to \mathbb{C}$ as the n-dimensional "box" containing containing all the snapshots contained in the matrices X and Y, this way we assure that $\mathcal{M} \subset \Omega$. We now choose the elements of $\mathcal{D}_N$, such that they form a basis of $\tilde{\mathcal{F}}_{\mathcal{D}_N} \subset \tilde{\mathcal{F}}$, where $\tilde{\mathcal{F}}$ is the space of all observables whose domain is Ω, i.e of the functions mapping $\Omega \to \mathbb{C}$. Because $\mathcal{M} \subset \Omega$, it also holds $\mathcal{F} \subset \tilde{\mathcal{F}}$, we are thus able to apply the EDMD method with such a dictionary $\mathcal{D}_N$, however, we cannot assure that the components of $\mathcal{D}_N$ also form a basis of $\mathcal{F}_{\mathcal{D}_N}$, because there might be redundancies. In [43], Williams at al. recommend 3 particular dictionaries: Hermite Polynomials, radial basis functions and discontinuous spectral elements

1. **Hermite polynomials.** The simplest of the three recommended sets: it is best suited for systems on $\mathbb{C}^n$, when the snapshot data contained in the matrix X is normally distributed. The observables forming $\mathcal{D}_N$ are products of the Hermite polynomials in one dimension (for example $\psi_{2,3,1}(x) = H_2(x_1)H_3(x_2)H_1(x_3)$, where H_i is the ith Hermite polynomial and $x = (x_1, x_2, x_3)$). The two big advantages of this dictionary are the simplicity of implementation and the orthogonality of the Hermite polynomials with respect to gaussian weight (see [2]), this results in $G = \frac{1}{m}\sum_{k=1}^{m} \Psi(x_k)^* \Psi(x_k)$ being diagonal, if the snapshots x_k are drawn from a normal distribution, which is of course beneficial in terms of computational effort.

2. **Discontinuous spectral elements.** To use this set of functions, we first need to define a set of B_N boxes $\{\mathcal{B}_i\}_{i=1}^{B_N}$, such that $\bigcup_{i=1}^{B_N} \mathcal{B}_i \supset \mathcal{M}$. On each box $\mathcal{B}_i$ we define K_i (suitably transformed) Legendre polynomials.

   For example, in one dimension, each of the $\sum_{i=1}^{B_N} K_i$ observable functions is given by

   $$\psi_{ij}(x) = \begin{cases} L_j(\xi) & x \in \mathcal{B}_i = [x_{i-1}, x_i], \\ 0 & \text{otherwise}, \end{cases} \tag{6.33}$$

   where $L_j(x)$ is the jth Legendre polynomial and ξ is x transformed such that the "edges" of the box $\mathcal{B}$, given by x_{i-1} and x_i correspond to $\xi = \pm 1$, i.e $\xi(x) = 2\frac{x - x_{i-1}}{x_i - x_{i-1}} - 1$.

   By defining the observable functions in this manner, the matrix G results in being block-diagonal (because for 2 functions ψ_{ij} and ψ_{kl} with $k \neq i$, it will hold $\psi_{ij}(x)\psi_{kl}(x) = 0$), which makes the computation of its pseudoinverse $G^\dagger$ computationally easier. **Choice of the boxes $\mathcal{B}_i$:** The choice of the boxes can be a very hard task as well: Williams et al. use a method similar to the one used by GAIO (see the work of M. Dellnitz, G. Froyland and O. Junge in [13]). The procedure consists in the following steps: at first, all the snapshots in X and Y are contained within a single user-selected box, $B_1^{(0)}$. If this box contains more than a pre-specified number of data points, it is subdivided into 2^n domains of equal Lebesgue measure (e.g., in one dimension, $B_1^{(0)} = B_1^{(1)} \cup B_2^{(1)}$). We then proceed recursively: if any of $B_i^{(1)}$ contain more than a pre-specified number of points, then they too are subdivided; this proceeds until no box has an “overly large” number of data points. Any $B_i^{(j)}$ that do not contain any data points are pruned, which after j iterates leaves the set of subdomains, $\{B_i^{(j)}\}$, on which we define the Legendre polynomials. This procedure works at best for the dimension of the snapshots n being smaller than the amount of snapshots m, i.e. $n < m$.

3. **Radial Basis Functions (RBFs):** This set of functions is particularly appealing, because it does not require a computation mesh or grid ([15]), resulting in it being particularly effective for manifolds

$\mathcal{M}$ with complex geometries. Many different RBFs could be effective, but one particularly useful set of RBFs are the thin plate splines ([39, 42]) because they do not require any scaling parameter (in contrast to Gaussian RBFs for example). However, we still must choose the "centers" about which the RBFs are defined, which we do with k-means clustering ([6]) with a pre-specified value of k on the combined data set. The density of the RBF centers appears to be directly related to the density of data points, which is, intuitively, a reasonable method for distributing the RBF centers as regions with more samples will also have more spatial resolution.

6.1.5 EDMD algorithm

We now provide the EDMD algorithm for a given set of snapshot pairs $\{(x_k, y_k)\}_{k=1}^{m}$ and a dictionary of observable functions $\mathcal{D}_N = \{\psi_1, \psi_2, ..., \psi_N\}$, its related dictionary function $\Psi(x) = [\psi_1(x), \psi_2(x)..., \psi_N(x)]$ and the matrix B satisfying equation $g(x) = (\Psi(x)B)^\intercal$ for g being the full-state vector-valued observable function.

Step 1: Compute the matrices of observables

$$\Psi_X = \begin{bmatrix} \Psi(x_1) \\ \Psi(x_2) \\ \vdots \\ \Psi(x_m) \end{bmatrix}, \quad \Psi_Y = \begin{bmatrix} \Psi(y_1) \\ \Psi(y_2) \\ \vdots \\ \Psi(y_m) \end{bmatrix}.$$

Step 2: Compute the matrices

$$G = \frac{1}{m}\Psi_X^*\Psi_X = \frac{1}{m}\sum_{k=1}^{m}\Psi(x_k)^*\Psi(x_k),$$
$$A = \frac{1}{m}\Psi_X^*\Psi_Y = \frac{1}{m}\sum_{k=1}^{m}\Psi(x_k)^*\Psi(y_k).$$

to assemble the EDMD approximation of the Koopman operator

$$K_{EDMD} = G^\dagger A.$$

Step 3: Compute the eigendecomposition of the matrix K_{EDMD}

$$K_{EDMD} = \Xi M W^*$$

and compute the matrix of Koopman modes $V = [v_1, ..., v_N] = (W^*B)^\intercal$.

6.2 Kernel-based Extended Dynamic Mode Decomposition

In the previous section, we assumed that the number of snapshots m is large compared to the number of observables N, making the computation of $K_{EDMD} \in \mathbb{C}^{N\times N}$ by Equation (6.9) feasible. However, as we have also seen in the previous chapters, the opposite is the more common case, where even the dimension of a snapshot n is large compared to the number of snapshots m, resulting in $N > n >> m$. The difficulty of this case is that the EDMD method requires $O(N_k^2 m)$ time to assemble K and $O(N_k^3)$ time to compute the modes and eigenvalues. However, the value of N for a "rich" set of basis functions grows rapidly as the dimension of state space increases, which makes this approach computationally infeasible.

Example 1. Consider the case where $\mathcal{F}_N$ is the space of all (multivariate) polynomials on $\mathbb{R}^{256}$ with degree up to 20. Then, N 10^{30}, which is far too large for practical computations [6]. This explosion in the size of the problem is common in machine learning applications, and is one facet of the curse of dimensionality.

In this section, following the work of Williams et al. in [44], we will outline the kernel-based EDMD method, which makes the case $N >> m$ tractable, by computing a companion matrix $\hat{K} \in \mathbb{R}^{m\times m}$ in a similar manner as Schmid did in [36] (for further information, see Section 3.2).

6.2.1 Derivation of the algorithm

With the next theorem, we are going to define the companion matrix $\hat{K}$, show that it shares the eigenvalues with K_{EDMD} and establish the relation between their eigenvectors

Theorem 10 *Let the SVD of the matrix Ψ_X from Equation (6.2) be $\Psi_X = U\Sigma V^*$, where $U \in \mathbb{C}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{C}^{N\times m}$. The pair $(\mu, \hat{v})$ with $\mu \neq 0$ be an eigenpair of the companion matrix*

$$\hat{K} = (\Sigma^\dagger U^*)(\Psi_Y \Psi_X^*)(U\Sigma^\dagger) = (\Sigma^\dagger U^*)\hat{A}(U\Sigma^\dagger), \tag{6.34}$$

if and only if (μ, v) with $v = V\hat{v}$ is an eigenpair of K_{EDMD}.

Proof: We want to prove the following equivalence

$$\hat{K}\hat{v} = \mu\hat{v} \text{ with } \mu \neq 0 \Leftrightarrow K_{EDMD}v = \mu v \text{ with } v = V\hat{v}$$

" $\Longrightarrow$ " Let the pair $(\mu, \hat{v})$ with $\mu \neq 0$ be an eigenpair of $\hat{K}$, then, with $V = (\Sigma^\dagger U^* \Psi_X)^* = \Psi_X^* U \Sigma^\dagger$, the following equation holds:

$$\begin{aligned} K_{EDMD}v &= (\Psi_X^\dagger \Psi_Y)V\hat{v} \\ &= (V\Sigma^\dagger U^* \Psi_Y)(\Psi_X^* U\Sigma^\dagger)\hat{v} \\ &= V(\Sigma^\dagger U^*(\Psi_Y \Psi_X^*)U\Sigma^\dagger))\hat{v} \\ &= V\hat{K}\hat{v} \\ &= \mu V\hat{v} = \mu v. \end{aligned}$$

" $\Longleftarrow$ " Now let (μ, v) be an eigenpair of K_{EDMD}, then

$$\begin{aligned} \hat{K}\hat{v} &= (\Sigma^\dagger U^*)(\Psi_Y \Psi_X^*)(U\Sigma^\dagger)(V^* v) \\ &= ((\Sigma^\dagger U^*)(\Psi_Y \Psi_X^*)(\Psi_X^*)^\dagger v \\ &= V^*(V(\Sigma^\dagger U^*)\Psi_Y)v \\ &= V^*(V\Psi_X^\dagger \Psi_Y)v \\ &= V^* K_{EDMD}v = \mu V^* = \mu\hat{v}. \end{aligned}$$

Note that the identity $(\Psi_X^\intercal)(\Psi_X^\intercal)^\dagger v = v$ holds because $\mu \neq 0$, so v lies in the image of $\Psi_X^\dagger$ and therefore the image of V. □

Thus, if we could form $\hat{K}$, then we could compute the non-zero eigenvalues and the corresponding left and right eigenvectors of K_{EDMD}, which would yield the desired approximations of the Koopman eigenvalues, eigenfunctions and modes (see Section 6.1.1). However,this has simply moved the computational bottleneck from the eigendecomposition of K_{EDMD} to computing the SVD of Ψ_X, which is an $O(Nm^2)$ task. While this is an improvement, due to the (possibly extremely large) size of N, even linear dependence on the number of observables will make the resulting problem computationally intractable. In order to develop a more efficient algorithm, Williams et al. use the following *method of snapshots* in [44].

Method of snapshots: The matrix $\hat{K}$ from Equation (6.34) can be obtained from matrices whose elements are inner products in feature space evaluated at pairs of data points. To prove this, we first define the matrix $\hat{G} = \Psi_X \Psi_X^*$ and reuse the matrix $\hat{A} = \Psi_Y \Psi_X^*$ from (6.34). The ijth elements of $\hat{G}$ and $\hat{A}$ are given by

$$\hat{G}_{ij} = \Psi(x_i)\Psi(x_j)^*, \quad \hat{A}_{ij} = \Psi(y_i)\Psi(x_j)^*. \tag{6.35}$$

Now, with the SVD $\Psi_X = U\Sigma V^*$, for the matrix $\hat{G}$ holds $\hat{G} = (U\Sigma V^*)(V\Sigma U^*) = U\Sigma^2 U^*$; therefore if we are able to compute the matrices $\hat{G}$ and $\hat{A}$, then we can obtain the matrices U and Σ from the eigendecomposition of $\hat{G}$ and assemble $\hat{K}$ from Equation (6.34). Because the components of $\hat{G}$ and $\hat{A}$ are given by the inner product of the feature map Ψ evaluated at the given data points, with the following *kernel trick*, we will highlight a computationally efficient way to compute these inner products [5, 8, 32, 37].

Definition 11 (Kernel function) *A kernel function has the form $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$, and is associated with a feature map $\Psi : \mathcal{M} \to \mathbb{R}^{1\times N}$, which maps from state space to feature space. The relationship between f and Ψ is $f(x_i, x_j) = \Psi(x_i)\Psi(x_j)^*$ for all $x_i, x_j \in \mathcal{M}$.*

The big advantage of these kernel functions is that, in order to compute the inner product of the feature map, being in the feature space $\mathbb{R}^N$, we only need to compute the inner product in the state space $\mathbb{R}^n$, which results to be very beneficial in terms of computational costs.

Example (quadratic kernel): The simplest example of a kernel function is given by $f : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ defined as

$$\begin{aligned} f(x, y) &= (1 + x^\intercal y)^2 \\ &= (1 + x_1y_1 + x_2y_2)^2 \\ &= 1 + 2x_1y_1 + 2x_2y_2 + 2x_1x_2y_1y_2 + x_1^2y_1^2 + x_2^2y_2^2 \\ &= \Psi(x)\Psi(y)^\intercal \end{aligned}$$

where $\Psi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]$. As already mentioned, while the results are identical, calculating $f(x, y)$ requires an inner product in $\mathbb{R}^2$, while calculating it directly as $\Psi(x)\Psi(y)^\intercal$ requires an inner product in $\mathbb{R}^6$. With the following definition we define the polynomial kernel in a general manner.

Definition 12 (Polynomial kernel) *Polynomial kernel functions have the form*

$$f(x, y) = \left(1 + \frac{x^\intercal y}{d}\right)^\alpha,$$

which are equivalent to the inner products of all polynomials up to and including terms of degree α, but require only $O(n)$ time to compute. The parameter d is used to scale the data, affecting the influence of higher order terms in the expansion. This choice of kernel, similarly to the Hermite polynomials in the previous section, is an approach conceptually related to the approximation of the Koopman operator via a Taylor expansion.

Definition 13 (Gaussian kernel) *Gaussian kernel functions have the form*

$$f(x, y) = exp\left(-\frac{\|x - y\|_2^2}{\sigma^2}\right)$$

and are often used in Machine learning applications [6, 12, 37]. Similarly to d, σ is a scaling parameter which depends on characteristic lengths of the underlying system.

It can be proven that the Gaussian kernel is indeed a kernel function using Bochner's theorem, stating that a function K that can be rewritten as $K(x, y) = f(\|x - y\|_2)$ is a kernel function if and only if the Fourier transform of f is nonnegative. We now prove that the Fourier transform of the Gaussian function $f(x) = exp\left(-\frac{x^2}{\sigma^2}\right)$ respects this condition:

$$\begin{aligned} \mathcal{F}_X\left(e^{-\frac{x^2}{\sigma^2}}\right)(k) &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} e^{-2\pi ikx} dx \\ &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} (\cos(2\pi kx) - i\sin(2\pi kx))dx \\ &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} \cos(2\pi kx)dx - i\int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} \sin(2\pi kx))dx \\ &= \int_{-\infty}^{\infty} e^{-\frac{x^2}{\sigma^2}} \cos(2\pi kx)dx, \end{aligned}$$

where the second term disappeared in the last equality because $e^{-\frac{x^2}{\sigma^2}}\sin(2\pi kx)$ is an odd function. The remaining term is given by Abramowitz and Stegun in [2] (Equation (7.4.6)), such that the Fourier transform is given by:

$$\mathcal{F}_X\left(e^{-\frac{x^2}{\sigma^2}}\right)(k) = \sqrt{\pi\sigma^2}e^{-\pi^2k^2\sigma^2} \geq 0.$$

The Gaussian kernel is thus a kernel function, even though that the corresponding feature map Ψ is implicitly defined. With the kernel trick we are hence able to obtain the matrices $\hat{G}$ and $\hat{A}$ and thus also U and Σ, needed to construct $\hat{V}$; however, to compute the eigenvectors of K_{EDMD}, the matrix V is also needed. With the following remark, we are going to show that the Koopman eigenfunctions can be expressed in terms of inner products, making the Kernel trick possible.

Remark: As we have seen in Equation (6.12), the Koopman eigenfunction can be calculated via the eigenvectors of the matrix K_{EDMD}. From Theorem 10 we also know that, for $\hat{V} = [\hat{v}_1, ..., \hat{v}_m]$ being the matrix of eigenvectors of $\hat{K}$, the matrix $\Xi = V\hat{V}$ is the matrix containing the eigenvectors of K_{EDMD}. From Equation (6.17), it hence follows

$$\Phi_X = \Psi_X V\hat{V} = (U\Sigma V^*)V\hat{V} = U\Sigma\hat{V} \tag{6.36}$$

where we recall the definition $[\Phi(x_1)^\intercal, \Phi(x_2)^\intercal, ..., \Phi(x_m)^\intercal]$; we are thus able to reconstruct the eigenfunctions using only inner products.

Approximate Koopman eigenfunctions The kth numerically approximated Koopman eigenfunction from (6.12) can be evaluated at new data points. Recalling $V = \Psi_X^* U\Sigma^\dagger$ and substituting $v_k = V\hat{v}_k$ yields

$$\begin{aligned}\varphi_k(x) &= \Psi(x)(V\hat{v}_k) = (\Psi(x)\Psi_X^*)(U\Sigma^\dagger\hat{v}_k)\\ &= (\Psi(x)[\Psi(x_1)^*, ..., \Psi(x_m)^*])(U\Sigma^\dagger\hat{v}_k)\\ &= [f(x,x_1), f(x,x_2), ..., f(x,x_m)](U\Sigma^\dagger\hat{v}_k),\end{aligned}$$

we are hence able to use the kernel function f to approximate the Koopman eigenfunctions.

Approximate Koopman modes

In the previous section, we assumed we could represent the full-state observable as g as a linear representation of the observable functions contained in the dictionary $\mathcal{D}_N$ as in Equation (6.13), however this can't be done now, because we didn't define the observables ψ_k explicitly. An alternative way is to find the matrix P minimizing the reconstruction error of the full-state data with then approximated Koopman eigenfunction, i.e. solving

$$\min_{P\in\mathbb{C}^{N\times n}} \|X^\intercal - \Phi_X P\|_F^2. \tag{6.37}$$

This solution is given, as we have already seen, by $P = \Phi_x^\dagger X^\intercal = \hat{V}^{-1}\Sigma^\dagger U^* X^\intercal$, provided that $\hat{V}$ has full rank, in this case its inverse is given by

$$\hat{V}^{-1} = \begin{bmatrix}\hat{w}_1^*\\ \hat{w}_2^*\\ \vdots\\ \hat{w}_m^*\end{bmatrix} \tag{6.38}$$

where $\hat{w}_k^*$ is the kth left eigenvector of $\hat{K}$ scaled such that $w_i^* v_j = \delta_{ij}$, the kth Koopman mode p_k is hence given by

$$p_k = (w_k^*\Sigma^\dagger U^* X^\intercal)^\intercal$$

6.2.2 kEDMD algorithm

We now provide the kernel-based EDMD algorithm for a given set of snapshot pairs $\{(x_k, y_k)\}_{k=1}^m$ stored in the matrices X and Y and a kernel function $f : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$.

Step 1: Compute the components of the matrices $\hat{G}$ and $\hat{A}$ with the kernel function

$$\hat{G}_{ij} = f(x_i, x_j), \quad \hat{A}_{ij} = f(y_i, x_j)$$

and the eigendecomposition of $\hat{G}$

$$\hat{G} = U\Sigma^2 U^{-1}.$$

Step 2: Compute the companion matrix

$$\hat{K} = (\Sigma^\dagger U^*)(\Psi_Y \Psi_X^*)(U\Sigma^\dagger) = (\Sigma^\dagger U^*)\hat{A}(U\Sigma^\dagger)$$

and its eigendecomposition

$$\hat{K} = \hat{V} M \hat{V}^{-1}.$$

Step 3: Compute the matrix of eigenfunctions

$$\Phi_X = U\Sigma\hat{V}$$

and compute the matrix of Koopman modes $V = [v_1, ..., v_N] = (W^* B)^\intercal$.

Step 4: Compute the matrix of Koopman modes

$$P = \hat{V}^{-1}\Sigma^\dagger U^* X^\intercal.$$

6.3 Examples

6.3.1 Linear Dynamical Systems

Still following Williams et al. in [43], we establish a connection with the 2-dimensional analytical linear example from Subsection 2.2.1: we consider the matrix

$$A = \begin{bmatrix} 0.17 & 0.36 \\ -0.24 & 0.83 \end{bmatrix}$$

and the linear dynamical system

$$x_{n+1} = Ax_n, \ \forall n \in \mathbb{N}_0$$

the first 8 Koopman eigenfunctions and eigenvalues are visualized in Figure 2.1. We apply EDMD and chose the dictionary $\mathcal{D}_N$ to be given by the direct product of Hermite polynomials in x (first component) and in y that include up to the fourth-order terms in x and y, i.e. $\mathcal{D}$ is given by the $N = (4+1)^2 = 25$ functions

$$\begin{aligned}
\mathcal{D}_{25} &= \{\psi_1, \psi_1, \psi_2, ...\psi_{25}\} \\
&= \{H_0(x)H_0(y),\ H_1(x)H_0(y),\ H_2(x)H_0(y),\ H_3(x)H_0(y),\ H_4(x)H_0(y), \\
&\quad H_0(x)H_1(y),\ H_1(x)H_1(y),\ H_2(x)H_1(y),\ H_3(x)H_1(y),\ H_4(x)H_1(y), \\
&\quad H_0(x)H_2(y),\ H_1(x)H_2(y),\ H_2(x)H_2(y),\ H_3(x)H_2(y),\ H_4(x)H_2(y), \\
&\quad H_0(x)H_3(y),\ H_1(x)H_3(y),\ H_2(x)H_3(y),\ H_3(x)H_3(y),\ H_4(x)H_3(y), \\
&\quad H_0(x)H_4(y),\ H_1(x)H_4(y),\ H_2(x)H_4(y),\ H_3(x)H_4(y),\ H_4(x)H_4(y)\}.
\end{aligned}$$

Moreover, the data $X \in \mathbb{R}^{2\times 100}$ consists of 100 normally distributed initial conditions x_i and their images $y_i = Ax_i$ contained in the matrix $Y \in \mathbb{R}^{2\times 100}$. This choice of data and dictionary is due to the orthogonality of the Hermite polynomials with respect to the weight function $\rho(x) = e^{-\|x\|^2}$, which is implicit in the

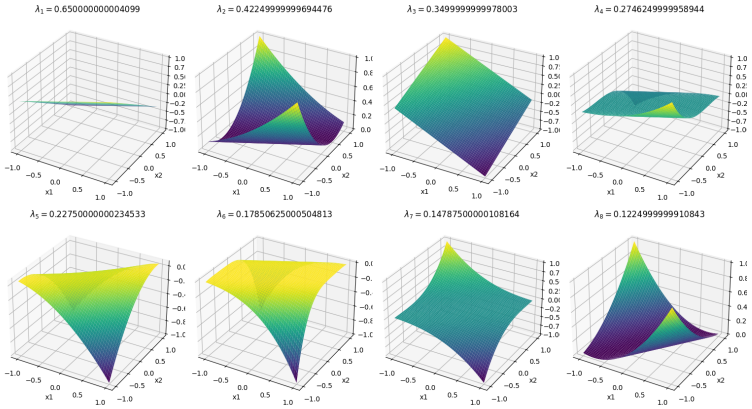

Figure 6.1 First 8 EDMD eigenfunctions

sampling strategy, this way, the calculation of the matrices G and A to apply the EDMD algorithm is computationally cheaper.

In Figure 6.1 we visualize the first 8 eigenfunctions together with their associated eigenvalues, the figure is identical to Figure 2.1 up to the sixth eigenfunction, which differs from the analytical one due to the orientation of the eigenvector v_6; the eigenvalues are exact up to an error of 10^{-11}.

Remark: The exact DMD algorithm is fully able to reconstruct the data Y, since the evolution of the data depends only on the two eigenvectors of A, however it fails to extract the next analytical Koopman eigenvalues, this difference makes a big impact in the reconstruction of chaotic dynamical systems, as we will see in the next example.

6.3.2 The Lorenz system

We now analyze the Lorenz attractor with 2 dictionaries on a self implemented python class: the polynomial dictionaries and the Radial Basis Functions as thin plate splines. To collect the data on the Lorenz attractor, we esploit its strongly mixing (see [28]) and hence ergodic properties, i.e. we sample the points along a very long trajectory in the matrix $X_0^{99999} = [x_0, x_1, ..., x_{99999}] \in \mathbb{R}^{3\times 100000}$ and apply a N-dimensional dictionary function $\Psi : \mathbb{R}^3 \to \mathbb{R}^{1\times N}$ with $\Psi(x) = [\psi_1(x), \psi_2(x), ..., \psi_N(x)]$ to obtain the observables matrix $\Psi_0^{99999} = [\psi_1(x)^\intercal, \psi_2(x)^\intercal, ..., \psi_N(x)^\intercal]^\intercal$, the matrices $\Psi_0^{99998} = [\Psi(x_0)^\intercal, \Psi(x_1)^\intercal, ..., \Psi(x_{99998})^\intercal]$ and $\Psi_1^{99999} = [\Psi(x_1)^\intercal, \Psi(x_2)^\intercal, ..., \Psi(x_{99999})^\intercal]$ are then fed to the EDMD algorithm.

Polynomial dictionary

We take into account the dictionary containing the direct product of the monomials $P_n(x) = x^n$ in the 3 components x_1, x_2 and x_3 up to a maximal degree d, therefore containing $N = (d+1)^3$ observables

$$\mathcal{D}_N = \{\psi_1, ..., \psi_N\} \tag{6.39}$$

$$= \{P_0(x_1)P_0(x_2)P_0(x_3), P_1(x_1)P_0(x_2)P_0(x_3), ..., P_n(x_1)P_0(x_2)P_0(x_3), \tag{6.40}$$

$$P_0(x_1)P_1(x_2)P_0(x_3), P_1(x_1)P_1(x_3)P_0(x_3), ..., P_n(x_1)P_1(x_2)P_0(x_3), P_0(x_1)P_2(x_2)P_0(x_3), ..., P_n(x_1)P_n(x_2)P_0(x_3), \tag{6.41}$$

$$P_0(x_1)P_0(x_2)P_1(x_3), P_1(x_1)P_0(x_2)P_1(x_3)..., P_n(x_1)P_n(x_2)P_1(x_3), P_0(x_1)P_0(x_2)P_2(x_3), ..., P_n(x_1)P_n(x_2)P_n(x_3)\}. \tag{6.42}$$

The observables corresponding to the three components x_1, x_2 and x_3 are hence $\psi_2 = P_1(x_1)P_0(x_2)P_0(x_3) = x_1$, $\psi_{n+2} = P_0(x_1)P_1(x_2)P_0(x_3) = x_2$ and $\psi_{(n+1)^2+1} = P_0(x_1)P_0(x_2)P_1(x_3) = x_3$, therefore we obtain the matrix $B = [e_2, e_{n+2}, e_{(n+1)^2+1}] \in \mathbb{R}^{N\times 3}$, where $e_i \in \mathbb{R}^N$ is the ith unit vector.

Table 6.1 Prediction error E vs maximal degree d

d	1	2	3	4	5
E	0.1068611	0.05198063	0.4341987	4690.7162587	770.6846373

First, we analyze the contribution of higher order components by measuring the normalized prediction error $E = \frac{\|X_1^{99999} - \Psi_0^{99998} KB\|_F}{\|X_1^{99999}\|_F}$. From Table 6.1 we recognize that the error significantly decreases from 0.1 to 0.05 introducing second order terms in the dictionary, this is consistent with the results of Champion et al. in [9], since the second order terms x_1x_3 and x_1x_2 are directly involved in the ODE from Equation **??**. It is however also evident that, for a higher maximal degree d, the error rapidly increases until it becomes astronomically large, this phenomenon has been studied by Ghanem and Spanos in [16] and is due to the sensitivity of high order terms to small changes (e.g. noise), which leads to numerical instability. Another important reason for this phenomenon is spectral pollution: higher order terms tend to individuate dynamics of the system which are already captured by lower order terms, leading to the presence of EDMD eigenvalues which are not contained in the spectrum of the Koopman operator corresponding to the Lorenz system.

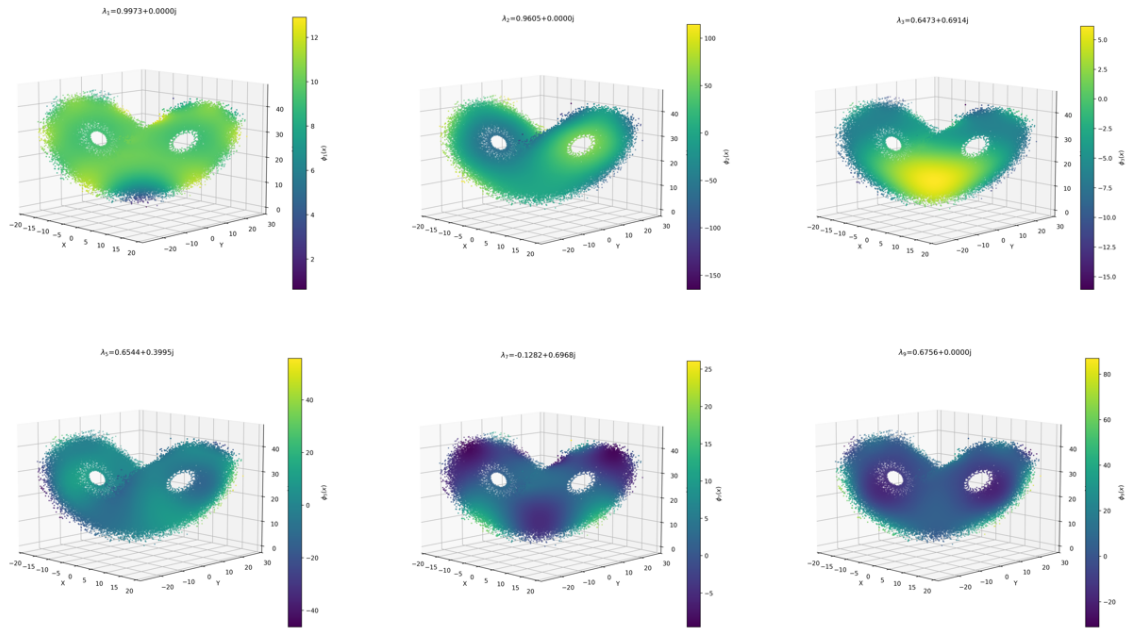

Figure 6.2 Eigenfunctions found by the EDMD algorithm with second order polynomials

Figure 6.2 illustrates the first 6 eigenfunctions identified by EDMD with the second-order dictionary on the Lorenz attractor, out of them the second one seems to be the most interesting, as it identifies the two lobes of the attractor and their corresponding unstable manifolds (See also Prof. Swift's visualization on his page[1]). We finally visualize the EDMD eigenvalues found using the second-order polynomial dictionary in Figure 6.3.

Thin plate splines

We now use the dictionary containing thin plate splines: due to their local nature, these are particularly suited to adapt to Lorenz attractor's complex manifold's geometry. As already mentioned in Subsection 6.1.4, thin-plate splines excel due to their lack of scaling parameters, however we still have to chose the

[1] url: https://ac.nau.edu/ jws8/classes/667.2023.1/lorenz.html

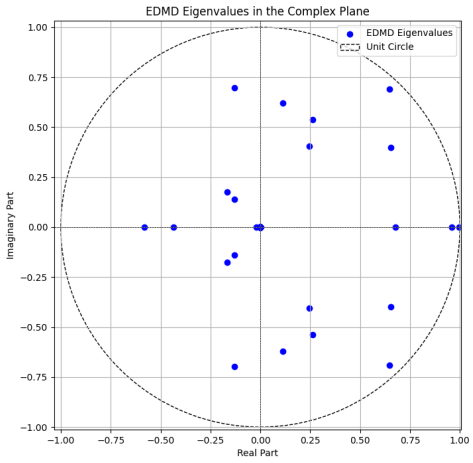

Figure 6.3 Eigenvalues extracted by the EDMD algorithm with second order polynomials

centers of the observables. In order to choose the centers we apply k-means clustering over all 3 dimensional points contained in X_0^{99999}.

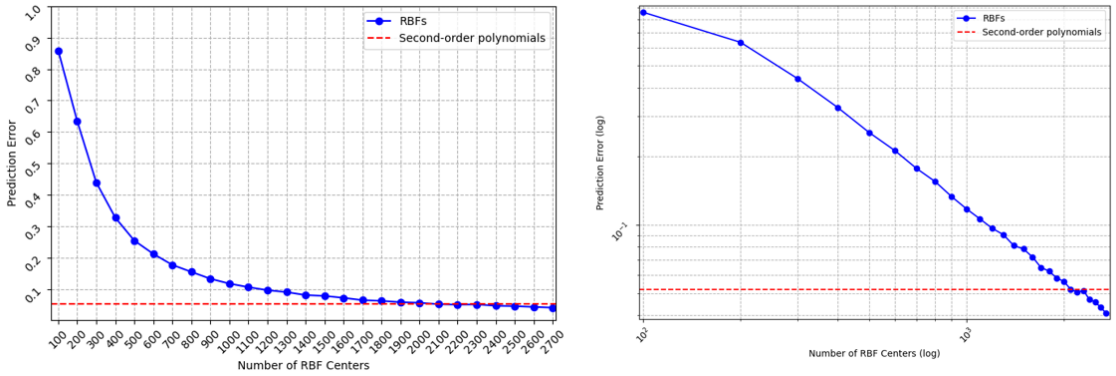

Figure 6.4 Prediction error for a raising amount of RBFs observables

In Figure 6.4 (left), we visualize the normalized prediction error $E = \frac{\|X_1^{99999} - \Psi_0^{99998} KB\|_F}{\|X_1^{99999}\|_F}$ decreasing as the number of thin plate splines N increases, as we can see, this dictionary needs a significantly higher number of observables (around 2200) than second order polynomials (27 observables) to achieve the same degree of accuracy, however increasing further the number of observables allows us to obtain even better results than polynomials. Figure 6.4 (right) shows the same plot in a double logarithmic scale, its near-linear trend

$$\log(E) = a + b\log(N), \tag{6.43}$$

for two parameters $a > 0$ and $b < 0$, suggests a power law dependence between E and N, since

$$E = 10^{a+b\log(N)} = 10^a \cdot 10^{b\log(N)} = \tilde{a}N^b \tag{6.44}$$

holds with $\tilde{a} = 10^a$.

In Figure 6.5 the eigenvalues of the matrix $\mathbb{K}$ with 2500 are scattered on the complex plane, the local nature of the thin plate splines leads them to be all stable. Finally in Figure 6.6 we visualize six eigenfunctions identified by the matrix $\mathbb{K}$ on the Lorenz attractor: the first one associated with $\lambda_1 \approx 1$ seems to capture key areas of density, the points near the stationary points $P_{1,2} = (\pm\sqrt{\frac{b(\rho-1)}{\sigma}}, \pm\sqrt{\frac{b(\rho-1)}{\sigma}}, \rho - 1)$ points are more dense and show a more negative eigenfunction value, moreover points next to the divergence point

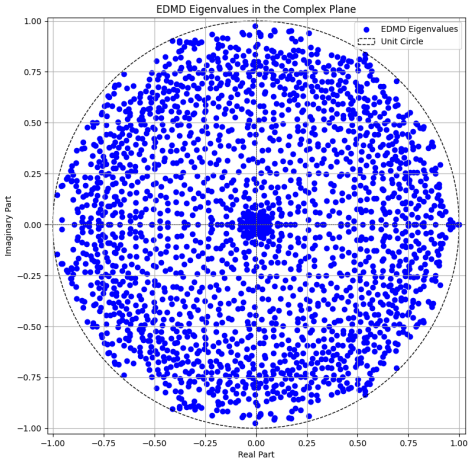

Figure 6.5 Eigenvalues extracted by the EDMD algorithm with 2500 thin plate splines

$P_3 = (0, 0, 0)$ are more sparse and show eigenfunction values very close to P_3, the difference of density of the 2 lobes is due to the particular trajectory chosen for the analysis, which seems to have spent more time in the left lobe than in the right one. The spiral patterns of the next 3 eigenfunctions show the effect of the points P_1 and P_2 "trapping" the particles in their vicinity and making them orbit repeatedly further from these points. The last two eigenfunctions show finer dynamics of the Lorenz attractor: the spiral structures around P_1 and P_2 are still recognizable, moreover the lines spanning between the two lobes seem to map the transition region, where the values show more chaotic jumps, in this region the particles seem to have different possible paths.

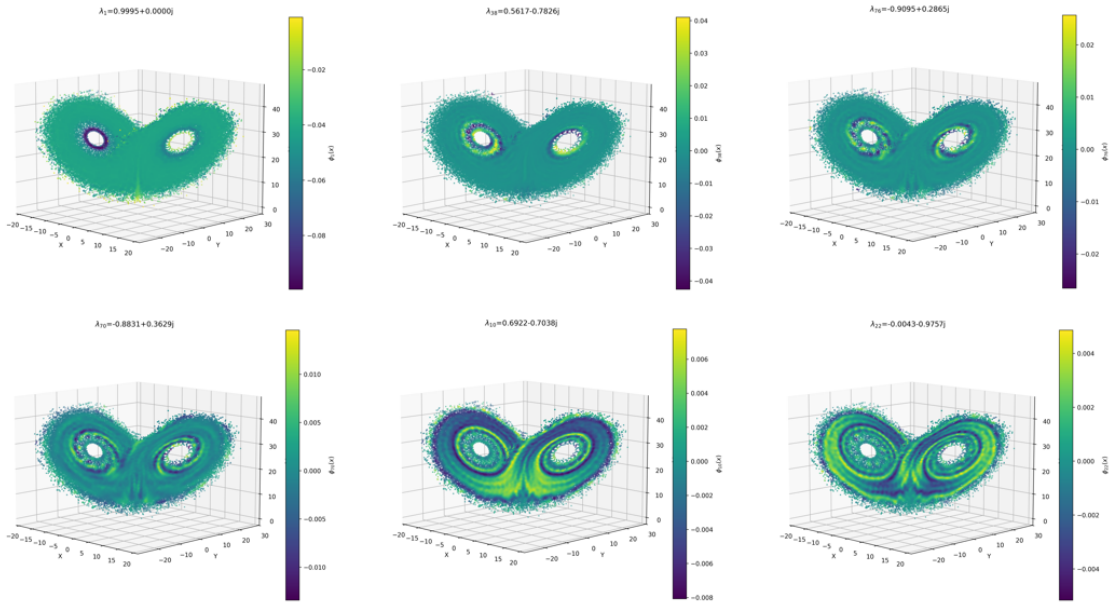

Figure 6.6 Six eigenfunctions found by the EDMD algorithm with 2500 RBFs observables

7 Measure-preserving EDMD (mpEDMD)

In this chapter we will showcase the measure-preserving EDMD algorithm, developed by M. J. Colbrook in [10]. This variant is particularly suited for measure-preserving dynamical systems, whose Koopman operator, as we will see later, is an isometry, allowing us to set some interesting constraints for the determination of its finite dimensional approximation. We will furthermore provide approach (implemented by M.J. Colbrook in his code[1]) based on a QR factorization, which results in a more computational friendly approach.

7.1 Derivation of the algorithm

7.1.1 Standard derivation

We now assume that the discrete dynamical system (2.1) is measure-preserving with respect to some measure ω on $\mathcal{M}$, i.e. that $\omega(f^{-1}(A)) = \omega(A)$ holds for all measurable sets $A \subset \mathcal{M}$.

Theorem 14 *Let a dynamical system $f : \mathcal{M} \to \mathcal{M}$ be measure-preserving with respect to a measure ω on $\mathcal{M}$, then its Koopman operator is an isometry on $L^2(\mathcal{M}, \omega)$ with inner product $\langle \cdot, \cdot \rangle_{L^2(\mathcal{M},\omega)}$ and norm $\|\cdot\|_{L^2(\mathcal{M},\omega)}$.*

Proof: Let $g \in L^2(\mathcal{M}, \omega)$, applying the Koopman operator we have the following L^2 norm

$$\|\mathcal{K}g\|^2_{L^2(\mathcal{M},\omega)} = \int_{\mathcal{M}} |g(f(x))|^2 d\omega(x), \tag{7.1}$$

performing a change of variables $u = f(x)$,

$$\int_{\mathcal{M}} |g(f(x))|^2 d\omega(x) = \int_{\mathcal{M}} |g(u)|^2 \left|\frac{d\omega(x)}{d\omega(u)}\right| d\omega(u), \tag{7.2}$$

where $\left|\frac{d\omega(x)}{d\omega(u)}\right|$ is the Radon-Nikodym derivative, which scales the measure after the transformation. However, because this transformation is measure-preserving, the measure is not affected by the transformation and hence the Radon-Nikodym derivative is equal to 1 almost everywhere, it results

$$\|\mathcal{K}g\|^2_{L^2(\mathcal{M},\omega)} = \int_{\mathcal{M}} |g(f(x))|^2 d\omega(x) = \int_{\mathcal{M}} |g(u)|^2 d\omega(u) = \|g\|^2_{L^2(\mathcal{M},\omega)} \tag{7.3}$$

hence proving that $\mathcal{K}$ is an isometry. □

We now seek a matrix $\mathbb{K} \in \mathbb{C}^{N\times N}$ that approximates the Koopman operator $\mathcal{K}$ on $\mathcal{F}_{\mathcal{D}_N}$ and corresponds to a unitary operator on $\mathcal{F}_{\mathcal{D}_N}$. Let the Gram matrix $G = \Psi_X^* W \Psi_X$ be given, such that W is the diagonal matrix of quadrature weights corresponding to the quadrature nodes $\{x_i\}_{i=1}^m$ used to discretize the problem (see subsection 6.1.2, using other quadrature rules), and $\psi_a, \psi_b \in \mathcal{F}_{\mathcal{D}_N}$ as in (6.3), then we can approximate the inner products in the following manner:

$$\begin{aligned}
\langle \psi_a, \psi_b \rangle_{L^2(\mathcal{M},\omega)} &= \int_{\mathcal{M}} \Psi(x) a \overline{\Psi(x) b} d\omega(x) && (7.4)\\
&= \int_{\mathcal{M}} b^* \Psi(x)^* \Psi(x) a d\omega(x) && (7.5)\\
&\approx b^* \left(\sum_{k=1}^m w_k \Psi(x_k)^* \Psi(x_k) \right) a && (7.6)\\
&= b^* (\Psi_X^* W \Psi_X) a = b^* G a && (7.7)
\end{aligned}$$

[1] url: https://github.com/MColbrook/Measure-preserving-Extended-Dynamic-Mode-Decomposition

where the approximation follows from the quadrature rule. For the norm it similarly holds $\|\psi_a\|^2_{L^2(\mathcal{M},\omega)} = \langle\psi_a, \psi_a\rangle_{L^2(\mathcal{M},\omega)} \approx a^*Ga = \|G^{1/2}a\|_2$ and $\|\Psi\mathbb{K}a\|_2^2 = a^*\mathbb{K}^*G\mathbb{K}a$. As we have already proven, $\mathcal{K}$ is an isometry and hence always satisfies $\|\mathcal{K}g\|_{L^2(\mathcal{M},\omega)} = \|g\|_{L^2(\mathcal{M},\omega)}$ for all $g \in L^2(\mathcal{M},\omega)$, the matrix $\mathbb{K}$ we are aiming to obtain should hence satisfy

$$\|\Psi\mathbb{K}g\|_2^2 = g^*\mathbb{K}^*G\mathbb{K}g = g^*Gg = \|\Psi g\|_2^2 \tag{7.8}$$

for all vectors $g \in \mathbb{R}^N$, but for this to hold, $\mathbb{K}^*G\mathbb{K} = G$ has to hold too.

As we have already mentioned, the original EDMD optimization problem (6.6) is the discretization of the continuous problem

$$\min_{K\in C^{N\times N}} \left\{ \int_{\mathcal{M}} \|\Psi(F(x)) - \Psi(x)B\|_2^2 \mathrm{d}x \right\}, \tag{7.9}$$

via quadrature rule with weights $w_k = \frac{1}{m}$ and nodes x_k for all $k \in \{1, 2, ...m\}$. For the measure-preserving approach, it is hence natural to substitute problem (7.9) with

$$\min_{\substack{K\in C^{N\times N} \\ K^*GK=G}} \left\{ \int_{\mathcal{M}} \|\Psi(F(x))G^{-\frac{1}{2}} - \Psi(x)BG^{-\frac{1}{2}}\|_2^2 \mathrm{d}x \right\}, \tag{7.10}$$

where we introduce the matrix $G^{-\frac{1}{2}}$ on the right using the fact that $\|gG^{\frac{1}{2}}\|_2^2 = gG^{-\frac{1}{2}}GG^{-\frac{1}{2}}g^* = gg^* = \|g\|_2^2$ for all $g \in \mathbb{C}^{1\times N}$. Using a quadrature rule with the weights $w_k \in \mathbb{R}$ contained in the diagonal matrix W, the problem can be discretized as

$$\min_{\substack{K\in C^{N\times N} \\ K^*GK=G}} \sum_{k=1}^{m} w_k \|\Psi(y_k)G^{-\frac{1}{2}} - \Psi(x_k)KG^{-\frac{1}{2}}\|_2^2 = \min_{\substack{K\in C^{N\times N} \\ K^*GK=G}} \|W^{\frac{1}{2}}\Psi_Y G^{-\frac{1}{2}} - W^{\frac{1}{2}}\Psi_X KG^{-\frac{1}{2}}\|_F^2. \tag{7.11}$$

Let $K = G^{-\frac{1}{2}}CG^{\frac{1}{2}}$ for some matrix C, then the measure-preserving constraint

$$K^*GK = G^{\frac{1}{2}}C^*G^{-\frac{1}{2}}GG^{-\frac{1}{2}}CG^{\frac{1}{2}} = G^{\frac{1}{2}}C^*CG^{\frac{1}{2}} = G \tag{7.12}$$

holds if and only if $C^*C = I$. Problem (7.11) is hence equivalent to

$$\min_{\substack{C\in C^{N\times N} \\ C^*C=I}} \|W^{\frac{1}{2}}\Psi_X G^{-\frac{1}{2}}C - W^{\frac{1}{2}}\Psi_Y G^{-\frac{1}{2}}\|_F^2, \tag{7.13}$$

which is an orthogonal Procrustes problem. To solve this problem, it is sufficient to take the SVD of

$$G^{-\frac{1}{2}}\Psi_Y^* W\Psi_X G^{1/2} = U_1\Sigma U_2^*, \tag{7.14}$$

the solution is then given by the matrix $C = U_2U_1^*$ and the measure-preserving Koopman approximator $\mathbb{K} = G^{-\frac{1}{2}}U_2U_1^*G^{\frac{1}{2}}$, if the matrix Σ is degenerate, then $\mathbb{K}$ is not the unique solution of the problem, this however seems to have no impact on the quality of the solution. For stability reasons, Colbrook recommends to decompose the similarity transform C, if $\hat{V}$ is the matrix containing the eigenvectors of C, the eigenvectors of $\mathbb{K}$ are then given by the columns of $V = G^{-\frac{1}{2}}\hat{V}$.

7.1.2 Alternative derivation with QR factorization

There is an alternative method to solve problem (7.11), taking the QR factorization $W^{\frac{1}{2}}\Psi_X = QR$, we see that

$$\begin{aligned}
\arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|W^{\frac{1}{2}}\Psi_Y G^{-\frac{1}{2}} - W^{\frac{1}{2}}\Psi_X K G^{-\frac{1}{2}}\|_F^2 &= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|W^{\frac{1}{2}}\Psi_Y - W^{\frac{1}{2}}\Psi_X K\|_F^2 \\
&= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|W^{\frac{1}{2}}\Psi_Y - QRK\|_F^2 \\
&= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|Q^*W^{\frac{1}{2}}\Psi_Y - RK\|_F^2 \\
&= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|\Psi_Y^*W^{\frac{1}{2}}Q - K^*R^*\|_F^2 \\
&= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|\Psi_Y^*W^{\frac{1}{2}}Q - K^*R^*\|_F^2 \\
&= \arg\min_{\substack{K\in C^{N\times N}\\ K^*GK=G}} \|(R^*)^{-1}(\Psi_Y^*W^{\frac{1}{2}}Q - K^*R^*)\|_F^2. \qquad (7.15)
\end{aligned}$$

Substituting $T = (R^*)^{-1}K^*R^*$ and recalling that $G = \Psi_X^*W\Psi_X = (W^{\frac{1}{2}}\Psi_X)^*(W^{\frac{1}{2}}\Psi_X) = (QR)^*(QR) = R^*Q^*QR = R^*R$ we notice that the constraint

$$K^*GK = \left(R^*T(R^*)^{-1}\right)(R^*R)\left(R^{-1}T^*R\right) = R^*T^*TR = R^*R = G$$

holds if and only if $T^*T = I$. Problem (7.15) is hence equivalent to

$$\arg\min_{\substack{T\in C^{N\times N}\\ T^*T=I}} \|(R^*)^{-1}\Psi_Y^*W^{\frac{1}{2}}Q - T\|_F^2, \qquad (7.16)$$

which is again an orthogonal Procrustes problem: taking the SVD of $(R^*)^{-1}\Psi_Y^*W^{\frac{1}{2}}Q = U_1\Sigma U_2^*$, we can compute the solution as $T = U_2U_1^*$, resulting in $\mathbb{K} = R^{-1}T^*R$. This approach requires less computational cost compared to the other one and is the one we are going to use in the examples.

Advantages of the QR-approach

This approach has several advantages compared to the standard version of mpEDMD: first, it doesn't require to compute the Gram matrix $G = \Psi_X^*W\Psi_X$ and the matrix $A = \Psi_X^*W\Psi_Y$, being faster for large dictionaries; moreover, the QR factorization allows an easier computation of the inverse because of the orthogonality of Q and of the triangular form of R, which makes this approach more stable.

7.2 mpEDMD algorithms

7.2.1 Standard mpEDMD algorithm

We now provide the stardard mpEDMD algorithm starting with the two observable matrices Ψ_X, Ψ_Y and the diagonal matrix of quadrature weights W:

Step 1: Assemble the matrices

$$G = \Psi_X^*W\Psi_X, \qquad (7.17)$$

$$A = \Psi_X^*W\Psi_Y. \qquad (7.18)$$

Step 2: Compute the SVD

$$G^{-\frac{1}{2}}\Psi_Y^*W\Psi_X G^{-\frac{1}{2}} = G^{-\frac{1}{2}}A^*G^{-\frac{1}{2}} = U_1\Sigma U_2^*.$$

Step 3: Assemble the matrix $C = U_2 U_1^*$ and compute its eigendecomposition

$$C = \hat{V}\Lambda\hat{V}^{-1}. \tag{7.19}$$

Step 4: The eigenvectors of the matrix $\mathbb{K} = G^{-\frac{1}{2}} C G^{\frac{1}{2}}$ are then given by the columns of the matrix $V = G^{-\frac{1}{2}}\hat{V}$.

7.2.2 mpEDMD algorithm with QR factorization

We now provide the mpEDMD algorithm with QR factorization, starting as well with the two observable matrices Ψ_X, Ψ_Y and the diagonal matrix of quadrature weights W:

Step 1: Compute the QR factorization

$$W^{\frac{1}{2}}\Psi_X = QR. \tag{7.20}$$

Step 2: Compute the SVD

$$(R^*)^{-1}\Psi_Y^* W^{\frac{1}{2}} Q = U_1 \Sigma U_2^*.$$

Step 3: Assemble the matrix $T = U_2 U_1^*$ and compute its eigendecomposition

$$T = \hat{V}\Lambda\hat{V}^{-1}. \tag{7.21}$$

Step 4: The eigenvectors of the matrix $\mathbb{K} = R^{-1} T^* R$ are then given by the columns of the matrix $V = R^{-1}\hat{V}$.

7.3 Properties of mpEDMD

The measure-preserving EDMD algorithm has very useful properties, which were listed and showed by Colbrook in the following theorem, in part (iii) the notion of pseudospectrum of an operator $\mathcal{A}$

$$\sigma_\epsilon(\mathcal{A}) = \{\lambda \in \mathbb{C} | \|(\mathcal{A} - \lambda)^{-1}\| \geq 1/\epsilon\} = \bigcup_{\|\mathcal{B}\| \leq \epsilon} \sigma(\mathcal{A} + \mathcal{B}), \epsilon > 0. \tag{7.22}$$

as well as the notion of condition number of a matrix A

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} = \|A^{-1}\|_2 \|A\|_2 \tag{7.23}$$

are needed. Moreover, in order to make clear the dependence of the matrices on the number of snapshots m, we will recall the definitions of the matrices G and A and their relation with $\mathcal{K}$,

$$G_m = \sum_{k=1}^{m} \Psi(x_k)^* \Psi(x_k), \tag{7.24}$$

$$A_m = \sum_{k=1}^{m} \Psi(x_k)^* \Psi(y_k), \tag{7.25}$$

$$\mathbb{K}_m = G_m^{-\frac{1}{2}} C_m G_m^{\frac{1}{2}}, \tag{7.26}$$

where the subfix m is added to all matrices related with G_m and A_m.

Theorem 15 *The matrix $\mathbb{K}_m$ obtained via the mpEDMD algorithm satisfies the following properties:*

(i) If (6.30) holds, then any limit point $B = \lim_{m\to\infty} \mathbb{K}_m$ corresponds to an operator that is the unitary part of the polar decomposition $\mathbb{P}_{\mathcal{D}_N}\mathcal{K}\mathbb{P}^*_{\mathcal{D}_N}$, where $\mathbb{P}_{\mathcal{D}_N}$ is the orthogonal projection from $L^2(\mathcal{M}, \omega)$ on $\mathcal{F}_{\mathcal{D}_N}$.

(ii) If (6.30) holds and $\psi_g = \Psi g \in \mathcal{F}_{\mathcal{D}_N}$ is given such that $\mathcal{K}\psi_g \in \mathcal{F}_{\mathcal{D}_N}$, then $\lim_{m\to\infty} \mathbb{K}_m g$ exists and $\lim_{m\to\infty} \Psi \mathbb{K}_m g = \mathcal{K} g$.

(iii) For any $\epsilon \geq 0$, $\sigma_\epsilon(\mathbb{K}_m) \subset \sigma_{\epsilon\kappa(G^{\frac{1}{2}})}(U_2 U_1^*) \subset \{z \mid \ ||z| - 1| \leq \epsilon\kappa(G_m^{\frac{1}{2}})\}$

(iv) $\kappa(V) \leq \kappa(G_m^{\frac{1}{2}})$

Proof:

"(i)" Let $B = \lim_{m\to\infty} \mathbb{K}_m \in \mathbb{C}^{N\times N}$, by taking subsequences if necessary, we may assume that $B = \hat{G}^{\frac{1}{2}}\hat{U}_2\hat{U}_1^*\hat{G}^{\frac{1}{2}}$, where $\hat{U}_j = lim_{m\to\infty}U_{j_m}$, $\hat{G} = lim_{m\to\infty}G_m$, and $\hat{\Sigma} = lim_{m\to\infty}\Sigma_m$. In the large data limit, the problem (7.11) is independent of the choice of basis for $\mathcal{D}_N$ and property (i) is also basis independent. Hence, we may assume without loss of generality that $\hat{G}$ is the identity matrix corresponding to an orthonormal basis. It follows that $\mathbb{K}_{EDMD} = \hat{G}^{\dagger}\hat{A} = \hat{A} = \hat{U}_2\hat{\Sigma}\hat{U}_1^*$, since $\hat{G}^{-\frac{1}{2}}\hat{A}^*\hat{G}^{1/2} = A^* = \hat{U}_1\hat{\Sigma}\hat{U}_2^*$ holds. The matrix $\mathbb{K}_m = U_{2_m}U_{1_m}^*$ is hence the unitary part of $\mathbb{K}_{EDMD} = \hat{U}_2\hat{\Sigma}\hat{U}_1^*$ which approaches $\mathbb{P}_{\mathcal{D}_N}\mathcal{K}\mathbb{P}_{\mathcal{D}_N}^*$ as $m \to \infty$, because (6.30) holds.

"(ii)" Since $\mathcal{K}\psi_g \in \mathcal{F}_{\mathcal{D}_N}$ and $\mathcal{K}$ is an isometry, it holds

$$\|\psi_g\| = \|\mathcal{K}\psi_g\| = \lim_{m\to\infty}\|U_{2_m}\Sigma U_{1_m}^* G_m^{\frac{1}{2}}g\| = \lim_{m\to\infty}\|\Sigma U_{1_m}^* G_m^{\frac{1}{2}}g\|, \tag{7.27}$$

where the last equality follows from the fact that U_{2_m} is unitary. Moreover $\|g\| = \lim_{m\to\infty}\|G_m^{\frac{1}{2}}g\| = \lim_{m\to\infty}\|U_{1_m}G_m^{\frac{1}{2}}g\|$ holds, with the last equality justified for the same reason as before. It follows that $\lim_{m\to\infty}\|\Sigma U_{1_m}^* G_m^{\frac{1}{2}}g\| = \lim_{m\to\infty}\|U_{1_m}G_m^{\frac{1}{2}}g\|$ and hence that the eigenvalues contained in the matrix Σ_m are all contained in the interval $[0, 1]$.

We claim that $lim_{m\to\infty}[U_{1_m}^* G_m^{\frac{1}{2}} - \Sigma_m U_{1_m}^* G_m^{\frac{1}{2}}]g = 0$. If not, then by taking a subsequence if necessary, we may assume that $\lim_{m\to\infty}\Sigma_m$ and $\lim_{m\to\infty}U_{j_m}$ exist with $\lim_{m\to\infty}[U_{1_m}^+ G_m^{\frac{1}{2}}\Sigma U_{1_m}^* G_m^{\frac{1}{2}}]g \neq 0$, but this contradicts $\lim_{m\to\infty}\|\Sigma_m U_{1_m}^* G_m^{\frac{1}{2}}g\| = \lim_{m\to\infty}\|U_{1_m}G_m^{\frac{1}{2}}g\|$. Since $\lim_{m\to\infty} G_m^{-\frac{1}{2}}U_{2_m}\Sigma U_{1_m}^* G_m^{\frac{1}{2}}g = \lim_{m\to\infty}\mathbb{K}_{EDMD}g$ exists, $\lim_{m\to\infty}\mathbb{K}_m g$ exists and $\lim_{m\to\infty}\Psi\mathbb{K}_m g = \lim_{m\to\infty}\Psi\mathbb{K}_{EDMD}g = \mathcal{K}\Psi_g$.

"(iii)" For all $z \notin \sigma_\epsilon(\mathbb{K}_m)$ we have $\|(\mathbb{K}_m - z)^{-1}\| = \|G_m^{-\frac{1}{2}}(U_{2_m}U_{1_m}^* - z)^{-1}G_m^{\frac{1}{2}}\| \leq \|G_m^{-\frac{1}{2}}\|\|G_m^{\frac{1}{2}}\|\|(U_{2_m}U_{1_m}^* - z)^{-1}\| = \kappa(G_m^{\frac{1}{2}})\|(U_{2_m}U_{1_m}^* - z)^{-1}\|$, therefore $\sigma_\epsilon(\mathbb{K}_m) \subset \sigma_{\epsilon\kappa(G_m^{\frac{1}{2}})}(U_{2_m}U_{1_m}^*)$. Since $U_{2_m}U_{1_m}^*$ is a unitary matrix, its eigenvalues are inside of the unit circle $\mathbb{T}$, therefore $\sigma_{\epsilon\kappa(G_m^{\frac{1}{2}})}(U_{2_m}U_{1_m}^*) \subset \{z| \ ||z| - 1| \leq \epsilon\kappa(G_m^{\frac{1}{2}})\}$.

"(iv)" We recall that $V = G_m^{\frac{1}{2}}\hat{V}$, where $\hat{V}$ is the matrix of eigenvectors of the unitary matrix $U_2U_1^*$, therefore $\hat{V}$ is unitary too and $\kappa(V) \leq \kappa(G_m^{\frac{1}{2}})$ holds.

7.3.1 Convergence theory

In [10], M. J. Colbrook has shown convergence of the matrix $\mathbb{K}$ to the spectral information of the Koopman operator $\mathcal{K}$. These are very important results, which show that the mpEDMD algorithm is suited for the study of continuous spectra, a very difficult task in the study of infinite dimensional operators. In this subsection we are going to showcase the results of Colbrook's studies, however the proofs will not be provided in this thesis and can be found in Appendix A of [10].

Approximation of projection-valued spectral measures

In Chapter I of [4], Nagy at al. show that the Koopman operator $\mathcal{K} : L^2(\mathcal{M}, \omega) \to L^2(\mathcal{M}, \omega)$ of a measure-preserving dynamical system has a unitary extension $\mathcal{K}'$ defined on an extended Hilbert space $\mathcal{H}'$ with $L^2(\mathcal{M}, \omega) \subset \mathcal{H}'$. Since $\mathcal{K}'$ is unitary, its spectrum is contained inside the unit circle $\mathbb{T}$ and, by the spectral theorem for normal operators, it is possible to decompose the Hilbert space $\mathcal{H}'$ and diagonalize $\mathcal{K}'$ via a spectral measure $\mathcal{E}'$ assigning an orthogonal projector to each Borel measurable subset of $\mathbb{T}$

$$f = \left(\int_{\mathbb{T}} d\mathcal{E}'(\lambda)\right)f, \ \ \mathcal{K}f = \left(\int_{\mathbb{T}} \lambda d\mathcal{E}'(\lambda)\right)f, \ \ \forall f \in \mathcal{H}'. \tag{7.28}$$

In other words, the spectral measure $\mathcal{E}'$ evaluated at a subset of the spectrum $A \subset \mathbb{T}$ captures the contribution of the spectra contained in A to the whole operator $\mathcal{K}$. Let $\mathbb{P}$ denote the orthogonal projection from $\mathcal{H}'$ to $L^2(\mathcal{M}, \omega)$, although the unitary extension $\mathcal{K}'$ is not unique, Colbrook has shown that the spectral measure $\mathcal{E} = \mathbb{P}\mathcal{E}'\mathbb{P}^*$ of $\mathcal{K}$ is independent of the choice of unitary extension, we are hence able to access the spectral information of $\mathcal{K}$ by considering $\mathcal{K}'$. To approximate the spectral measure $\mathcal{E}$, we consider the spectral measure, $\mathcal{E}^{(N,m)}$, of the matrix $\mathbb{K}$. Since the matrix $\mathbb{K}$ is unitary, the spectral theorem state that there exists an G-orthonormal (i.e. $v_i^* G v_j = \delta_{ij}$) basis of vectors $\{v_1, ..., v_N\}$ on the Hilbert space C^N, with the inner product in (7.7) induced by G, such that

$$v = (\sum_{j=1}^{N} v_j v_j^* G)v, \quad \mathbb{K}v = (\sum_{j=1}^{N} \lambda_j v_j v_j^* G)v, \quad \forall v \in \mathbb{C}^N \tag{7.29}$$

To approximate the spectral measure $\mathcal{E}$, we consider the spectral measure, $\mathcal{E}^{(N,m)}$, of the matrix $\mathbb{K}$, this is obtained from (7.29)

$$d\mathcal{E}^{(N,m)}(\lambda) = \sum_{j=1}^{N} v_j v_j^* G \delta(\lambda - \lambda_i) d\lambda, \tag{7.30}$$

where $\delta(\lambda - \lambda_j)$ is the Dirac delta function centered at the eigenvalue λ_j. This provides an approximation to the spectral properties of the Koopman operator through the eigenvalues of $\mathbb{K}$, the following theorem shows weak convergence of $\mathcal{E}^{(N,m)}$ if the operator $\mathcal{K}$

Theorem 16 *Suppose that* $\lim_{N\to\infty} dist(h, \mathcal{D}_N) = 0$ *for all* $h \in L^2(\mathcal{M}, \omega)$*, (6.30) holds,* $\mathcal{K}$ *is unitary, and that* $\phi : \mathbb{T} \to \mathcal{R}$ *is Lipschitz continuous. Then for any* $g \in L^2(\mathcal{M}, \omega)$ *and* $g_N \in \mathbb{C}^N$ *with* $\lim_{N\to\infty} |g - \Psi g_N| = 0$*,*

$$\lim_{N\to\infty} \limsup_{M\to\infty} \left\| \int_{\mathbb{T}} \psi(\lambda) d\mathcal{E}(\lambda) g - \Psi \int_{\mathbb{T}} \phi(\lambda) g_N \right\| = 0 \tag{7.31}$$

In the proof provided by Colbrook, it is possible to see that the rate of convergence of the integral depends on the Laurent series of the function ϕ and how well the powers of $\mathbb{K}$ approximate the powers of $\mathcal{K}$.

Approximation of scalar-valued spectral measures

We now consider a function $g \in L^2(\mathcal{M}, \omega)$ with $\|g\| = 1$ and its scalar-valued spectral measure $\mu_g(U) = \langle \mathcal{E}(U), g \rangle$, we can again use equation (7.7) to approximate μ_g by $\mu_{g_N}^{(N,m)}$, defined as

$$\mu_{g_N}^{(N,m)}(U) = g_N^* \mathcal{E}^{(N,m)}(U) g_N = \sum_{\lambda_j \in U} |v_j^* G g_N|^2 \tag{7.32}$$

where $g_N \in \mathbb{C}^N$ is also normalized such that $g_N^* G g_N = 1$. Since $\{G^{\frac{1}{2}} v_j\}_{j=1}^N$ is an orthonormal basis for $\mathbb{C}^N$ with the standard inner product, $\mu_{g_N}^{(N,m)}$ is a probability measure. To measure the distance between probability measures, we use the 1-Wasserstein distance, W_1. For two Borel probability measures μ and 4 on $\mathbb{T}$, the W_1 distance is defined as

$$W_1(\mu, \nu) := \sup \left\{ \int_{\mathbb{T}} \phi(\lambda) d(\mu - \nu)(\lambda) | \phi : \mathbb{T} \to \mathbb{R} \text{ Lip. cts., Lip. constant} \leq 1 \right\}. \tag{7.33}$$

Convergence in this metric is equivalent to the usual weak convergence of measures. The following theorem explicitly bounds $W_1(\mu_g, \mu^(N,M)_{g_N})$, before taking any limits.

Theorem 17 *For any* $L \in \mathbb{N}$ *,* $g \in L^2(\mathcal{M}, \omega)$*, and* $g_N \in \mathbb{C}^N$ *with* $g_N^* * G g_N = 1$*,*

$$W_1(\mu_g, \mu_{g_N}^{(N,m)}) \leq C \left(\frac{\log(L)}{L} + \sum_{1 \leq l \leq L} \frac{|\langle \mathcal{K}^{|l|} g, g \rangle - g_N^* G \mathbb{K}^{|l|} g_N|}{l} \right) \tag{7.34}$$

for a universal constant C that can be made explicit.

This theorem, together with Lemma 2 from Appendix A of [10] leads to the following result of weak convergence of the scalar-valued spectral measures.

Theorem 18 *Suppose that* $lim_{N\to\infty} dist(h, \mathcal{D}_N) = 0$ *for all* $h \in L^2(\mathcal{M}, \omega)$*, and (6.30) holds. Then for any* $g \in L^2(\mathcal{M}, \omega)$ *and* $g_N \in \mathbb{C}^N$ *with* $\lim_{N\to\infty} \|g - \Psi g_N\| = 0$*,*

$$\lim_{N\to\infty} \limsup_{m\to\infty} W_1(\mu_g, \mu_{g_N}^{(N,m)}) = 0. \tag{7.35}$$

As we have seen in the previous chapters, a popular choice of dictionary is the Krylov subspace $\text{span}\{g, \mathcal{K}g, ..., \mathcal{K}^L g\}$, which corresponds to delay-embedding. Generally one can consider the case $\{g, \mathcal{K}g, ..., \mathcal{K}^L g\} \subset \mathcal{D}_N$. Part (ii) of Theorem 15 shows that if $g, \mathcal{K}g, ..., \mathcal{K}^l g \in \mathcal{D}_N$ and $g = \Psi g_N$, then $\lim_{m\to\infty} |\langle \mathcal{K}_m^{|L|} g, g\rangle - g_N^* G \mathbb{K}_m^{|L|} g_N| = 0$. Combining this with Theorem 5.3 shows the following theorem, which provides an explicit rate of convergence.

Theorem 19 *If* $\{g, \mathcal{K}g, ..., \mathcal{K}^L g\} \subset \mathcal{D}_N$ *,* $g = \Psi g_N$*, and (6.30) holds, then*

$$\limsup_{M\to\infty} W_1(\mu_g, \mu^(N, M) g_N) \leq C \frac{\log(L)}{L} \tag{7.36}$$

for a universal constant C *that can be made explicit.*

Approximation of spectra

Finally, in the next theorem, Colbrook shows that the spectrum $\sigma(\mathbb{K}) = \{\lambda_1, ..., \lambda_N\}$ converges to the approximate point spectrum of $\mathcal{K}$, defined as

$$\sigma_{ap}(\mathcal{K}) = \left\{\lambda \in \mathbb{C} | \exists \{g_n\}_{n\in\mathbb{N}} \text{ such that } \|g_n\| = 1, \lim_{n\to\infty} \|(\mathcal{K} - \lambda) g_n\| = 0\right\} \tag{7.37}$$

which is a generalization of the spectrum $\sigma(\mathcal{K})$

Theorem 20 *If* $\lim_{N\to\infty} dist(h, \mathcal{D}_N) = 0 \ \forall h \in L^2(\mathcal{M}, \omega)$ *and (6.30) holds, then*

$$\lim_{N\to\infty} \limsup_{m\to\infty} \sup_{\lambda\in\sigma_{ap}(\mathcal{K})} dist(\lambda, \{\lambda_1, ...\lambda_N\}) = 0 \tag{7.38}$$

Despite this result, it is important to notice that the matrix $\mathbb{K}_m$ can suffer from spectral pollution, i.e. eigenvalues may approximate points which are not contained in the approximate point spectrum $\sigma_{ap}(\mathcal{K})$. To counter this effect, it is possible to apply Residual Dynamical Mode Decomposition, developed by J. Colbrook et al. in [11], which is able to evaluate whether an eigenfunction $\phi_i(x) = \Psi(x) v_i$ satisfies $\|\mathcal{K}\phi_i - \lambda_i \phi_i\| \leq \epsilon$, for a user defined $\epsilon > 0$.

7.4 Examples

7.4.1 Characterizing the Lorenz System's Continuous Spectra

Following the work of Colbrook, we analyze the spectral information of the Koopman operator associated with the Lorenz system using a self-implemented python class (see Appendix A.1.2) adapted from Colbrook's MATLAB code[2]. This system is chaotic, strongly mixing (see [28]), and therefore ergodic, allowing us to use ergodic sampling by taking snapshots along a single long trajectory with a time step of $\Delta t = 0.1$. We choose the quadrature weights to be uniformly distributed, set to $\frac{1}{m}$.

We first examine the scalar-valued spectral measures μ_{g_j}, with $\mu_{g_j}(x) = c_j x_j$ being the projection onto the jth component of $x \in \mathbb{R}^3$, normalized with respect to the ergodic measure ω. For each component, we construct a delay-embedding dictionary $g_j, \mathcal{K}g_j, \ldots, \mathcal{K}^{N-1} g_j$.

[2] url: https://github.com/MColbrook/Measure-preserving-Extended-Dynamic-Mode-Decomposition

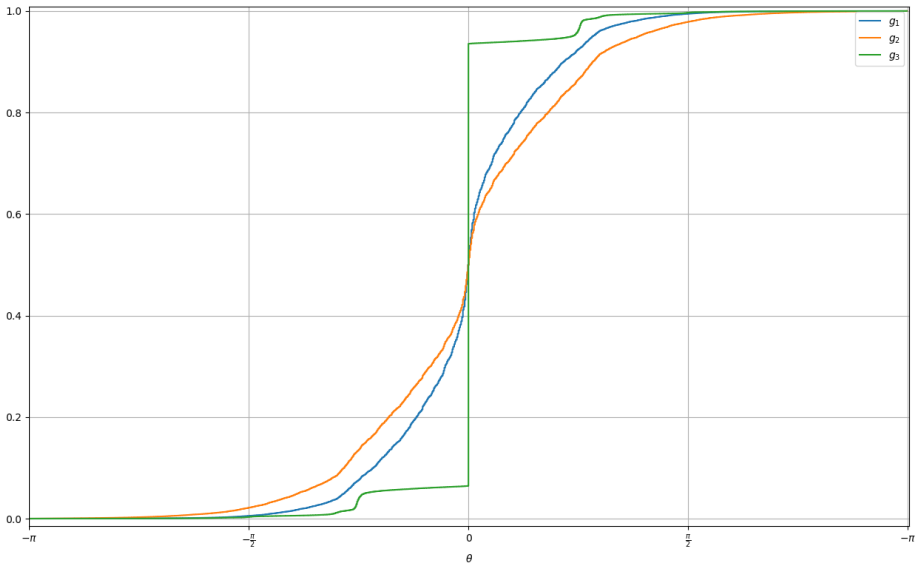

Figure 7.1 CDFs of the scalar-valued spectral measures μ_{g_j}

In Figure 7.1 we plot the cumulative distribution functions (CDFs) of the scalar-valued spectral measures $\mu_{g_j}^{(10^3,10^4)}$. Although memory constraints limited our snapshots to two orders of magnitude fewer than those used by Colbrook, the CDFs exhibit the same qualitative behavior observed in Figure 1 (right) of [10]. This agreement highlights the robustness of our analysis.

As seen in the plot, the discontinuous jump of the CDF for μ_{g_3} at phase $\theta = 0$ (i.e., $\lambda = 1$) suggests the presence of a discrete eigenvalue, with a continuous spectrum in its vicinity. In contrast, the CDFs of μ_{g_1} and μ_{g_2} appear to follow a more uniform distribution, suggesting a continuous spectrum across the phase space $[-\pi, \pi]$. This discrepancy aligns with the theoretical understanding of the Lorenz system: the components x_1 and x_2 govern the more chaotic dynamics, driving the butterfly-shaped trajectories of the Lorenz attractor. In comparison, x_3 evolves more slowly and exhibits less chaotic behavior, primarily influenced by the product of x_1 and x_2, which averages out over time. The constant mode associated with $\lambda = 1$ is thus more influential for x_3.

The CDFs for μ_{g_1} and μ_{g_2} indicate continuous spectra, but we observe that low-phase eigenvalues seem to have a stronger impact, as evidenced by the sharp slope near $\theta = 0$. This observation points to the particular importance of low-phase eigenvalues in the Lorenz system, aligning with the system's persistent nature. Despite the sensitivity to initial conditions, chaotic systems like the Lorenz attractor exhibit bounded, non-repetitive behavior within a complex but structured shape.

This intuition is further supported by the analysis of the most influential modes of the Lorenz system through low-rank and sparse DMD (see Section 4.3.1, particularly Figures 4.5 and 4.3), where the most significant modes correspond to low-phase eigenvalues. This result reinforces the conclusion that low-phase eigenvalues play a crucial role in capturing the persistent features of the Lorenz attractor.

8 EDMD with dictionary learning (EDMD-DL)

Finally, in this last chapter we showcase the extended dynamic mode decomposition algorithm with dictionary learning provided by Li et al. in [26]: this variant of the EDMD algorithm has the goal of alleviating the problem of the choice of dictionary, which, as we have seen in the previous chapters, plays a fundamental role in the prediction of the state of a system.

8.1 Derivation of the algorithm

8.1.1 Dictionary learning and approach to EDMD applications

Dictionary learning This is a classical problem in machine learning (see [14]): given a set of data $X = [x_1, x_2, ..., x_m] \in \mathbb{R}^{n\times m}$, the objective is to find a sparse representation $X = DK$, where $D \in \mathbb{R}^{n\times N}$ is a size-N set of dictionary vectors and $K \in \mathbb{R}^{N\times m}$ is a sparse representation. For any fixed dictionary D, it is difficult to fulfill accuracy (i.e. $X \approx DK$) **and** sparsity (i.e. $\|K\|_0$ small) at the same time. A better approach is to solve the problem

$$(K, D) = \arg\min_{(\tilde{K},\tilde{D})} J(\tilde{K}, \tilde{D}) = \|X - \tilde{K}\tilde{D}\|_F^2 + \lambda\|\tilde{K}\|_1 \tag{8.1}$$

where the ℓ_1 penalty term induces sparsity of the matrix K without turning the optimization problem into a combinatorial one, in a similar fashion to Sparse DMD's optimization problem (4.4).

Dictionary learning adapted to EDMD applications The key idea of EDMD-DL is to make the dictionary function $\Psi : \mathbb{R}^n \to \mathbb{R}^{1\times N}$ adaptive so that one can minimize the norm of the residual $\Psi \circ f - \Psi K$. So instead of solving the standard EDMD problem (6.5), we can consider the extended minimization problem

$$\min_{(\tilde{K},\tilde{\Psi})} J(\tilde{K}, \tilde{\Psi}) = \min_{(\tilde{K},\tilde{\Psi})} \sum_{k=1}^{m} \|\tilde{\Psi}(y_k) - \tilde{\Psi}(x_k)\tilde{K}\|_2^2 + \lambda(\tilde{K}, \tilde{\Psi}), \tag{8.2}$$

where $\lambda(K, \Psi)$ is a suitable regularizer. Unlike problem (8.1), where the dictionary D is made of vectors, the dictionary functions in Ψ are not assumed to be linear, therefore iterative nonlinear optimization methods are needed; nevertheless, provided one can solve (8.2), the solution (K, Ψ) provides a set of dictionary functions that give the optimal residual term (in other words, an almost Koopman invariant subspace $\mathcal{F}_{\mathcal{D}_N} \subset L^2(\mathcal{M}, \omega)$).

8.1.2 Practical algorithm

In [26], Li et al. parametrize the dictionary function Ψ via a universal function approximator, i.e. $\Psi(x) = \Psi(x; \theta)$ for some parameters $\theta \in \Theta$ that can be varied; for their algorithm, they choose a simple feed-forward, 3-layer neural network with tanh as activation function. Concretely, one chooses an hidden dimension $\ell \in \mathbb{N}$ and let the dictionary function be given by

$$\Psi(x) = W_{out}h_3 + b_3 \tag{8.3}$$

$$h_k = \tanh(W_k h_k + b_k), \quad k = 0, 1, 2, \tag{8.4}$$

where $h_0 = x \in \mathbb{R}^n$, $W_0 \in \mathbb{R}^{\ell\times n}$, $b_0 \in \mathbb{R}^\ell$, $W_{out} \in \mathbb{R}^{N\times\ell}$, $b_{out} \in \mathbb{R}^N$ and $W_k \in \mathbb{R}^{\ell\times\ell}$, $b_k \in \mathbb{R}^\ell$ for $k = 1, 2$. Thus, the set of all trainable parameters $\theta = \{W_{out}, b_{out}, \{W_k, b_k\}_{k=0}^2\}$ contains $\ell(2\ell+n+3)+N(\ell+1)$ scalar variables.

With this parametrization of Ψ, we can explicitly state the minimization problem (8.2) as

$$\min_{(\tilde{K},\tilde{\theta})} J(\tilde{K},\tilde{\theta}) = \min_{(\tilde{K},\tilde{\theta})} \sum_{k=1}^{m} \|\Psi(y_k;\tilde{\theta}) - \Psi(x_k,\tilde{\theta})\tilde{K}\|_2^2 + \lambda\|\tilde{K}\|_F^2, \tag{8.5}$$

where the Tikhonov regularizer (see [30] and [1]) with identity matrix $\|K\|_F^2$ was chosen to improve stability.

It is important to notice that, if there is a $\theta \in \Theta$ such that $\Psi \equiv 0$, then the minimization problem (8.5) is trivially solved, to this end Li et al. suggest to add some non-trainable functions to the dictionary, such as the constant function $\psi_1(x) \equiv 1$ and the n projection maps, which are also particularly helpful to reconstruct the full state by equation (6.13). To solve problem (8.5), Li et al. provide an iterative procedure based on the following 2 steps: (a) Fix θ, optimize K; (b) fix K, optimize θ.

(a) **Fix θ, optimize K.** For a fixed $\tilde{\theta}$ (and hence a fixed dictionary function Ψ^θ), problem (8.5) is similar to (6.5) and analytically solvable: recalling the equality $\|A\|_F^2 = trace(A^*A)$ and $\Psi_X^\theta = [\Psi^\theta(x_1)^\intercal, \dots, \Psi^\theta(x_m)^\intercal]^\intercal$, $\Psi_Y^\theta = [\Psi^\theta(y_1)^\intercal, \dots, \Psi^\theta(y_m)^\intercal]^\intercal$, we reformulate the term J as follows

$$\begin{aligned} J(\tilde{K}) &= \sum_{k=1}^{m} \|\Psi^\theta(y_k) - \Psi^\theta(x_k)\tilde{K}\|_2^2 + \lambda\|\tilde{K}\|_F^2 && (8.6)\\ &= \|\Psi_Y^\theta - \Psi_X^\theta\tilde{K}\|_F^2 + \lambda\|\tilde{K}\|_F^2 && (8.7)\\ &= trace((\Psi_Y^\theta - \Psi_X^\theta\tilde{K})^*(\Psi_Y^\theta - \Psi_X^\theta\tilde{K})) + \lambda trace(\tilde{K}^*\tilde{K}) && (8.8)\\ &= trace(\Psi_Y^{\theta*}\Psi_Y^\theta) - trace(\Psi_Y^{\theta*}\Psi_X^\theta\tilde{K}) - trace(\tilde{K}^*\Psi_X^{\theta*}\Psi_Y^\theta) + trace(\tilde{K}^*(\Psi_X^{\theta*}\Psi_X^\theta + \lambda I)\tilde{K}), && (8.9) \end{aligned}$$

taking its gradient with respect to $\tilde{K}$ and setting it equal to zero, we find the optimal solution K

$$\begin{aligned} &\nabla J(K) = -2\Psi_X^{\theta*}\Psi_Y^\theta + 2(\Psi_X^{\theta*}\Psi_X^\theta + \lambda I)K = 0 && (8.10)\\ &\implies K = (\Psi_X^{\theta*}\Psi_X^\theta + \lambda I)^\dagger \Psi_X^{\theta*}\Psi_Y^\theta = (G^\theta + \lambda I)^\dagger A^\theta, && (8.11) \end{aligned}$$

where G^θ and A^θ are defined as in (6.11) and (6.10) and I is the N-dimensional identity matrix.

(b) **Fix K, optimize θ.** As there is no linear structure in the problem, we cannot write down its analytical solution. Instead, we proceed by iterative updates in the form of gradient descent, i.e., we set

$$\tilde{\theta}_{i+1} = \tilde{\theta}_i + \delta\nabla_\theta J(\tilde{K},\tilde{\theta}_i) \tag{8.12}$$

If the number of dictionary functions N or the dimension n are large, then $\nabla_\theta J$ will be extensive to calculate. In those cases we can employ stochastic gradient descent methods to approximate the gradient $\nabla_\theta J$ with randomly sampled unbiased estimators.

8.2 EDMD-DL algorithm

We now provide the EDMD-DL algorithm for a given set of snapshot pairs $\{(x_k, y_k)\}_{k=1}^m$, a set of initial parameters θ_0 and a scalar $\epsilon > 0$:

Step 1: For a fixed θ_i and hence fixed Ψ_i, compute the matrix

$$K_{i+1} = (\Psi_X^*\Psi_X + \lambda I)^\dagger \Psi_X^*\Psi_Y = (G_i + \lambda I)^\dagger A_i. \tag{8.13}$$

Step 2: For a fixed K_{i+1}, compute the set of parameters

$$\theta_{i+1} = \theta_i + \delta\nabla_\theta J(K_{i+1},\theta_i). \tag{8.14}$$

Step 3: If $J(K_{i+1},\theta_{i+1}) > \epsilon$ return to Step 1, otherwise finish.

8.3 Examples

We will evaluate the methods applied in the following example by the two metrics:

- **Accuracy of the trajectory reconstruction.** We reconstruct trajectories for unseen initial conditions $x_0 \in \mathbb{R}^n$ using equation (6.19). We then monitor the reconstruction error as

$$E = \sqrt{\frac{1}{m+1}\sum_{k=0}^{m} |x_k - \tilde{x}_k|^2} \tag{8.15}$$

  where x is the true trajectory data generated by (2.1) and $\tilde{x}$ is the reconstructed trajectory generated by (6.19).

- **Accuracy of eigenfunction approximation.** To measure the invariance to the Koopman operator of the subspace spanned by the dictionary $\mathcal{D}_N$, we define for each $j \in \{1, \ldots, N\}$ the eigenfunction approximation error

$$E_j = \|\phi_j \circ f - \mu_j \phi_j\|_{L^2(\mathcal{M},\omega)}, \tag{8.16}$$

  where ϕ_j and μ_j are the jth eigenfunction and eigenvalue found by the algorithm, respectively. The above quantity can be approximated by Monte-Carlo integration

$$E_j \approx \sqrt{\frac{1}{I}\sum_{k=1}^{I} |\phi_j \circ f(x_k) - \mu_j \phi(x_k)|^2} \tag{8.17}$$

  where the points $x_k \sim \omega$ are identically and independently distributed.

8.3.1 Lorenz system

We now apply the EDMD-DL algorithm with 23 trainable dictionary functions collected in $\tilde{\Psi}_{23}$ and 4 fixed functions being given by the constant function $\psi_1 \equiv 1$ and the three projection maps $\psi_2(x) = x_1$, $\psi_3(x) = x_2$, $\psi_4(x) = x_3$, such that the dictionary function is given by $\Psi_{27} = [\psi_1, \psi_2, \psi_3, \psi_4, \tilde{\Psi}_{23}]$ with 27 observables, mirroring the structure of a second-order polynomial dictionary for three-dimensional states. For comparison, we use EDMD with a dictionary of 2500 thin-plate radial basis functions (RBFs), as prior analysis has shown that RBFs outperform second-order polynomials when applied to the Lorenz system (see Example 6.3.2). To generate the training data, we draw 1000 initial conditions on the Lorenz system and take the next 10 iterations given by the fourth-order Runge-Kutta integrator of each initial condition, we collect all those points in the matrix $X \in \mathbb{R}^{3\times 10000}$, we apply the Runge-Kutta integrator once again on each of these points to obtain the matrix $Y \in \mathbb{R}^{3\times 10000}$.

Reconstruction error comparison

Table 8.1 Reconstruction error of the 3 runs

run \ method	EDMD-DL	EDMD (RBFs)
1	0.079913752	0.375551104
2	0.089276859	0.037775504
3	0.091907794	0.218830334

We now generate 3 sets of training data matrices via the procedure described above and feed them to the two methods in 3 different runs; for each run, we draw a random initial condition on the Lorenz attractor and reconstruct its next 50 iterations via equation (6.19), the normalized reconstruction errors in Table

8.1 show that EDMD-DL has a much lower reconstruction error than EDMD with thin-plate RBFs in the first and third run, being still competitive in the second one. For each run, both methods use dictionaries constructed based on the same training data. In the case of RBFs, their centers are determined via k-means clustering on the training data, ensuring that their placement is adapted to the specific dataset. Similarly, EDMD-DL retrains its trainable dictionary functions for each run to adapt to the data. Despite this shared dependence on training data, EDMD-DL exhibits a greater ability to consistently reconstruct the Lorenz attractor's dynamics across runs. This suggests that its learned dictionary is more efficient at capturing the essential features of the system, even with far fewer observables (27 for EDMD-DL compared to 2500 RBFs). The reduced error variability of EDMD-DL further highlights its stability, which may be attributed to the flexibility of its trainable functions in identifying structures in the data that align closely with the system's true dynamics. In contrast, while the RBF dictionary benefits from high dimensionality, its fixed functional form may limit its ability to adapt fully to the specific details of each dataset, leading to greater variability in performance. The results underscore the strength of dictionary learning in dynamically identifying observables tailored to the underlying dynamics, offering both accuracy and stability with far fewer resources.

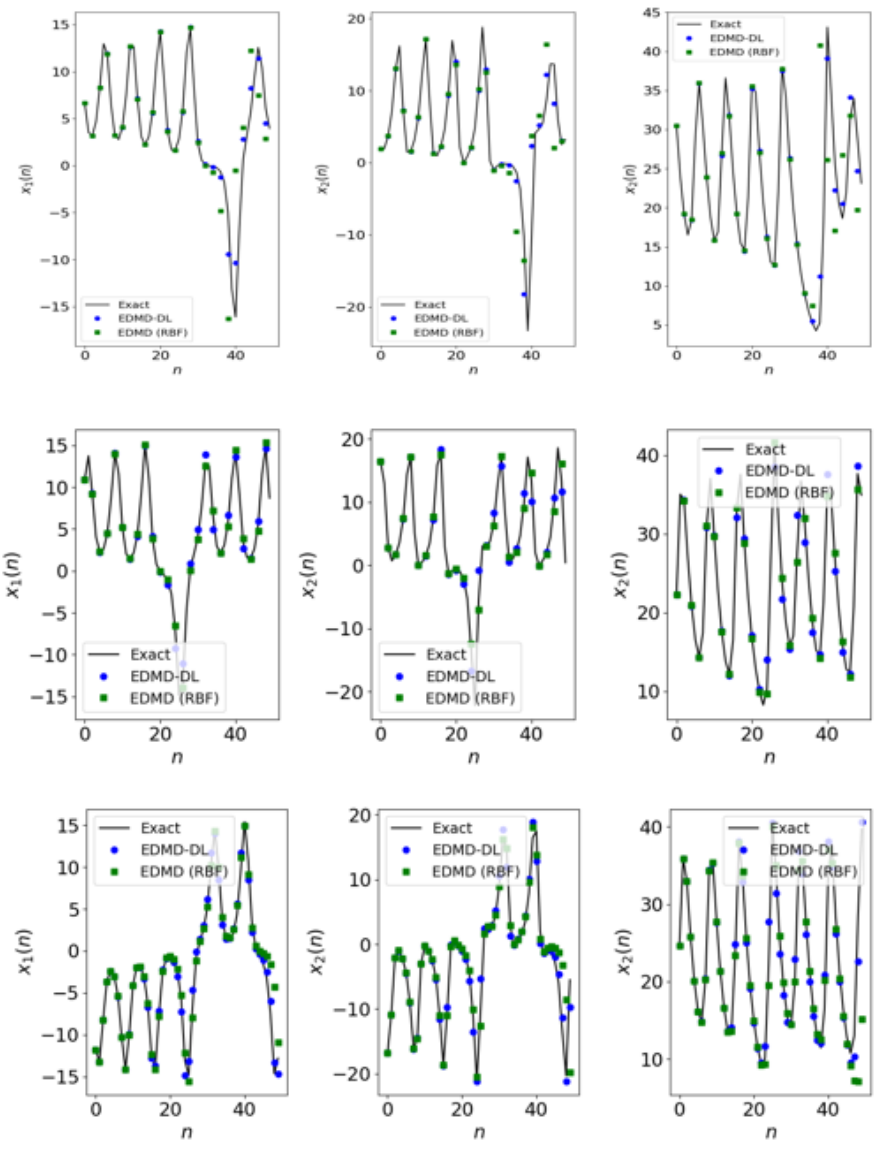

Figure 8.1 The real trajectory (black) next to the EDMD-DL reconstruction (blue dots) and RBFs reconstruction (green squares)

In Figure 8.1 we visualize the reconstructed trajectories, the dynamics are captured very well by both systems. This result underlines how well EDMD-DL adapts its dictionary without prior knowledge of the system, although some hyperparameter tuning for the training phase is required. Specifically, the number of training epochs, the size of the regularization term, and the proportions of training and validation data were tuned to achieve optimal performance. This tuning process was carried out empirically by testing several configurations and selecting the ones yielding the lowest reconstruction errors on validation data. Despite this additional step, the ability of EDMD-DL to dynamically adjust its dictionary enables it to compete effectively with a manually selected dictionary of 2500 observables while using only 27 observable functions.

Accuracy of eigenfunction approximations

We further compare the accuracy of the approximated eigenfunctions of the 2 methods, in Figure 8.2, we visualize the error E_j of the first 8 eigenfunctions on a single, very long (100000 iterations with time step $\Delta t = 0.1$) trajectory, using once again Lorenz attractor's ergodic properties. The eigenfunction errors of the EDMD-DL method are 1 to 2 orders of magnitude lower than the eigenfunction errors of EDMD with thin-plate splines (with the exception of the first eigenfunction, which corresponds to the constant function and has an error basically equal to zero). This result also proves that the subspace spanned by the dictionary identified by dictionary learning leans indeed to Koopman invariance.

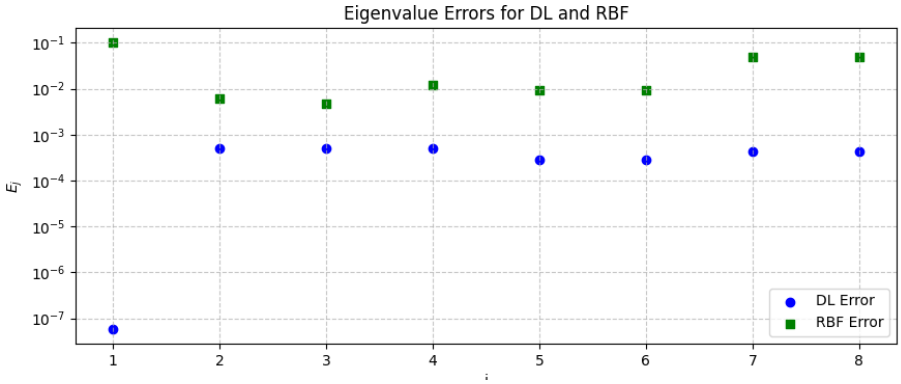

Figure 8.2 The first 8 eigenfunction errors of EDMD-DL (blue dots) and thin-plate splines (green squares)

In Figure 8.3 we visualize the eigenvalues identified by the 2 methods in the first run, it's interesting to notice that the set of eigenvalues found by EDMD-DL is very well approximated by a subset of the eigenvalues found by EDMD with thin-plate splines, this behaviour is observable in the other runs too.

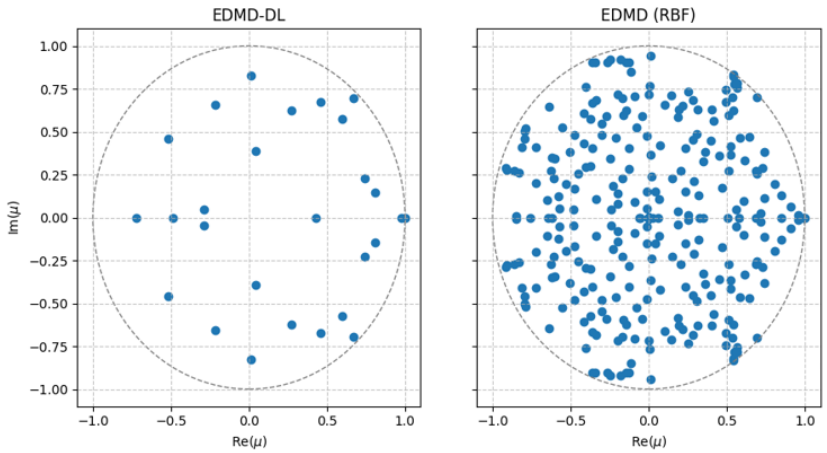

Figure 8.3 The eigenvalues identified by the EDMD-DL algorithm (left) and EDMD with 2500 thin-plate splines (right)

Visualization of eigenfunctions

Finally, we visualize the first six eigenfunctions identified by EDMD-DL in Figure 8.4, the first eigenfunction, similarly to the second one identified by the second order polynomials from Example 6.3.2 (see Figure 6.2), seems to correctly identify the two lobes of attraction with their corresponding unstable manifolds.

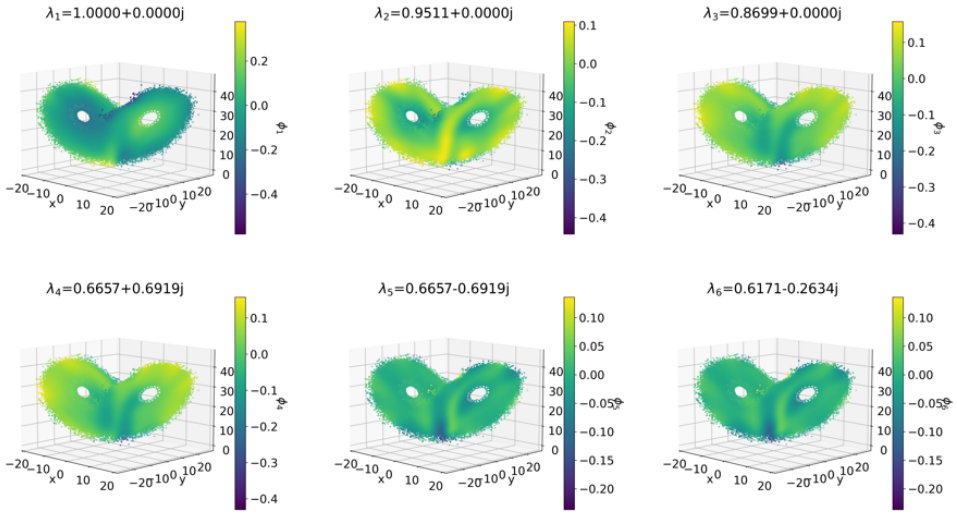

Figure 8.4 The first 6 eigenfunctions identified by the EDMD-DL algorithm

9 Conclusion

Summary

Throughout this thesis, we have systematically explored the theoretical foundations and practical applications of Dynamic Mode Decomposition (DMD) and its variants. Beginning with the derivation of DMD and its connection to Koopman operator theory, we established the mathematical framework that underpins these techniques. We then addressed various challenges associated with standard DMD by deriving and studying advanced variants, each tailored to overcome specific limitations such as noise sensitivity, computational complexity, and adaptability to nonlinear systems. We introduced Low-Rank DMD (lrDMD) and Sparse DMD (spDMD), two variants designed to address the curse of dimensionality, aiming to capture the most significant modes of high-dimensional systems while improving computational efficiency. The exploration of Optimized DMD (OptDMD) demonstrated how targeted optimization methods can significantly enhance noise resilience. In addition, Extended DMD (EDMD) and Kernelized EDMD were introduced to extend the capabilities of DMD to nonlinear systems by leveraging nonlinear dictionaries and kernel methods. The measure-preserving EDMD (mpEDMD) variant, along with its QR-factorized version, was discussed for studying measure-preserving dynamical systems, providing an effective method for analyzing continuous spectra. Finally, EDMD with Dictionary Learning (EDMD-DL) was presented as a promising technique for learning dynamical systems through neural networks, optimizing the representation of system states. The Lorenz system served as the primary testbed for demonstrating the utility and versatility of most of these methods.

Results and contributions

One of the most visually impactful contributions of this thesis is the novel visualization of the eigenfunctions of the Koopman operator on the Lorenz system, derived using EDMD with thin-plate splines as a dictionary (see Figure 6.6). These visualizations, depicted as color gradients on a very long trajectory, provide a unique perspective on the interplay between the system's chaotic dynamics and the spectral features of the Koopman operator. They not only highlight the power of EDMD in producing interpretable and high-quality representations of dynamical systems but also set a benchmark for future studies to explore more complex systems using similar techniques.

Another primary contribution of this work is the detailed spectral analysis of the Lorenz system using mpEDMD, which builds on Colbrook's work but introduces several new observations (see Example 7.4.1). Our analysis of the cumulative distribution functions (CDFs) of scalar-valued spectral measures has revealed important differences in the spectral behavior of the components of the Lorenz system, particularly highlighting the distinct roles of the components in the chaotic dynamics. The identification of the influence of low-phase eigenvalues, particularly in the x_3 component, is a significant step in linking spectral measures to the system's underlying chaotic behavior. Such insight furthers our understanding of the persistent and non-repetitive nature of the Lorenz attractor. Furthermore, the observations surrounding the spectral properties of the Lorenz system, such as the critical influence of low-phase eigenvalues, were corroborated across multiple methods, including the pioneering application of Sparse DMD, and Low-Rank DMD to the Lorenz system (see Example 4.3.1). These findings reinforce the persistent nature of the Lorenz attractor and its inherent connection to the Koopman spectral framework.

The formalized derivation of the mpEDMD algorithm via QR-factorization (see Subsection 7.1.2) addresses gaps in the existing literature and provides a more stable framework for analyzing measure-preserving systems.

Additionally, we have applied EDMD with Dictionary Learning (EDMD-DL) to the Lorenz system (see Example 8.3.1), a pioneering application that demonstrates the potential of machine learning techniques

in improving the predictive accuracy and robustness of DMD methods. Through careful parameter tuning, we have shown that EDMD-DL can enhance the understanding of chaotic dynamics and provide valuable insights into the system's evolution without any prior knowledge of the system's dynamics

Future Directions

One promising avenue for future research lies in combining measure-preserving EDMD (mpEDMD) with EDMD with Dictionary Learning (EDMD-DL). In this approach, one could first use dictionary learning techniques to train an optimal set of functions or representations of the system's dynamics. These trained functions could then serve as the input to mpEDMD, allowing for a more efficient and accurate analysis of measure-preserving systems, especially in high-dimensional settings. This hybrid method could potentially enhance the predictive power of mpEDMD by leveraging the adaptability of EDMD-DL to discover nearly Koopman-invariant dictionaries. The combination could also mitigate the limitations of both techniques individually, improving performance in noisy or high-dimensional data scenarios. While this hybrid approach holds great promise, it would be important to investigate the computational cost and scalability of combining mpEDMD with EDMD-DL, particularly in high-dimensional systems where the choice of dictionary and optimization procedures may present significant challenges. This consideration is crucial, as the complexity of implementing such an approach could increase substantially in large-scale or highly complex systems. Although this approach was not explored within the scope of this thesis due to space constraints, it represents an exciting direction that could significantly advance the applicability of DMD-based methods. Furthermore, exploring the scalability of this hybrid approach to larger, more complex systems could unveil new opportunities for analyzing chaotic or turbulent dynamics across a wide range of scientific domains.

Final reflection

This research has not only deepened my understanding of dynamical systems and their analysis but has also sharpened my skills in computational methods and algorithm development, providing a more comprehensive grasp of both the theory and practical challenges involved. The challenges faced and the problem-solving required throughout the journey have been instrumental in shaping my approach to scientific inquiry, as the work required combining optimization methods for the derivation and the application of the DMD algorithms, together with numerical methods for data generation, and machine learning methods for advanced variants such as EDMD-DL. The interdisciplinary nature of this work has taught me the value of integrating diverse tools and perspectives to tackle complex problems. As I navigated the intricacies of DMD and its variants, I came to appreciate the delicate balance between theory and computational feasibility. This experience has broadened my understanding not only of how powerful these methods can be but also of their limitations and the careful consideration needed when applying them to real-world data. A significant aspect of this thesis has been its potential as a learning tool for others interested in dynamical systems analysis and DMD-based methods. Through clear derivations and explanations of each variant, this work aims at serving as a comprehensive starting point for new researchers seeking to explore the complexities of DMD and its applications. This work has also deepened my appreciation for the importance of clear communication and pedagogy, as I learned how challenging yet rewarding it is to explain complex concepts in an accessible way. Looking ahead, I am excited about the potential to integrate machine learning with traditional dynamical systems analysis, which could lead to more accurate and efficient methods for predicting complex behaviors across various scientific disciplines. Reflecting on the journey, I feel more equipped to take on future challenges, knowing that scientific progress often requires a deep understanding of both theory and computation, along with the ability to adapt and innovate as new problems arise. The skills and insights gained here will undoubtedly shape my approach to future research, as well as my broader perspective on how computational techniques can be harnessed to understand and control complex dynamical systems.

A Appendix

A.1 Some Code Listings

A.1.1 Low-rank DMD class

```
from scipy.linalg import svd
import numpy as np
import matplotlib.pyplot as plt


class LRDMD:
    def __init__(self, snapshots: np.array, rank: int):
        self.snapshots = snapshots
        self.rank = rank

    def preprocessing_data(self):
        X = self.snapshots[:, :-1]
        Y = self.snapshots[:, 1:]
        return X, Y

    def compute_h(self):
        X, Y = self.preprocessing_data()

        # Compute economy-sized svd of the first N-1 snapshots
        # V_hm is the hermitian conjugation
        self.U, self.S, self.V_hm = svd(X, full_matrices=False)
        self.V = self.V_hm.conj().T
        # Compute the reduced matrix A tilde
        H = self.U.conj().T @ Y @ self.V

        return H

    def compute_low_rank_approximation(self):
        H = self.compute_h()
        Uh, Sh, Vh = svd(H, full_matrices=False)
        # Set the last N - rank singular values equal to zero to obtain
            rank r, we don't delete them to preserve
        # dimensions
        Sh[-(self.snapshots.shape[1]-self.rank):] = 0
        lr_operator = Uh @ np.diag(Sh) @ Vh @ np.linalg.inv(np.diag(
           self.S))
        return lr_operator


    def _optimal_dmd_matrices(self):
        lr_op = self.compute_low_rank_approximation()
        # Eigendecomposition of the low rank operator
        self.eigvals, self.eigvecs = np.linalg.eig(lr_op)
        self.modes = self.U @ self.eigvecs

        # Vandermonde matrix for time dynamics
```

```
        timesteps = self.snapshots.shape[1]
        self.vander = np.vander(self.eigvals, N=timesteps-1, increasing
            =True)

        # Construct P and q for the least squares problem
        P = np.multiply(
            np.dot(self.eigvecs.conj().T, self.eigvecs),
            np.conj(np.dot(self.vander, self.vander.conj().T)),
        )

        q = np.conj(
            np.diag(
                np.linalg.multi_dot(
                    [
                        self.vander,
                        self.V,
                        np.diag(self.S).conj(),
                        self.eigvecs,
                    ]
                )
            )
        )
        return P, q

    def compute_optimal_amplitudes(self):
        P, q = self._optimal_dmd_matrices()
        amplitudes = np.linalg.solve(P, q)
        return amplitudes

    def reconstruct_data(self):
        amplitudes = self.compute_optimal_amplitudes()
        ampl_matrix = np.diag(amplitudes)
        reconstructed_data = self.modes @ ampl_matrix @ self.vander
        return reconstructed_data

    def obtain_eigenvalues(self):
        # Ensure eigenvalues are computed
        if not hasattr(self, 'eigvals'):
            self.compute_optimal_amplitudes()
        nonzero_eigvals = self.eigvals[:self.rank]
        return nonzero_eigvals
```

A.1.2 MpEDMD and EDMD class

```
import numpy as np
import scipy as sp

class MPEDMD:
    def __init__(self, psi_X, psi_Y, W=None):
        self.G = None
        self.A = None
        self.psi_X = psi_X
        self.psi_Y = psi_Y
        self.W = W
        self.K_edmd = None
        self.mp_K = None
        self.mp_K_qr = None
```

```
        self.edmd_evals = None
        self.edmd_levecs = None
        self.edmd_revecs = None
        self.mp_evals_qr = None
        self.mp_evecs_qr = None
        self.mp_evals = None
        self.mp_evecs = None

    def assemble_G_A(self):
        """Assembles Gram matrix G and A"""
        if self.W is None:
            """For random and ergodic sampling"""
            self.G = (self.psi_X.conj().T @ self.psi_X) / self.psi_X.
               shape[0]
            self.A = (self.psi_X.conj().T @ self.psi_Y) / self.psi_X.
               shape[0]
        elif len(self.W) == self.psi_X.shape[0]:
            """For quadrature rules"""
            self.G = self.psi_X.conj().T @ np.diag(self.W) @ self.psi_X
            self.A = self.psi_X.conj().T @ np.diag(self.W) @ self.psi_Y
        else:
            raise ValueError('Length of the quadrature weight array W
               does not correspond to dimension of data matrix'
                             'psi_X')
        return self.G, self.A

    def mp_edmd(self, truncation_rank=None, epsilon=1e-14):
        if self.G is None or self.A is None:
            self.G, self.A = self.assemble_G_A()

        self.G = (self.G + self.G.conj().T) / 2  # Symmetrize G to
           ensure it's Hermitian

        # Eigenvalue decomposition
        G_evals, G_evecs = np.linalg.eig(self.G)

        # Handle near-zero or negative eigenvalues by thresholding
        G_evals[G_evals < epsilon] = epsilon  # Ensure no eigenvalue is
            below the threshold

        # Compute G^1/2 and G^{-1/2} with thresholded eigenvalues
        G_sqrt = G_evecs @ np.diag(np.sqrt(G_evals)) @ G_evecs.conj().T
        G_sqrt_inv = G_evecs @ np.diag(np.sqrt(1 / G_evals)) @ G_evecs.
           conj().T

        # Compute SVD of G_sqrt_inv @ A.T @ G_sqrt_inv
        U1, Sigma, U2h = sp.linalg.svd(G_sqrt_inv @ self.A.conj().T @
           G_sqrt_inv)
        U2 = U2h.conj().T

        if truncation_rank is not None:
            U1 = U1[:, :truncation_rank]
            U2 = U2[:, :truncation_rank]
            Sigma = Sigma[:truncation_rank]  # Ensure Sigma is
               truncated too

        # Perform Schur decomposition
```

```
        mp_evals, mp_evecs = sp.linalg.schur(U2 @ U1.conj().T, output='complex')

        # Compute Koopman operator
        self.mp_K = G_sqrt_inv @ U2 @ U1.conj().T @ G_sqrt
        self.mp_evecs = G_sqrt_inv @ mp_evecs
        self.mp_evals = np.diag(np.diag(mp_evals))  # Diagonalize eigenvalue matrix

        return self.mp_K, self.mp_evecs, self.mp_evals

    def mp_edmd_qr(self):
        if self.W is None:
            self.W = np.ones(self.psi_X.shape[0])/self.psi_X.shape[0]

        W_sqrt = np.sqrt(self.W)

        Q, R = np.linalg.qr(np.diag(W_sqrt) @ self.psi_X)
        T = np.linalg.lstsq(R.conj().T, self.psi_Y.conj().T @ np.diag(W_sqrt) @ Q, rcond=None)[0]
        U1, Sigma, U2h = np.linalg.svd(T)
        U2 = U2h.conj().T
        mp_evals, mp_evecs = sp.linalg.schur(U2 @ U1.conj().T, output='complex')
        self.mp_K_qr = np.linalg.lstsq(R, U2 @ U1.conj().T, rcond=None)[0] @ R
        self.mp_evecs_qr = np.linalg.lstsq(R, mp_evecs)[0]
        self.mp_evals_qr = np.diag(np.diag(mp_evals))
        return self.mp_K_qr, self.mp_evecs_qr, self.mp_evals_qr

    def edmd(self, truncation_rank=None):
        if self.G is None or self.A is None:
            self.G, self.A = self.assemble_G_A()

        self.G = (self.G + self.G.conj().T) / 2  # Symmetrize G to ensure it's Hermitian

        # Use least squares to solve for K
        K, residuals, rank, s = np.linalg.lstsq(self.G, self.A, rcond=None)

        # Perform eigenvalue decomposition
        eigenvalues, right_eigenvectors = np.linalg.eig(K)

        # Sort eigenvalues and right eigenvectors
        sorted_indices = np.argsort(np.abs(eigenvalues))[::-1]
        eigenvalues = eigenvalues[sorted_indices]
        self.edmd_revecs = right_eigenvectors[:, sorted_indices]

        if truncation_rank is not None:
            eigenvalues = eigenvalues[:truncation_rank]
            self.edmd_revecs = self.edmd_revecs[:, :truncation_rank]

        # Using pseudo-inverse for left eigenvectors
        self.edmd_levecs = np.linalg.pinv(self.edmd_revecs)

        # Construct truncated K
        self.edmd_evals = np.diag(eigenvalues)
```

```
114         self.K_edmd = self.edmd_revecs @ self.edmd_evals @ self.
                edmd_levecs
115
116         return self.K_edmd, self.edmd_revecs, self.edmd_levecs, self.
                edmd_evals
117
118     def compute_power_mpK(self, exponent: int, truncation_rank=None):
119         # Check if mpK is already computed, if not run mp_edmd
120         if self.mp_K is None:
121             print("mp_edmd not yet run, computing mp_edmd now...")
122             self.mp_edmd(truncation_rank)
123
124         # Raise the eigenvalues to the desired power
125         eigenvalues_power = np.diag(self.mp_evals) ** exponent
126
127         # Reconstruct the Koopman operator for the given power
128         K_power = self.mp_evecs @ np.diag(eigenvalues_power) @ np.
                linalg.pinv(self.mp_evecs)
129
130         return K_power
131
132     def compute_power_mpK_qr(self, exponent: int):
133         # Check if mpK_qr is already computed, if not run mp_edmd_qr
134         if self.mp_K_qr is None:
135             print("mp_edmd_qr not yet run, computing mp_edmd_qr now..."
                    )
136             self.mp_edmd_qr()
137
138         # Raise the eigenvalues to the desired power
139         eigenvalues_power = np.diag(self.mp_evals_qr) ** exponent
140
141         # Reconstruct the Koopman operator for the given power
142         K_power = self.mp_evecs_qr @ np.diag(eigenvalues_power) @ np.
                linalg.pinv(self.mp_evecs_qr)
143
144         return K_power
145
146     def compute_power_edmd(self, exponent: int, truncation_rank=None):
147         # Check if K_edmd is already computed, if not run edmd
148         if self.K_edmd is None:
149             print("EDMD not yet run, computing EDMD now...")
150             self.edmd(truncation_rank)
151
152         # Raise the eigenvalues to the desired power
153         eigenvalues_power = np.diag(self.edmd_evals) ** exponent
154
155         # Reconstruct the Koopman operator for the given power
156         K_power = self.edmd_revecs @ np.diag(eigenvalues_power) @ self.
                edmd_levecs
157
158         return K_power
```

List of Figures

List of Tables

Bibliography

[1] V. V. S. A. N. Tikhonov A. V Goncharsky and A. G. Yagola. *Numerical Methods for the Solution of Ill-Posted Problems*. Springer Science and Business Media, 2013.

[2] S. I. Abramowitz M. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Applied Mathematics Series. Vol. 55 (Ninth reprint with additional corrections of tenth original printing with corrections (December 1972); first ed.). Washington D.C.; New York: United States Department of Commerce, National Bureau of Standards; Dover Publications., June 1964. ISBN: 978-0-486-61272-0.

[3] T. Askham and J. N. Kutz. "Variable Projection Methods for an Optimized Dynamic Mode Decomposition". In: *SIAM Journal on Applied Dynamical Systems* 17.1 (2018), pp. 380–416. eprint: https://doi.org/10.1137/M1124176.

[4] H. B. B. S. Nagy C. Foias and L. Kérchy. *Harmonic Analysis of Operators on Hilbert Space*. New York: Springer, 2010.

[5] G. Baudat and F. Anouar. "Kernel-based methods and function approximation". In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. Vol. 2. 2001, 1244–1249 vol.2.

[6] C. Bishop. "Pattern Recognition and Machine Learning". In: *Information Science and Statistics* ((2006)).

[7] K. BO. "Hamiltonian Systems and Transformation in Hilbert Space." In: *Proc Natl Acad Sci U S A.* (1931), pp. 315–318.

[8] C. J. Burges. "A tutorial on support vector machines for pattern recognition". In: *Data Mining and Knowledge Discovery, 2* ((1998)), (pp. 121–167).

[9] K. Champion et al. "Data-driven discovery of coordinates and governing equations". In: *Proceedings of the National Academy of Sciences* 116.45 (Oct. 2019), 22445–22451. ISSN: 1091-6490.

[10] M. J. Colbrook. "The mpEDMD Algorithm for Data-Driven Computations of Measure-Preserving Dynamical Systems". In: *SIAM Journal on Numerical Analysis* 61.3 (2023), pp. 1585–1608. eprint: https://doi.org/10.1137/22M1521407.

[11] M. J. Colbrook, L. J. Ayton, and M. Szőke. "Residual dynamic mode decomposition: robust and verified Koopmanism". In: *Journal of Fluid Mechanics* 955 (Jan. 2023). ISSN: 1469-7645.

[12] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[13] M. Dellnitz, G. Froyland, and O. Junge. "The Algorithms Behind GAIO — Set Oriented Numerical Methods for Dynamical Systems". In: *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*. Springer, 2001, (pp. 145–174).

[14] D. Donoho. "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306.

[15] L. G.-R. "Meshfree methods: Moving beyond the finite element method". In: *CRC Press, Boca Raton* ((2010)).

[16] S. P. Ghanem R. "Polynomial Chaos in Stochastic Finite Elements". In: *J. Appl. Mech.* 57 (Mar. 1990), pp. 197–202.

[17] G. H. Golub and R. J. LeVeque. “Extensions and Uses of the Variable Projection Algorith for Solving Nonlinear Least Squares Problems”. In: *Proceedings of the 1979 Army Numerical Analsysis and Computers Conference* (1979).

[18] G. H. Golub and V. Pereyra. “The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems Whose Variables Separate”. In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 413–432. eprint: https://doi.org/10.1137/0710036.

[19] G. Golub and V. Pereyra. “Separable nonlinear least squares: the variable projection method and its applications”. In: *Inverse Problems* 19.2 (2003), R1.

[20] B. M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Low-rank and sparse dynamic mode decomposition”. In: *Center for Turbulence Research Annual Research Briefs 2012* (2012).

[21] B. M. R. Jovanović, P. J. Schmid, and J. W. Nichols. “Sparsity-promoting dynamic mode de- composition.” In: *Physics of Fluids* (2014), 26(2).

[22] B. J.P. “Chebyshev and Fourier Spectral Methods”. In: *Courier Dover Publications* (2013).

[23] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods for Computational Fluid Dynamics*. Oxford University Press, June 2005. ISBN: 9780198528692.

[24] L. Kaufman. “A variable projection method for solving separable nonlinear least squares problems”. In: *BIT* 15.1 (1975), 49–57. ISSN: 0006-3835.

[25] K. LEVENBERG. “A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES”. In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. ISSN: 0033569X, 15524485.

[26] Q. Li et al. “Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.10 (Oct. 2017), p. 103111. ISSN: 1054-1500. eprint: https://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/1.4993854/13247075/103111_1_online.pdf.

[27] E. N. Lorenz. “Deterministic Nonperiodic Flow”. In: *Journal of Atmospheric Sciences* 20.2 (1963), pp. 130 –141.

[28] M. I. P. F. Luzzatto S. “Residual dynamic mode decomposition: robust and verified Koopmanism”. In: *Commun. Math. Phys. 260* 955 (2005), 393–401. ISSN: 1469-7645.

[29] D. W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. eprint: https://doi.org/10.1137/0111030.

[30] A. Y. Ng. “Feature selection, L1 vs. L2 regularization, and rotational invariance”. In: ICML ’04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. ISBN: 1581138385.

[31] D. O’Leary and B. Rust. “Variable Projection for Nonlinear Least Squares Problems”. en. In: *ACM Transactions on Mathematical Software* (2012).

[32] C. E. Rasmussen. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[33] C. W. Rowley et al. “Spectral analysis of nonlinear flows”. In: *Journal of Fluid Mechanics* 641 (2009), pp. 115 –127.

[34] Y. Saad. “Variations on Arnoldi’s method for computing eigenelements of large unsymmetric matrices”. In: *Linear Algebra and its Applications* 34 (1980), pp. 269–295. ISSN: 0024-3795.

[35] P. J. SCHMID and J. SESTERHENN. “Dynamic mode decomposition of numerical and experimental data”. In: *Sixty-First Annual Meeting of the APS Division of Fluid Dynamics* 656 (2008).

[36] P. J. SCHMID. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* (2010), 5–28.

[37] B. Schölkopf. “The Kernel Trick for Distances”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000.

[38] C. Schütte, P. Koltai, and S. Klus. “On the numerical approximation of the Perron-Frobenius and Koopman operator”. In: *Journal of Computational Dynamics* 3.1 (Sept. 2016), 1–12. ISSN: 2158-2491.

[39] B. T. et al. “Meshless methods: an overview and recent developments”. In: *Comput. Methods Appl. Mech. Eng. 139(1), pp. 3-47* ((1996)).

[40] L. Trefethen. “5. Polynomial Interpolation and Clustered Grids”. In: *Spectral Methods in MATLAB*. vol. 10. SIAM, 2000, pp. 41–50. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898719598.ch5.

[41] J. H. Tu et al. “On dynamic mode decomposition: Theory and applications”. In: *Journal of Computational Dynamics* 1.2 (2014), pp. 391–421. ISSN: 2158-2491.

[42] H. Wendland. “Meshless Galerkin Methods Using Radial Basis Functions”. In: *Mathematics of Computation* 68 (May 1997).

[43] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. “A Data–Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition”. In: *Journal of Nonlinear Science* 25.6 (June 2015), 1307–1346. ISSN: 1432-1467.

[44] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis. *A Kernel-Based Approach to Data-Driven Koopman Spectral Analysis*. 2015. arXiv: 1411.2260 [math.DS].