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

Improving power iteration algorithms for Koopman operator approximation

Hajime Sekiya

Thesis for the attainment of the academic degree

Master of Science

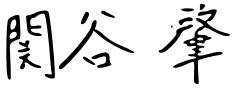
at the TUM School of Computation, Information and Technology of the Technical University of Munich

Supervisor:
Prof. Dr. Hans-Joachim Bungartz

Advisors:
Dr. Felix Dietrich

Submitted:
Munich, 30. November 2023

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, 30. November 2023 Hajime Sekiya 

Zusammenfassung

Das dynamische System oder die Dynamik ist ein mathematisches Konzept, das zur Modellierung und Analyse des Verhaltens von Systemen verwendet wird, die sich im Laufe der Zeit entwickeln. Seine breite Anwendbarkeit erstreckt sich über verschiedene Bereiche, darunter Physik, Biologie, Wirtschaft, Technik, Informatik und Mathematik. Das zentrale mathematische Konzept der Analyse dynamischer Systeme ist der Koopman-Operator, ein linearer Operator, der häufig auf infinite-dimensionalen Funktionsräumen definiert wird und nichtlineare Dynamik in einem linearen Kontext erfassen kann.

Auf der Grundlage der Koopman-Modus-Analyse haben sich die Dynamic Mode Decomposition (DMD) und ihre Varianten als datengesteuerte Algorithmen herauskristallisiert, die die Entwicklung vieler dynamischer Systeme anhand gegebener Daten, in der Regel Momentaufnahmen von Beobachtungen, vorhersagen können. Sie approximieren den Koopman-Operator durch die Matrixdarstellung im endlich dimensionalen Raum, und ihre spektrale Information wird zur Vorhersage der Entwicklung der Dynamik verwendet.

Der Koopman-Operator hat eine bemerkenswerte Eigenschaft: Das Produkt oder die Potenzierung von Eigenpaaren bildet ein weiteres Eigenpaar. Diese Eigenschaft bedeutet, dass man, sobald man einige Eigenpaare erfasst hat, ohne weiteres weitere Eigenpaare durch ihr Produkt oder ihre Potenzierung ableiten kann. Diese einzigartige Eigenschaft kann die Effizienz des gesamten Prozesses der Eigenwertzerlegung in DMD-Algorithmen erhöhen oder sogar Eigenpaare liefern, deren Informationen nicht im projizierten Raum enthalten sind.

In dieser Arbeit wird ausschließlich auf ergodische Systeme konzentriert, in denen zusätzliche Eigenschaften nachgewiesen werden können, die zur Weiterentwicklung von Algorithmen beitragen. Durch die Einbeziehung dieser Eigenschaften in den Eigenlöser werden in dieser Arbeit zwei exponentielle Algorithmen zur Berechnung der Eigenvektoren der Matrixdarstellung des Koopman-Operators oder der Eigenfunktionen des Koopman-Operators mit ihren numerischen Experimenten vorgestellt.

Abstract

The dynamical system, or dynamics, is a mathematical concept used to model and analyze the behavior of systems that evolve over time. Its broad applicability spans diverse domains, including physics, biology, economics, engineering, computer science, and mathematics. The central mathematical concept of dynamical system analysis is the Koopman operator, which is a linear operator often defined on infinite-dimensional function spaces and can capture non-linear dynamics in a linear context.

Based on the Koopman mode analysis, dynamic mode decomposition (DMD) and its variants emerged as data-driven algorithms that can predict the evolution of many dynamical systems just by given data, typically snapshots of observations. They approximate the Koopman operator by the matrix representation on finite-dimensional space, and their spectral information is used to forecast the evolution of dynamics.

The Koopman operator has a remarkable property: the product or exponentiation of eigenpairs forms yet another eigenpair. This property implies that once some eigenpairs are acquired, one can readily derive other eigenpairs through their product or exponentiation. This unique property can potentially enhance the efficiency of the entire eigendecomposition process performed in DMD-type algorithms or even yield eigenpairs whose information is not contained within the projected space.

In this thesis, our focus is exclusively directed towards ergodic systems, where additional properties can be demonstrated, contributing to the advancement of algorithms. By incorporating these attributes into the eigensolver, this thesis introduces two exponential algorithms designed to compute the eigenvectors of the Koopman operator's matrix representation or the eigenfunctions of the Koopman operator with their numerical experiments.

Contents

1 Introduction

The dynamical system, also referred to as dynamics, is a mathematical concept used to model and analyze the behavior of systems that evolve over time. At its core, a dynamical system is characterized by its state variables and the rules or equations governing their evolution. This system can be discrete or continuous and may represent anything from simple mechanical systems, like a swinging pendulum [Str15], to complex phenomena like population dynamics [BCC14], weather patterns [Han22], economic trends [BCS14], and numerical algorithms [Chu08]. Researchers use the dynamical system theory in various fields, such as physics, biology, economics, engineering, computer science, and mathematics, to gain insights into the behavior of dynamic phenomena. It provides a powerful framework for understanding and predicting the evolution of systems, making it a vital concept in both theoretical and applied mathematics.

For a long time, the dynamical system has been analyzed through the concept of "dynamics of states" rooted in the contributions of Poincaré [Via01], in which dynamics are analyzed directly at the state space trajectories. However, this type of analysis showed difficulties in handling high-dimensional, ill-described, and uncertain systems [Jon01; MB12]. Later in the stage, an alternative aspect called "dynamics of observables" arose, in which dynamics are analyzed through some so-called observable functions, which are defined on the state space, and the evolution of these functions' values [MB12]. In this framework, the central of the mathematical concept is the so-called Koopman operator, originally introduced by Koopman and von Neumann [Koo31; KN32], which is a linear operator often defined on infinite-dimensional function spaces and is able to capture non-linear dynamics in a linear sense.

The Koopman mode analysis generalizes linear mode analysis from linear system to nonlinear system without neglecting the global nonlinear feature of the system, unlike other methodologies such as Tylor and Fourier expansion [MB12; Mez05; ROW+09]. These analyses showed, with the help of the spectral theory [Bor20], that if the observable function is spanned by the eigenfunctions of the Koopman operator, then the evolution of systems can be decomposed into just three terms: the eigenvalues, the eigenfunctions, and the so-called Koopman modes of the Koopman operator. Even if they are not exactly spanned by the eigenfunctions, this decomposition works as the approximation of the Koopman operator while ignoring the continuous part of the spectrum. These results motivate researchers to study algorithms for approximating these three terms just by some parts of the dynamical systems' data. Typically, the dataset is a sequence of observations through a vector-valued observable along a trajectory. The Dynamic Mode Decomposition (DMD) and its variants [AM17; Col23; SCH10; WKR15], the detail is explained in the state of the arts section (section 2), are the commonly used algorithms that approximate the Koopman operator by its matrix representation on the finite-dimensional space and by eigenvalues and eigenvectors of the matrix. Thus, the calculation cost, the quality, and the quantity of the eigenpairs of the matrix representation are critical for the algorithms. These data-driven algorithms are successfully used in the above-mentioned wide range of application areas, such as finance [MK16], brain networks [Par+23], and numerical algorithms [DTK20].

Here, the Koopman operator has a remarkable property: for pairs of eigenvalues and eigenfunctions of the Koopman operator, denoted as (λ_1, φ_1) and (λ_2, φ_2), their product $(\lambda_1\lambda_2, \varphi_1\varphi_2)$ defined as $[\varphi_1\varphi_2](x) := \varphi_1(x)\varphi_2(x)$ forms yet another eigenpair. Moreover, by natural extension from product to power, for any real positive number $p \in \mathbb{R}_+$, $(\lambda_1^p, \varphi_1^p)$ also represents an eigenpair. If the eigenfunction φ_1 vanishes nowhere, then $\varphi_1^{-1} := \frac{1}{\varphi_1}$ is well defined and the same holds with $p \in \mathbb{R}$. This property implies that once some eigenpairs are acquired, one can readily derive other eigenpairs through their product or exponentiation. This unique property holds the potential to enhance the efficiency of the entire eigendecomposition process performed in the DMD-type algorithms because finding eigenvectors is often computationally costly, whereas taking exponential or combinations is not. Furthermore, it enables us to obtain the eigenpairs (eigenvalues and eigenfunctions) of the Koopman operator, whose information is neglected on the projected finite-dimensional space, i.e., there does not exist any corresponding eigenvalues and eigenvectors of the matrix representation.

In this thesis, we further restrict our attention only to ergodic systems, which are dynamical systems that, over time, explore and visit all possible states within their phase space (state space) with equal probability. These systems are essential in various fields, such as statistical mechanics and information theory.

Examples of such systems are Brownian motion [Ugb80] and certain chaotic systems, like the Lorenz system [ER85]. Under ergodic settings, one can further show more useful theorems, for example, the well-known Birkhoff's ergodic theorem [Bir31], and other properties of the spectral information of the Koopman operator, such as all eigenvalues lie on the unit circle, and each eigenvalue is of multiplicity 1. These theorems and properties can also be integrated into our algorithm to enhance the quality and efficiency of the eigensolver.

To summarize, this thesis aims to integrate these special properties into the eigendecomposition processes performed inside the DMD-type algorithms and to enhance the efficiency of the entire calculation procedure, surpassing the current methodologies in the sense of the number of computational arithmetic operations or the reconstruction error quality. The remainder of the thesis is organized as follows. We introduce the related state of the arts in the section 2, in the section 3, we explain our algorithms, which we call the real-exponential algorithm and the integer-exponential algorithm, and lastly, we state our conclusion of this thesis in the section 4.

2 State of the art

2.1 Dynamical systems, Koopman operator, and Ergodic theory

We first introduce the mathematical formulation of the dynamical systems, the Koopman operator, and Ergodic theory with their notation for the entire thesis. Note that we basically use the same notation as Budišić, Mohr, and Mezić [MB12].

Dynamical systems The dynamical system (or dynamics) is defined by a set $\mathcal{M}$ called the state space or the phase space and a map $T : \mathcal{M} \to \mathcal{M}$ called the evolution function. Typically, one considers the case where $\mathcal{M}$ is a measurable space, with a σ-algebra $\mathfrak{B}$, and T is $\mathfrak{B}$-measurable. In this thesis, we further assume that the evolution function T is invertible and measure-preserving, i.e., there exists an invariant measure μ such that for any $S \in \mathfrak{B}$,

$$\mu(S) = \mu(T^{-1}S),$$

where $T^{-1}S$ is understood as the pre-image of the set $S \in \mathfrak{B}$, i.e.,

$$T^{-1}S := \{x \in \mathcal{M} \mid T(x) \in S\}.$$

Moreover, we assume that the measure is normalized and complete, i.e., $\mu(\mathcal{M}) = 1$, and $S_1 \subset S_2 \in \mathfrak{B}, \mu(S_2) = 0$ implies $S_1 \in \mathfrak{B}$. In total, we denote the dynamical system as $(\mathcal{M}, \mathfrak{B}, \mu, T)$.

An example of such space $(\mathcal{M}, \mathfrak{B}, \mu)$ is, for instance, given by Cornfeld, Fomin, and Sinai [CFS12], which is m-dimensional torus, where μ is a normalized Harr measure, $\mathcal{M} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^1$, and $\mathfrak{B}$ is the completion of the σ-algebra of Borel sets of the space $\mathcal{M}$, on which the natural cyclic coordinates $x_1, \ldots, x_m$ have been introduced. In these coordinates $d\mu = dx_1 \ldots dx_m$.

A dynamical system can be either discrete or continuous. In the discrete case, the state evolution is written as the form

$$p_{n+1} = T(p_n),$$

where $p_n \in \mathcal{M}$ is the state at time n. In the continuous case, it is represented as

$$\dot{p} = T(p).$$

Throughout this thesis, we only consider a discrete-time dynamical system.

When one considers approximating the dynamics by a given dataset, one typically does not have direct access to the state evolution data $\{p_0, p_1, p_2, \ldots\}, p_n \in \mathcal{M}$ of the dynamics. Instead, one observes the evolution of state through some observable, defined as a complex-valued function $f \in \mathcal{F}$ where $\mathcal{F}$ is a set of complex-valued functions defined as $\mathcal{F} := \{f \mid f : \mathcal{M} \to \mathbb{C}\}$. In other words, for the initial state $p_0 \in \mathcal{M}$ and its evolution

$$\{p_0, p_1, p_2, \ldots, p_n\} = \left\{p_0, T(p_0), T^2(p_0), \ldots, T^n(p_0)\right\},$$

one obtains the dataset of the dynamics as a trace data, denote as $X \in \mathbb{C}^n$, defined by

$$X := \{f(p_0), f(p_1), f(p_2), \ldots, f(p_{n-1})\} = \left\{f(p_0), f(T(p_0)), f(T^2(p_0)), \ldots, f(T^{n-1}(p_0))\right\}.$$

Note, if one has direct access to the state evolution dataset, one can simply define the observable function f as the identity function $f(x) := x$. In this case, it is easy to see that the trace data X coincides with the state evolution data.

Koopman operator Here, the Koopman operator U_T comes in, which is based on the contributions of Koopman and von Neumann [Koo31; KN32]. The Koopman operator is simply a composition operator and is defined as

$$U_T : \mathcal{F} \to \mathcal{F}, U_T f := f \circ T.$$

With this concept, $f(T^k(x)) = [U_T f](T^{k-1}(x)) = \cdots = [U_T^k f](x)$, and therefore, the trace data X is equal to

$$X = \left\{[U_T^0 f](p_0), [U_T^1 f](p_0), [U_T^2 f](p_0), \ldots, [U_T^{n-1} f](p_0)\right\},$$

where U_T^0 is understood as an identity operator. In other words, each data point of the trace data X can be represented as the evolution of the Koopman operator. From now on, we denote the Koopman operator as $U_T = U$ wherever the evolution function is clear.

An important property of the Koopman operator is that it is always linear, although the evolution function T of the underlying dynamics might be non-linear. Because for $f_1, f_2 \in \mathcal{F}$ and $\lambda \in \mathbb{C}$,

$$U(\lambda f_1 + f_2) = (\lambda f_1 + f_2) \circ T = \lambda f_1 \circ T + f_2 \circ T = \lambda U f_1 + U f_2.$$

This property enables us to analyze non-linear dynamics in a linear sense. In this thesis, we further assume U is bounded and thus continuous. Furthermore, for eigenpairs $\{(\lambda_j, \phi_j)\}_j$ of the Koopman operator U, assuming $f \in \mathcal{F}$ is spanned by the eigenfunctions $\{\phi_j\}_j$ i.e. $f(x) = \sum_j v_j \phi_j(x)$ where $v_j \in \mathbb{C}$, then

$$[Uf](p) = f(T(p)) = \sum_j v_j \phi_j(T(p)) = \sum_j v_j [U\phi_j](p) = \sum_j \lambda_j v_j \phi_j(p).$$

Analogously,

$$[U^k f](p) = \sum_j v_j [U^k \phi_j](p) = \sum_j v_j \lambda_j [U^{k-1} \phi_j](p) = \cdots = \sum_j \lambda_j^k v_j \phi_j(p).$$

Hence, the trace data X can be represented as

$$X = \left\{\sum_j \lambda_j^0 v_j \phi_j(p_0), \sum_j \lambda_j^1 v_j \phi_j(p_0), \sum_j \lambda_j^2 v_j \phi_j(p_0), \ldots, \sum_j \lambda_j^n v_j \phi_j(p_0)\right\}.$$

In other words, the n-th trace data can be expressed just by three terms: eigenvalues $\{\lambda_j\}_j$, eigenfunctions $\{\varphi_j\}_j$, and the coefficients $\{v_j\}_j$ of the observable function with respect to the eigenfunctions. Therefore, the coefficients v_j are defined and called the Koopman modes.

Now, for notational convenience, we define vector-valued observables $F := \begin{pmatrix} f_1 & \ldots & f_K \end{pmatrix}^T \in \mathcal{F}^K$ and an extension of the Koopman operator $U_K : \mathcal{F}^K \to \mathcal{F}^K$, where

$$[U_K F](x) = \begin{pmatrix} [Uf_1](x) \\ \vdots \\ [Uf_K](x) \end{pmatrix}.$$

Note that the same results hold as the single element case since the operator is applied to each vector element. For eigenfunctions $\{\phi_j\}$, observable function $f_k \in \mathcal{F}$ spanned by these eigenfunctions, and the Koopman modes v_{jk} with respect to the ϕ_j,

$$[U_K^n F](x) = \begin{pmatrix} [U^n f_1](x) \\ \vdots \\ [U^n f_K](x) \end{pmatrix} = \begin{pmatrix} \sum_j \lambda_j^n v_{j1} \phi_j(x) \\ \vdots \\ \sum_j \lambda_j^n v_{jK} \phi_j(x) \end{pmatrix} = \sum_j \lambda_j^n \phi_j(x) \begin{pmatrix} v_{j1} \\ \vdots \\ v_{jK} \end{pmatrix}.$$

Thus, in this notation, the Koopman modes are also vector-valued. Note that we denote $U_K = U$ if it is clear by input whether it is vector-valued or single-valued.

Lastly, as it is explained in the introduction section (section 1), the following proposition 2.1 of the Koopman operator is shown by Budišić, Mohr, and Mezić [MB12].

Proposition 2.1. *Assume the function space $\mathcal{F}$ forms a vector space which is closed under pointwise products of functions, i.e., $\forall f_1, f_2 \in \mathcal{F}$, $f_1 f_2 \in \mathcal{F}$. Then, the set of eigenfunctions forms an Abelian semigroup under pointwise products of functions. In particular, if $\varphi_1, \varphi_2 \in \mathcal{F}$ are eigenfunctions of U with eigenvalues λ_1 and λ_2 , then $\varphi_1\varphi_2$ where $[\varphi_1\varphi_2](x) := \varphi_1(x)\varphi_2(x)$ is an eigenfunction of U with eigenvalue $\lambda_1\lambda_2$ i.e. $U(\varphi_1\varphi_2) = \lambda_1\lambda_2\varphi_1\varphi_2$. Furthermore, if $p \in \mathbb{R}^+$ and ϕ is an eigenfunction with eigenvalue λ, then ϕ^p is a eigenfunction with eigenvalue λ^p, where $\phi^p(x) := (\phi(x))^p$.*

Proof. First of all, for eigenfunctions $\varphi_1, \varphi_2 \in \mathcal{F}$ of the Koopman operator U with corresponding eigenvalues λ_1, λ_2, define their product $\varphi_1\varphi_2$ as $[\varphi_1\varphi_2](x) := \varphi_1(x)\varphi_2(x)$, then,

$$[U\varphi_1\varphi_2](x) = \varphi_1(T(x))\varphi_2(T(x)) = [U\varphi_1](x)[U\varphi_2](x) = (\lambda_1\lambda_2)[\varphi_1\varphi_2](x),$$

i.e., $(\lambda_1\lambda_2, \varphi_1\varphi_2)$ is also an eigenfunction of U. Therefore, one can consider for $n, m \in \mathbb{N}$, the integer power φ_1^n, the rational power $\varphi_1^{\frac{n}{m}}$ such that $\left(\varphi_1^{\frac{n}{m}}\right)^m = \varphi_1^n$, and lastly for irrational power, take the limit of rational numbers such that they squeeze the irrational number. In the end, $\phi^p, p \in \mathbb{R}_+$ is also an eigenfunction with the corresponding eigenvalue λ_1^p. Note that constant function $\varphi(x) := c \in \mathbb{C}$ is also an eigenfunction with eigenvalue 1 since

$$[U\varphi](x) = \varphi(T(x)) = c = \varphi(x).$$

Furthermore if φ_1 vanishes nowhere, $\varphi_1^{-1} = \frac{1}{\varphi_1}$ is well defined and therefore one can extend $p \in \mathbb{R}$ since for $p_0 \in \mathbb{R}_+$, a function $\varphi_1^{-p_0}$ is an eigenfunction with eigenvalue $\lambda_1^{-p_0}$ by

$$[U\varphi_1^{-1}](x) = \frac{1}{\varphi_1(T(x))} = \frac{1}{\lambda_1\varphi_1(x)} = \lambda_1^{-1}\varphi_1^{-1}(x).$$

□

Under measure-preserving systems, we have more useful properties. In general, a Koopman operator does not necessarily commute with its adjoint. However, if $\mathcal{F} = L^2(\mathcal{M}, \mu)$, then a Koopman operator $U : \mathcal{F} \to \mathcal{F}$ of a measure-preserving dynamical system has a unitary extension, denote as U', defined on an extended Hilbert space $\mathcal{H}$ with $\mathcal{F} \subset \mathcal{H}$ [SN+10]. Moreover, if the evolution function T is invertible and measure-preserving, which we assume in this thesis, then $U' = U$ and $\mathcal{H} = \mathcal{F}$, i.e., the Koopman operator U is unitary and U^{-1} is the adjoint operator of U [Col23]. Since the operator U is unitary, the spectrum of U, denote it as $\sigma(U)$, satisfies

$$\sigma(U) \subseteq \{z \in \mathbb{C} \mid |z| = 1\} = \mathbb{T},$$

and the spectral theorem [Bor20] ensures that the existence of a projection-valued spectral measure E supported on $\sigma(U)$ such that

$$U = \int_{\mathbb{T}} z dE(z).$$

Now, remember that the spectrum can be decomposed into three disjoint parts: point spectrum, continuous spectrum, and residual spectrum, and since U is unitary, especially normal, the residual part of the spectrum is empty. Thus, E can be decomposed into two measures, E_p and E_c, that are supported on the point spectrum and the continuous part of the spectrum, respectively,

$$E = E_\mathrm{p} + E_\mathrm{c}.$$

Therefore, for any $f \in L_2(M, \mu)$, the spectral resolution becomes

$$U^k f = \sum_j e^{i2\pi\omega_j k} P_j f + \left(\int_0^1 e^{i2\pi\theta k} dE_c(\theta)\right) f$$

where $P_j : L_2(M, \mu) \to L_2(M, \mu)$ is the orthogonal projection onto the eigenspace corresponding to the eigenvalue $\lambda_j = e^{i2\pi\omega_j}$ and E_c is the projection valued measure corresponding to the continuous part of the spectrum. In some later-introduced DMD-type algorithms, such as DMD [MB12] and EDMD [WKR15], they only consider the point spectrum part by assuming there does not exist a continuous part. In this case, by definition of the Koopman mode, $P_j f(p) = \phi_j(p) C_j(f)$, and get

$$\sum_j e^{i2\pi\omega_j k} P_j f = \sum_j e^{i2\pi\omega_j k} \phi_j C_j(f).$$

Note that this is the same as saying $\mathcal{F}$ is spanned by the eigenfunctions of the Koopman operator U. The continuous part of the spectrum is not understood at the same level as the point spectrum part. However, the algorithms, such as Hankel DMD [AM17; KPM20] and mpEDMD [Col23], which assume the ergodic or measure-preserve systems, showed the relationship with the continuous part.

Ergodic theory Ergodic theory is a branch of mathematics that deals with the statistical properties of dynamic systems and their long-term behavior. Intuitively, ergodic theory considers the case where the system explores its entire phase space uniformly over time. Mathematically, the theory is developed in the context of measure-preserving transformations in measure theory. As it is explained in the introduction section, it has applications in various fields, including physics, probability theory, and information theory. We introduce the results from the ergodic theory based on Cornfeld, Fomin, and Sinai [CFS12].

A measure-preserving function g is called *invariant* (with respect to T) if for all $x \in \mathcal{M}$,

$$g(T(x)) = g(x) = g(T^{-1}(x)).$$

If this is true for almost everywhere instead of for all $x \in \mathcal{M}$, i.e., if it is true for $x \in \mathcal{M} \setminus \{B \in \mathfrak{B} \mid \mu(B) = 0\}$, then it is said to be *invariant* mod 0. Analogously, a set $A \in \mathfrak{B}$ is called *invariant*, or *invariant* mod 0, if the indicator function

$$\chi_A(x) := \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is an invariant function or invariant mod 0 function, respectively.

A dynamical system $(\mathcal{M}, \mathfrak{B}, \mu, T)$ is said to be *ergodic* if the measure $\mu(A)$ of any invariant set A equals 0 or 1. There are several equivalent conditions of ergodicity, and we use the following definition (Definition 2.1). Furthermore, under the ergodic settings, the well-known Birkhoff-Khinchin's ergodic theorem (Theorem 2.1) can be shown.

Definition 2.1. *(Equivalent definition of ergodic system) A dynamical system $(\mathcal{M}, \mathfrak{B}, \mu, T)$ is said to be ergodic if any function $f \in L^2(\mathcal{M}, \mathfrak{B}, \mu)$ invariant with respect to the Koopman operator U_T, is a constant almost everywhere.*

Theorem 2.1. *(The Birkhoff-Khinchin Ergodic Theorem) Suppose $(\mathcal{M}, \mathfrak{B}, \mu)$ is a space with normalized measure and $f \in L^1((\mathcal{M}, \mathfrak{B}, \mu))$. Then, for almost every (in the sense of the measure μ) $x \in \mathcal{M}$, the following limit exists and holds*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} f(T^{k-1}x) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} [U^{k-1} f](x) := \bar{f}(x).$$

Furthermore,

$$\bar{f}(x) \in L^1(\mathcal{M}, \mathfrak{B}, \mu) \text{ and } \int_{\mathcal{M}} \bar{f}(x) d\mu = \int_{\mathcal{M}} f d\mu = \frac{1}{\mu(\mathcal{M})} \int_{\mathcal{M}} f d\mu$$

Proof. Check Cornfeld, Fomin, and Sinai [CFS12]. □

In the Birkhoff-Khinchin Ergodic Theorem, the limits $\lim_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} f(T^{k-1}x)$ represents the time average and the integral $\frac{1}{\mu(\mathcal{M})} \int_{\mathcal{M}} f d\mu$ represents the space average. Since any invariant function is constant on any set of full measures when the dynamical system is ergodic, $\bar{f}$ is constant almost everywhere. Therefore, one can state the ergodic theorem as follows:

For almost every point $x \in \mathcal{M}$, the time means equals to the space means.

Lastly, under the ergodic setting, one can further show that every eigenvalue of the Koopman operator U_T is of multiplicity 1, and the absolute value of every eigenfunction of U_T is constant almost everywhere. For detailed proof, please refer to chapters 1 and 12 of the Cornfeld, Fomin, and Sinai [CFS12].

2.2 Dynamic Mode Decomposition (DMD) and its variants

Now, we introduce the data-driven algorithms to approximate eigenvalues, eigenfunctions, and Koopman modes of dynamical systems.

Dynamic Mode Decomposition (DMD) Based on the Koopman mode analysis, the data-driven method Dynamic Mode Decomposition (DMD) was originally introduced by Schmid and Sesterhenn [SCH10] in the fluid mechanics community as a powerful tool for analyzing the dynamics of nonlinear systems. The DMD can be seen as a variant of a standard Arnoldi method [ROW+09], and it approximates the Koopman operator by considering the projection onto the finite-dimensional space and a matrix representation of them. The matrix representation is calculated as the companion matrix, where the last columns are the coefficients of the least-squares approximation of the last data points. Even though this does not approximate eigenfunctions, it is still powerful tool for understanding the underlying dynamics by data.

In detail, for fixed $f_i \in \mathcal{F}$ i.e. $f_i : \mathcal{M} \to \mathbb{C}$, and vector-valued observables $F : \mathcal{M} \to \mathbb{C}^m$, $F = (f_1, \cdots, f_m)^T$, consider the cyclic subspace $\mathcal{K}_\infty$ and the Krylov subspace $\mathcal{K}_r$ with $r < \infty$,

$$\mathcal{K}_\infty = \text{span}\{U^k F\}_{k=0}^{\infty}, \mathcal{K}_r = \text{span}\{U^k F\}_{k=0}^{r-1}.$$

Suppose $\{U^k F\}_{k=0}^{r-1}$ are linearly independent so that these functions form a basis for $\mathcal{K}_r$. For fixed initial point $p_0 \in \mathcal{M}$, assume input data is the $\left\{U^k F\right\}_{k=0}^{r}$ which is the trace observations through the vector-valued observables F. This means that the input trace data X is

$$X = \{x_0, x_1, \cdots, x_{r-1}\} = \{F(p_0), F(T(p_0)), \cdots, F(T^r(p_0))\} \in \mathbb{C}^{m\times(r+1)}.$$

Here, consider approximating the projection $P_r : \mathcal{F}^m \to \mathcal{K}_r$ as the minimizer of the $\mathbb{C}^m$ norm of the residual η_r defined as

$$\eta_r := x_r - P_r(x_r) \in \mathbb{C}^m.$$

Since $P_r(x_r) \in \mathcal{K}_r$, one can represent $P_r(x_r)$ as a linear combination of elements of $\mathcal{K}_r$, that is, $P_r(x_r) = \sum_{k=0}^{r-1} c_k x_k$ with some coefficients $c_k \in \mathbb{C}$. Thus, finding the projection P_r is equivalent to finding the coefficients $c = (c_0, \cdots, c_{r-1})^T \in \mathbb{C}^r$ such that

$$\begin{aligned} c &= \underset{c'\in\mathbb{C}^r}{\arg\min} \left\| x_r - \sum_{k=0}^{r-1} c_k x_k \right\|_{\mathbb{C}^m} = \underset{c'\in\mathbb{C}^r}{\arg\min} \|x_r - P_r(x_r)\|_{\mathbb{C}^m} = \underset{c'\in\mathbb{C}^r}{\arg\min} \|\eta_r\|_{\mathbb{C}^m} \\ &= \underset{c'\in\mathbb{C}^r}{\arg\min} \|[U^r F](p_0) - P_r[U^r F](p_0)\|_{\mathbb{C}^m}. \end{aligned}$$

For the eigenvalues and Koopman mode on the space $\mathcal{K}_r$, since $P_r U|_{\mathcal{K}_r} : \mathcal{K}_r \to \mathcal{K}_r$ is a finite-dimensional linear operator, it has a matrix representation $A_r : \mathbb{C}^r \to \mathbb{C}^r$ in the $\{U^k F\}_{k=0}^{r-1}$ basis. Write

$$K_r = \begin{pmatrix} x_0 & x_1 & \cdots & x_{r-1} \end{pmatrix} \in \mathbb{C}^{m\times r},$$

then $x_r = K_r c + \eta_r \in \mathbb{C}^m$. Since $U x_k = x_{k+1}$,

$$UK_r = \begin{pmatrix} Ux_0 & Ux_1 & \dots & Ux_{r-1} \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \dots & x_r \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & \dots & K_r c + \eta_r \end{pmatrix} = K_r A_r + \eta_r e_r^T,$$

where

$$e_r = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \in \mathbb{C}^m, A_r = \begin{pmatrix} 0 & 0 & \cdots & 0 & c_0 \\ 1 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & \cdots & 0 & c_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{r-1} \end{pmatrix} \in \mathbb{C}^{r\times r}.$$

Since A_r is a matrix, consider eigenvalue decomposition $A_r = V^{-1}\Lambda V$, then

$$UK_r = K_r V^{-1}\Lambda V + \eta_r e_r^T \implies UK_r V^{-1} = K_r V^{-1}\Lambda + \eta_r e_r^T V^{-1}.$$

Define $E := K_r V^{-1}$, then

$$UE = E\Lambda + \eta_r e_r^T V^{-1}.$$

Since we are minimizing $\|\eta_r\|_{\mathbb{C}^m}$, it is natural to suppose for large r, $\left\|\eta_r e_r^T V^{-1}\right\|$ is small. Then, $UE \approx E\Lambda$, the columns of E approximate some eigenvectors of U, and the diagonal elements of Λ approximate some eigenvalues of U. This means, for the i-th columns of E, w_i, and the i-th diagonal element of Λ, λ_i, w_i approximates $\phi_i(p)C_i(F)$ and λ_i approximates the corresponding eigenvalue of U. Even though $\{w_i\}$ are not exactly the i-th Koopman mode $C_i(F)$, they are considered as the Koopman mode approximation generated by DMD. A summary of Dynamic Mode Decomposition (DMD) is given in algorithm 1.

Algorithm 1 Dynamic Mode Decomposition (DMD) by Schmid and Sesterhenn [SCH10]

Input: $X = (x_0, x_1, \cdots, x_r)$ where $x_k = U^k F(p_0) \in \mathbb{C}^m$ with some initial point $p_0 \in \mathcal{M}$.
Output: (E, Λ) where $E \in \mathbb{C}^{m\times r}$ are the Koopman modes and $\Lambda \in \mathbb{C}^{r\times r}$ are the eigenvalues.

1: $K_r \leftarrow \begin{pmatrix} x_0 & x_1 & \cdots & x_{r-1} \end{pmatrix} \in \mathbb{C}^{m\times r}$.
2: $c \leftarrow \arg\min_{c'\in\mathbb{C}^r} \left\|x_r - \sum_{k=0}^{r-1} c'_k x_k\right\|$
3: $A_r \leftarrow \begin{pmatrix} 0 & 0 & \cdots & 0 & c_0 \\ 1 & 0 & \cdots & 0 & c_1 \\ 0 & 1 & \cdots & 0 & c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_{r-1} \end{pmatrix} \in \mathbb{C}^{r\times r}$
4: $A_r = V^{-1}\Lambda V$ ▷ eigenvalue decomposition
5: $E \leftarrow K_r V^{-1} \in \mathbb{C}^{m\times r}$

The introduced DMD algorithm is comprehensible but is numerically unstable if the inputs are ill-conditioned [SCH10]. Therefore, DMD is often implemented with the slite modification by using Singular Value Decomposition (SVD) to make the algorithm numerically stable, and this approach is called SVD-enhanced DMD [AM17; CTR12]. For calculating matrix representation A_r, by defining for $X = (x_0, x_1, \cdots, x_{r-1})$ and $Y = (x_1, x_2, \cdots, x_r)$, A_r to be the minimizer of the $\|A_r X - Y\|$. Thus, by using SVD, one can obtain A_r by

$$A_r = U^* Y V S^{-1} \text{ where } X = USV^*,$$

and the dynamic mode is $v_j = Uw_j$.

Hankel DMD The Hankel DMD is a DMD-type algorithm that was introduced by Arbabi and Mezić [AM17] and is used when the dynamical system is ergodic. The key idea of the Hankel DMD is the dynamic eigenvalues and dynamic modes obtained by applying DMD to the Hankel matrix converge to the eigenvalues and eigenfunctions of the Koopman operator U. Also, it combines the idea of SVD-enhanced DMD mentioned above.

Assume $\mathcal{F}$ is the Hilbert space $L^2(\mathcal{M}, \mu)$. For observables $f, g \in \mathcal{F}$, denote their trace data from point $z_0 \in \mathcal{M}$ as

$$\tilde{f}_m(z_0) = \begin{pmatrix} f(z_0) & f\circ T(z_0) & \cdots & f\circ T^{m-1}(z_0) \end{pmatrix}^T \in \mathbb{C}^m$$
$$\tilde{g}_m(z_0) = \begin{pmatrix} g(z_0) & g\circ T(z_0) & \cdots & g\circ T^{m-1}(z_0) \end{pmatrix}^T \in \mathbb{C}^m.$$

Then, by ergodicity and Birkhoff's theorem (Theorem 2.1), for almost every $z_0 \in M$,

$$\lim_{m\to\infty} \frac{1}{m}\langle \tilde{f}_m(z_0), \tilde{g}_m(z_0)\rangle = \lim_{m\to\infty} \frac{1}{m}\sum_{k=0}^{m-1} f(T^k(z_0))\overline{g(T^k(z_0))} = \lim_{m\to\infty} \frac{1}{m}\sum_{k=0}^{m-1} (fg^*)(T^k(z_0)) = \int_{\mathcal{M}} fg^* d\mu$$
$$= \langle f, g\rangle_{\mathcal{F}},$$

where $\langle\cdot,\cdot\rangle$ is the vector inner product and $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ is the functional inner product of the Hilbert space $\mathcal{F}$. This means, the vector inner product converges to the functional inner product as the number of the sampled sequence m goes to infinity.

Now, consider the Hankel matrix $\tilde{H}$ defined as

$$\tilde{H} := \begin{pmatrix} f(z_0) & Uf(z_0) & \cdots & U^n f(z_0) \\ Uf(z_0) & U^2 f(z_0) & \cdots & U^{n+1} f(z_0) \\ \vdots & \vdots & \vdots & \vdots \\ U^{m-1} f(z_0) & U^m f(z_0) & \cdots & U^{m+n-1} f(z_0) \end{pmatrix} = \begin{pmatrix} \tilde{f}_m(z_0) & U\tilde{f}_m(z_0) & \cdots & U^n \tilde{f}_m(z_0) \end{pmatrix},$$

where

$$U^k \tilde{f}_m(z_0) := \begin{pmatrix} f \circ T^k(z_0) & f \circ T^{k+1}(z_0) & \cdots & f \circ T^{k+m-1}(z_0) \end{pmatrix}^T.$$

In the same way as the DMD, define Krylov subspace along an observable f as $\mathcal{F}_n := [f, Uf, \cdots, U^n f]$, and let k be the dimension of the minimal Koopman-invariant subspace, denoted by $\mathcal{F}$, which contains f. Then the first k iterates of f under the action of the Koopman operator span $\mathcal{F}$, i.e.,

$$\mathcal{F} = \text{span}\mathcal{F}_n. \quad (\forall n \geq k-1)$$

Now apply DMD to the first $k+1$ columns of $\tilde{H}$. Since $\mathcal{F}$ is a Hilbert space and the last column c of the matrix representation was obtained by the coefficients of the projection onto $\mathcal{F}$,

$$c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{k-1} \end{pmatrix} = G^{-1} \begin{pmatrix} \langle f, U^k f \rangle_{\mathcal{F}} \\ \langle Uf, U^k f \rangle_{\mathcal{F}} \\ \vdots \\ \langle U^{k-1} f, U^k f \rangle_{\mathcal{F}} \end{pmatrix},$$

where G is the Gramian matrix of the basis given by $G_{ij} = \langle U^{i-1} f, U^{j-1} f \rangle_{\mathcal{F}}$. Since vector inner product convergeced to functional inner product, i.e., $\lim_{m\to\infty} \frac{1}{m} \langle \tilde{f}_m(z_0), \tilde{g}_m(z_0) \rangle = \langle f, g \rangle_{\mathcal{F}}$, consider numerical calculation of c denote as $\tilde{c}$, which is given by

$$\tilde{c} = \begin{pmatrix} \tilde{c_1} \\ \tilde{c_2} \\ \vdots \\ \tilde{c_{k-1}} \end{pmatrix} = \tilde{G}^{-1} \begin{pmatrix} \frac{1}{m} \langle \tilde{f}_m, U^k \tilde{f}_m \rangle \\ \frac{1}{m} \langle U\tilde{f}_m, U^k \tilde{f}_m \rangle \\ \vdots \\ \frac{1}{m} \langle U^{k-1} \tilde{f}_m, U^k \tilde{f}_m \rangle \end{pmatrix}, \tilde{G}_{ij} = \frac{1}{m} \langle U^{i-1} \tilde{f}_m, U^{j-1} \tilde{f}_m \rangle.$$

As it is shown, the averaged inner product converges to the Hilbert space inner product i.e.,

$$\frac{1}{m} \langle \tilde{f}_m(z_0), \tilde{g}_m(z_0) \rangle \to \langle f, g \rangle_{\mathcal{F}}, \ \tilde{G}_{ij} \to G_{ij}, \ \lim_{m\to\infty} \tilde{G}^{-1} = \left(\lim_{m\to\infty} \tilde{G} \right)^{-1} = G^{-1}.$$

Thus the last column c of the matrix representation given by DMD can be calculated as the limit of $\tilde{c}$. To summarize, the Hankel DMD is shown as algorithm 2.

Algorithm 2 Hankel Dynamic Mode Decompotision by Arbabi and Mezić [AM17]

Input: $(f_i(z_i), Uf_i(z_i), \cdots, U^{m+n-1} f_i(z_i))$ for $i \in \{1, 2, \ldots, l\}$.
Output: (Λ, Φ) eigenvalues and eigenfunctions

1: $H^i \leftarrow \begin{pmatrix} f_i(z_i) & Uf_i(z_i) & \cdots & U^n f_i(z_i) \\ Uf_i(z_i) & U^2 f_i(z_i) & \cdots & U^{n+1} f_i(z_i) \\ \vdots & \vdots & \vdots & \vdots \\ U^{m-1} f_i(z_i) & U^m f_i(z_i) & \cdots & U^{m+n-1} f_i(z_i) \end{pmatrix}$
2: $\alpha_i \leftarrow \frac{\|H^i_{n+1}\|}{\|H^1_{n+1}\|}$ where H^i_{n+1} is n+1-th columns of H^i
3: $X \leftarrow (H_1, \alpha_2 H_2, \cdots, \alpha_l H_l)$
4: $Y \leftarrow (UH_1, \alpha_2 UH_2, \cdots, \alpha_l UH_l)$
5: perform SVD; $X = WS\tilde{V}^*$
6: $\tilde{A} \leftarrow W^* Y \tilde{V} S^{-1}$
7: perform eigenvalue decomposition; $\tilde{A} = \Xi^{-1} \Lambda \Xi$
8: $\Phi \leftarrow W\Xi$

Extended Dynamic Mode Decomposition (EDMD) As introduced, the DMD or Hankel DMD returns only two terms: dynamic eigenvalues and dynamic modes. This means that they cannot capture the desired three terms: eigenvalues, eigenfunctions, and Koopman modes. The Extended Dynamic Mode Decomposition (EDMD), which was introduced by Williams, Kevrekidis, and Rowley [WKR15], addresses this issue by considering state space observables and approximates not only the eigenvalues and Koopman modes but also the eigenfunctions of Koopman operator. Note that the Koopman modes correspond to the state space observable function.

For $\mathcal{D} = \{\phi_1, \phi_2, \cdots, \phi_K\}$, $\phi_i \in \mathcal{F}$, define, in the same way as the DMD and Hankel-DMD, that $\mathcal{F}_\mathcal{D} := \text{span}\{\phi_i\}_{i=1}^K$, and a vector-valued function Φ as

$$\Phi : M \to \mathbb{C}^{1\times K}, \Phi(x) := \begin{pmatrix}\phi_1(x) & \phi_2(x) & \cdots & \phi_K(x)\end{pmatrix}.$$

Then, any function $f \in \mathcal{F}_\mathcal{D}$ can be written in the form of linear combinations $\{\phi_i\}_{i=1}^K$, i.e.,

$$f = \sum_{i=1}^K a_i\phi_i = \Phi a,$$

where $a = (a_1, \cdots, a_K)^T \in \mathbb{C}^K$. Thus,

$$Uf = U(\Phi a) = (\Phi \circ T)a.$$

Note $\Phi \circ T : M \to \mathbb{C}^{1\times K}$, $[\Phi \circ T](x) = \Phi(T(x)) = \begin{pmatrix}\phi_1(T(x)) & \phi_2(T(x)) & \cdots & \phi_K(T(x))\end{pmatrix}$. Now, consider a finite-dimensional approximation of U, denote it as $K_{EDMD} \in \mathbb{C}^{K\times K}$. for some residual $\eta \in \mathcal{F}$, we have

$$Uf = \Phi(K_{EDMD}a) + \eta.$$

As we did in the DMD, K_{EDMD} is defined as a minimizer of the residual term η_m with L_2 loss for each data (x_m, y_m), i.e.

$$K_{EDMD} := \arg\min_K \frac{1}{2}\sum_{m=1}^M |\eta_m|^2 = \arg\min_K \frac{1}{2}\sum_{m=1}^M |[Uf - \Phi(Ka)](x_m)|^2.$$

Here, for the objective $J := \frac{1}{2}\sum_{m=1}^M |\eta_m|^2$,

$$\begin{aligned} J &= \frac{1}{2}\sum_{m=1}^M |\eta_m|^2 = \frac{1}{2}\sum_{m=1}^M |[Uf - \Phi(Ka)](x_m)|^2 = \frac{1}{2}\sum_{m=1}^M |[\Phi \circ T](x_m)a - \Phi(x_m)(Ka)|^2 \\ &= \frac{1}{2}\sum_{m=1}^M |(\Phi(T(x_m)) - \Phi(x_m)K)\, a|^2 = \frac{1}{2}\sum_{m=1}^M |(\Phi(y_m) - \Phi(x_m)K)\, a|^2. \end{aligned}$$

Since J is convex, it has either a unique global minimum or a continuous family of minimizers, and the closed form of the solution is well-known as

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

where † denotes the pseudo-inverse and $G, A \in \mathbb{C}^{K\times K}$ are defined as

$$G = \frac{1}{M}\sum_{m=1}^M \Phi(x_m)^*\Phi(x_m), A = \frac{1}{M}\sum_{m=1}^M \Phi(y_m)^*\Phi(x_m).$$

Note that G, A can be calculated by $\Phi, \{x_m\}_{m=1}^M, \{y_m\}_{m=1}^M$, i.e., by snapshots of states $\{x_m\}_{m=1}^M, \{y_m\}_{m=1}^M$ with $y_m = T(x_m)$ and dictionaries of observables $\mathcal{D}$ which spans the function space $\mathcal{F}$ of the observable functions $f \in \mathcal{F}$.

For approximating eigenvalues and eigenfunctions of U, the eigenvalues and eigenvectors of K_{EDMD} can be used. Let λ_i, ξ_i be i-th eigenvalue and eigenvector of K_{EDMD}, respectively, then define

$$\varphi_i := \Phi\xi_i.$$

With this definition, λ_i and φ_i are approximations of an eigenvalue and the corresponding eigenfunction of U, respectively, because

$$U\varphi_i = (\Phi \circ T)\xi_i \approx \Phi(K_{EDMD}\xi_i) = \Phi(\lambda_i\xi_i) = \lambda_i(\Phi\xi_i) = \lambda_i\varphi_i.$$

For the Koopman modes, assume $\mathcal{M} \subset \mathbb{R}^N$ and define the full state observables, $g : M \to \mathbb{R}^N$, $g(x) = x$, i.e.,

$$g(x) = \begin{pmatrix} g_1(x) & \dots & g_N(x) \end{pmatrix}^T = \begin{pmatrix} e_1^* x & \dots & e_N^* x \end{pmatrix}^T,$$

and consider approximating Koopman modes of this function. Note that if g is spanned by the eigenfunctions $\{\varphi_k\}_{k=1}^K$ with coefficients $\{v_k\}_{k=1}^K$ where $v_k \in \mathbb{R}^N$, then $\{v_k\}_{k=1}^K$ are the Koopman modes with respect to the function g and for the evolution function $T : \mathcal{M} \to \mathcal{M}$ can be written as

$$T(x) = T(g(x)) = g \circ T(x) = [Ug](x) = \sum_{k=1}^{K} v_k [U\varphi_k](x) = \sum_{k=1}^{K} \lambda_k v_k \varphi_k(x),$$

i.e., the underlying evolution function T is defined only by eigenfunctions, eigenvalues, and the Koopman modes with respect to the full state observables g. For notational convenience, define $\Psi : \mathcal{M} \to \mathbb{C}^{1\times K}$ such that

$$\Psi(x) := \begin{pmatrix} \varphi_1(x) & \varphi_2(x) & \cdots & \varphi_K(x) \end{pmatrix},$$

and denote the corresponding Koopman modes as $V_g = (v_1, v_2, \cdots, v_K) \in \mathbb{R}^{N\times K}$. Then, by the definition of the Koopman modes,

$$g = \sum_{k=1}^{K} v_k \varphi_k = V_g \Psi^T.$$

Further assume that $g_i \in \mathcal{F}_{\mathcal{D}}$ so that they can be written as the linear combinations of the each element of the vector-valued observables, namely, $g_i = \sum_{k=1}^{K} \phi_k b_{k,i} = \Phi b_i,\ b_i \in \mathbb{C}^K$. Also, define $B = (b_1, b_2, \cdots, b_K) \in \mathbb{C}^{K\times N}$ and $\Xi = (\xi_1, \xi_2, \cdots, \xi_K) \in \mathbb{C}^{K\times K}$. Then, since $\Psi = \Phi\Xi \implies \Phi = \Psi\Xi^{-1}$,

$$g = (\Phi B)^T = B^T \Xi^{-T} \Psi^T,$$

By combining the results,

$$V_g = B^T \Xi^{-T}.$$

Note that for obtaining B, one can use the equation $x = g(x) = B^T\Phi(x)^T$. For the input data $X = \begin{pmatrix} x_1 & x_2 & \cdots & x_M \end{pmatrix} \in \mathbb{R}^{N\times M}$,

$$X = B^T \Phi(X)^T \in \mathbb{C}^{N\times M}.$$

Note that the Koopman modes computed using EDMD are equivalent to DMD modes if the dictionary of observables $\mathcal{D}$ used in EDMD is the scalar observables of the form $\phi_i(x) = e_i^* x$ for $i = 1, \dots, N$. For details, please refer to Williams, Kevrekidis, and Rowley [WKR15]. Lastly, a summary of Extended Dynamic Mode Decomposition (EDMD) is given as algorithm 3.

Algorithm 3 Extended Dynamic Mode Decomposition by Williams, Kevrekidis, and Rowley [WKR15]

Input: $(X, Y, \mathcal{D})$ where $X = \begin{pmatrix} x_1 & x_2 & \cdots & x_M \end{pmatrix}$, $Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_M \end{pmatrix}, y_k = T(x_k)$, and $\mathcal{D} = \{\phi_1, \phi_2, \cdots, \phi_K\}, \phi_k \in \mathcal{F}$

Output: (K, Λ, V_g)

1: $\Phi(x) := \begin{pmatrix} \phi_1(x) & \phi_2(x) & \cdots & \phi_K(x) \end{pmatrix}$
2: $G \leftarrow \frac{1}{M} \sum_{m=1}^{M} \Phi(x_m)^* \Phi(x_m)$
3: $A \leftarrow \frac{1}{M} \sum_{m=1}^{M} \Phi(x_m)^* \Phi(y_m)$
4: $K \leftarrow G^\dagger A$
5: $\Lambda, \Xi \leftarrow$ eigendecomposition(K) such that $K = \Xi^{-1}\Lambda\Xi$
6: $B \leftarrow \arg\min_B \left\| B^T\Phi(X)^T - X \right\|$
7: $V_g \leftarrow (\Xi^{-1}B)^T$

measure-preserving extended dynamic mode decomposition (mpEDMD) As mentioned in [Col23], spectral pollution is a major concern of the EDMD because, for a large number of data snapshot M, the eigenvalues computed by EDMD corresponds to the finite selection method [BS83], which can suffer from spectral pollution. This is problematic, especially if the dynamical system is measure-preserving, such as Hamiltonian flows [Arn89], geodesic flows [DFN85], and ergodic systems [CFS12]. Colbrook introduced the measure-preserving extended dynamic mode decomposition (mpEDMD) to overcome the issue [Col23].

Recall, in EDMD, one chooses the matrix representation K_{EDMD} as a solution of

$$\underset{B\in\mathbb{C}^{K\times K}}{\arg\min}\left(\int_{\Omega}\|\Phi(T(x))-\Phi(x)B\|_2^2\,d\omega(x)\right),$$

and it is numerically calculated by the input data $(X, Y, \mathcal{D})$ where

$$X=\begin{pmatrix}x_1 & x_2 & \cdots & x_M\end{pmatrix}, Y=\begin{pmatrix}y_1 & y_2 & \cdots & y_M\end{pmatrix}, y_k=T(x_k), \mathcal{D}=\{\phi_1,\phi_2,\cdots,\phi_K\}, \phi_k\in\mathcal{F},$$

and by quadrature rule

$$K_{EDMD}\in\underset{B\in\mathbb{C}^{K\times K}}{\arg\min}\left(\sum_{m=1}^{M}w_m\|\Phi(y_m)-\Phi(x_m)B\|_2^2\right).$$

Note that in the original EDMD, w_m are defined to be the equal weight, i.e. $w_m=\frac{1}{M}$ for all $m\in\{1,2,\ldots,M\}$. The solution is given by $K_{EDMD}=G^{\dagger}A$ where $G=\frac{1}{M}\sum_{m=1}^{M}\Phi(x_m)^*\Phi(x_m), A=\frac{1}{M}\sum_{m=1}^{M}\Phi(x_m)^*\Phi(y_m)$, with $\Phi(x):=\begin{pmatrix}\phi_1(x) & \phi_2(x) & \cdots & \phi_K(x)\end{pmatrix}$. By definition, $(G)_{k,l}=\frac{1}{M}\sum_{m=1}^{M}\overline{\phi_k(x_m)}\phi_l(x_m)$. Therefore, if quadrature approximation converges, then similar to the Hankel-DMD case,

$$\lim_{M\to\infty}(G)_{k,l}=\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\overline{\phi_k(x_m)}\phi_l(x_m)=\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\langle\phi_l(x_m),\phi_k(x_m)\rangle=\langle\phi_l,\phi_k\rangle_{\mathcal{F}}.$$

Analogously, $\lim_{M\to\infty}(A)_{k,l}=\langle U\phi_l,\phi_k\rangle_{\mathcal{F}}$. In this case, for any function $f\in\mathcal{F}$, with $\|f\|=1$, its EDMD approximation is given by $f'\in\mathcal{F}_{\mathcal{D}}$ with $f'=\sum_{i=1}^{K}a_i\phi_i=\Phi a$ where $a=(a_1,\cdots,a_K)^T\in\mathbb{C}^K$ and $\|f'\|=1$. Now, for the matrix representation of the Koopman operator $B\in\mathbb{C}^{K\times K}$, $\|f'\|^2=\|\Phi a\|^2\approx a^*Ga=\left\|G^{\frac{1}{2}}a\right\|^2$ and $\|\Phi Ba\|\approx a^*B^*GBa$. Since B is an isometry on the space $L^2(\mathcal{M},\mu)$, mpEDMD considers the matrix representation K_{mpEDMD} which satisfies

$$a^*Ga=a^*K_{mpEDMD}^*GK_{mpEDMD}a,\forall a\in\mathbb{C}^K\Leftrightarrow K_{mpEDMD}^*GK_{mpEDMD}=G.$$

Thus, the K_{mpEDMD} is defined as the solution of

$$\underset{B\in\mathbb{C}^{K\times K},B^*GB=G}{\arg\min}\left(\int_{\Omega}\left\|\Phi(T(x))G^{-\frac{1}{2}}-\Phi(x)BG^{-\frac{1}{2}}\right\|_2^2d\omega(x)\right),$$

and by the quadrature rule,

$$K_{mpEDMD}\in\underset{B\in\mathbb{C}^{K\times K},B^*GB=G}{\arg\min}\left(\sum_{m=1}^{M}w_m\left\|\Phi(y_m)G^{-\frac{1}{2}}-\Phi(x_m)BG^{-\frac{1}{2}}\right\|_2^2\right).$$

By letting $B=G^{-\frac{1}{2}}CG^{\frac{1}{2}}$ for some $C\in\mathbb{C}^{K\times K}$, $K_{mpEDMD}=G^{-\frac{1}{2}}CG^{\frac{1}{2}}$ with

$$C\in\underset{C\in\mathbb{C}^{n\times n},C^*C=I}{\arg\min}\left\|W^{\frac{1}{2}}\Phi(Y)G^{-\frac{1}{2}}-W^{\frac{1}{2}}\Phi(X)G^{-\frac{1}{2}}C\right\|_F^2,$$

where

$$W=\begin{pmatrix}w_1 & 0 & \ldots & 0\\ 0 & w_2 & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & w_M\end{pmatrix}\in\mathbb{C}^{M\times M},\Phi(X)=\begin{pmatrix}\Phi(x_1)\\ \Phi(x_2)\\ \vdots\\ \Phi(x_M)\end{pmatrix}\in\mathbb{C}^{M\times K},\Phi(Y)=\begin{pmatrix}\Phi(y_1)\\ \Phi(y_2)\\ \vdots\\ \Phi(y_M)\end{pmatrix}\in\mathbb{C}^{M\times K}.$$

This problem is known as the orthogonal Procrustes problem [Sch66]. Recall for

$$R = \underset{Q,Q^*Q=I}{\arg\min} \|QA - B\|_F^2,$$

the solution is given by $R = UV^*$ with the singular value decomposition $BA^* = U\Sigma V^*$. Noting that $\|A\|_F^2 = \|A^*\|_F^2$, $C \in \arg\min_{C\in\mathbb{C}^{K\times K}, C^*C=I} \left\| C^* \left(W^{\frac{1}{2}}\Phi(x_m)G^{-\frac{1}{2}} \right)^* - \left(W^{\frac{1}{2}}\Phi(y_m)G^{-\frac{1}{2}} \right)^* \right\|_F^2$. Thus, $C^* = U_1U_2^*$ with singular value decomposition

$$U_1\Sigma U_2^* = \left(W^{\frac{1}{2}}\Phi(y_m)G^{-\frac{1}{2}} \right)^* \left(W^{\frac{1}{2}}\Phi(x_m)G^{-\frac{1}{2}} \right) = G^{-\frac{1}{2}}\Phi(y_m)^*W\Phi(x_m)G^{-\frac{1}{2}} = G^{-\frac{1}{2}}A^*G^{-\frac{1}{2}}.$$

Note that since K_{mpEDMD} is unitary similar to a unitary matrix, it is also a unitary matrix, and thus its eigenvalues have absolute value 1. To summarize, we get the algorithm 4, measure-preserving Extended Dynamic Mode Decomposition.

Algorithm 4 measure-preserving Extended Dynamic Mode Decomposition by Colbrook [Col23]

Input: $(X, Y, \mathcal{D})$ where $X = \begin{pmatrix} x_1 & x_2 & \cdots & x_M \end{pmatrix}$, $Y = \begin{pmatrix} y_1 & y_2 & \cdots & y_M \end{pmatrix}, y_k = T(x_k)$, and $\mathcal{D} = \{\phi_1, \phi_2, \cdots, \phi_K\}, \phi_k \in \mathcal{F}$.

Output: (K, Λ, V_g)

1: $\Phi(x) := \begin{pmatrix} \phi_1(x) & \phi_2(x) & \cdots & \phi_K(x) \end{pmatrix}$
2: $G \leftarrow \frac{1}{M}\sum_{m=1}^{M} \Phi(x_m)^*\Phi(x_m)$
3: $A \leftarrow \frac{1}{M}\sum_{m=1}^{M} \Phi(x_m)^*\Phi(y_m)$
4: $U_1, S, U_2 \leftarrow \mathrm{svd}(G^{-1/2}A^*G^{-1/2})$ i.e. $U_1SU_2^* = G^{-1/2}A^*G^{-1/2}$ ▷ Singular Value Decomposition
5: $\bar{V}, \Lambda \leftarrow \mathrm{eig}(U_2U_1^*)$ i.e. $U_2U_1^* = \bar{V}\Lambda\bar{V}^*$ ▷ Eigenvalue Decomposition
6: $K \leftarrow G^{-1/2}U_2U_1^*G^{1/2}$
7: $V \leftarrow G^{-1/2}\bar{V}$

2.3 Eigenvalue problem

As it is introduced, DMD-type algorithms approximate the eigenvalues and eigenfunctions of the Koopman operator by the eigenvalues and eigenvectors of the matrix representation of them. Therefore, we review the methods for finding eigenvalue and eigenvectors of a matrix. Recall, for a square matrix $A \in \mathbb{C}^{n\times n}$, the eigenvalue problem is the determination of nontrivial solutions of

$$Av = \lambda v,$$

where $v \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$. Since the above equation has a nonzero solution v if and only if the determinant of $(A - \lambda I)$ is zero with some λ, the eigenvalues λ are calculated by the solution of the characteristic equation defined as

$$\det(A - \lambda I) = 0.$$

If the matrix A is diagonalizable, then A can be factorized as

$$A = V\Lambda V^*,$$

where Λ is a diagonal matrix whose diagonal elements are eigenvalues, and each column vector of V are eigenvectors corresponding to the eigenvalues. Note that the following explanation is mainly based on Stoer and Bulirsch [SB10].

Power iteration The power iteration (or power method) is one of the most intuitive methods for finding the eigenvector of a matrix. Since the foundation is back in a century, it is difficult to mention who invented this approach. Nonetheless, Golub and van der Vorst mention the algorithm stems from the contribution of Müntz [GV00].

For detail, write the eigenpair of $A \in \mathbb{C}^{n\times n}$ as $(\lambda_1, v_1), (\lambda_2, v_2), \dots (\lambda_n, v_n)$ and assume $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. Then for $q = \sum_{i=1}^{n} c_i v_i \in \mathbb{C}^n$, its easy to see that

$$\frac{1}{|\lambda_1|^m} A^m q = \sum_{i=1}^{n} c_i \frac{A^m v_i}{|\lambda_1|^m} = \sum_{i=1}^{n} \left(\frac{\lambda_i}{|\lambda_1|}\right)^m c_i v_i \to c_1 v_1 (m \to \infty).$$

Note that $\left(\frac{\lambda_i}{|\lambda_1|}\right)^m \to 0$ if $i \neq 1$ due to $\frac{|\lambda_i|}{|\lambda_1|} < 1$. Therefore, the convergence of this algorithm is linear with respect to the ratio $\frac{|\lambda_2|}{|\lambda_1|}$. Also, note that the initial vector q should not be orthogonal to v_1. If this happens, the power iteration finds the eigenvector with respect to the second-largest eigenvalue (again, if they are not orthogonal to each other). In the numerical algorithm, the initial vector q is chosen to be a random vector, which has a small chance of being orthogonal to the largest vector. Note that one uses $\|A^m q\|$ as the denominator as we often do not know the largest eigenvalue λ. The power iteration is summarized in the algorithm 5.

Algorithm 5 Power iteration

Input: A matrix $A \in \mathbb{C}^{n\times n}$ and number of iteration $N \in \mathbb{N}$
Output: $v \in \mathbb{C}$ such that $Av = \lambda v$ with largest eigenvalue λ.

1: Define $v \in \mathbb{C}$ be some random vector.
2: $k \leftarrow 1$
3: **while** k < N **do**
4: $\quad v_k \leftarrow Av$
5: $\quad v \leftarrow \frac{v_k}{\|v_k\|}$
6: $\quad k \leftarrow k + 1$
7: **end while**

However, in our case, since the matrix representation is unitary and all eigenvalue has absolute value 1, i.e., $|\lambda_1| = |\lambda_2| = \cdots = |\lambda_n|$, the assumption $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$ does not hold. One way to overcome this issue is the so-called shift technique.

Eigenvalue shift For $A \in \mathbb{C}^{n\times n}$ and its eigenpair (λ, v), consider the eigenpair of $A - \sigma I$, then

$$(A - \sigma I)v = Av - \sigma v = \lambda v - \sigma v = (\lambda - \sigma)v.$$

This means that $(\lambda - \sigma, v)$ is the eigenpair of $A - \sigma I$. In other words, the matrix $A - \sigma I$ has the same eigenvector as A while having a different eigenvalue. By using this technique, one can enforce $|\lambda_i| \neq |\lambda_j|, i \neq j$ for the eigenvalues of a unitary operator, and it allows us to find the eigenvector of a unitary matrix with power iteration. For the fast convergence, one needs to set σ such that $|\lambda - \sigma| \gg |\lambda' - \sigma|, \forall \lambda' \neq \lambda$, however, this is difficult since one does not know the exact value of all eigenvalues. Instead, assume one can obtain the smallest eigenvalue, i.e., λ_n of $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$, then one can set σ as a good approximation of λ and can make $|\lambda - \sigma| \ll |\lambda' - \sigma|, \forall \lambda' \neq \lambda$. To obtain the eigenvector corresponding to the smallest eigenvalue λ_n, the inverse iteration is used.

Inverse iteration The inverse iteration is simply the power iteration for the inverse of the original matrix. Again, assume a matrix $A \in \mathbb{C}^{n\times n}$ has its eigenpairs $\{(\lambda_i, v_i)\}_{i=1}^{n}$ such that $|\lambda_1| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|$. If A^{-1} exists, then

$$Av_i = \lambda_i v_i \implies A^{-1}Av_i = \lambda_i A^{-1} v_i \implies \frac{1}{\lambda_i} v_i = A^{-1} v_i.$$

Therefore A^{-1} has eigenvalues $\frac{1}{\lambda_n}, \frac{1}{\lambda_{n-1}}, \dots, \frac{1}{\lambda_1}$ and corresponding eigenvectors $v_n, v_{n-1}, \dots, v_1$. One can observe $\frac{1}{|\lambda_n|} > \frac{1}{|\lambda_{n-1}|} \geq \cdots \geq \frac{1}{|\lambda_1|}$, and therefore, by the same way as the power iteration, for some $v = \sum_{i=1}^{n} c_i v_i \in \mathbb{C}^n$, especially $c_n \neq 0$,

$$|\lambda_n|^m A^{-m} v \to v_n \ (m \to \infty).$$

This means the inverse iteration converges to the eigenvector corresponding to the eigenvalue with the smallest absolute value. Thus, as it is explained in the shift technique, one can consider the shift with σ which is a good approximation of one eigenvalue λ such that $|\lambda - \sigma| \ll |\lambda' - \sigma|$, i.e. $\frac{1}{|\lambda-\sigma|} \gg \frac{1}{|\lambda'-\sigma|}$ and find the eigenvector of unitary matrix corresponding to the eigenvalue. There is also a variant of inverse iteration called Rayleigh quotient iteration [SB10], which updates σ each step. Rayleigh quotient iteration can be used if the matrix is Hermitian and converses faster than the inverse iteration.

Algorithm 6 Inverse iteration with shift technique

Input: A matrix $A \in \mathbb{C}^{n\times n}$, eigenvalue approximation σ, and number of iteration $N \in \mathbb{N}$
Output: $v \in \mathbb{C}$ such that $Av = \lambda v$ with eigenvalue λ which closest to σ.

```
Define v ∈ ℂ be some random vector.
k ← 1
while k < N do
    v_k ← (A − σI)^{-1} v
    if ‖v_k‖ < ε then
        k ← N                          ▷ finish iteration if the norm is almost 0
    else
        v ← v_k / ‖v_k‖
        k ← k + 1
    end if
end while
```

Lanczos algorithm The Lanczos algorithm is introduced by Lanczos [Lan50], and is an adaptation of power methods to find first $m \leq n$ different eigenvectors and eigenvalues. This algorithm also works for converting a Hermitian matrix to be tri-diagonal form, which is desirable for later-introduced QR decomposition (Algorithm 9) for the eigendecomposition.

For a matrix $A \in \mathbb{C}^{n\times n}$, assume there exists an unitary matrix $U = \begin{pmatrix} u_1 & u_2 & \dots & u_n \end{pmatrix} \in \mathbb{C}^{n\times n}$ and tri-diagonal matrix $B \in \mathbb{C}^{n\times n}$ such that

$$B = U^* A U.$$

Since A is Hermitian, B is also Hermitian as it is unitary similar to A. Thus, B can be represented as

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \dots & 0 \\ 0 & \beta_2 & \alpha_3 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \beta_{n-1} \\ 0 & 0 & \dots & \beta_{n-1} & \alpha_n \end{pmatrix}.$$

Now, rewrite the relationship as $AU = UB$, then

$$\begin{pmatrix} Au_1 & Au_2 & \dots & Au_n \end{pmatrix} = AU = UB = \begin{pmatrix} \alpha_1 u_1 + \beta_1 u_2 & \beta_1 u_1 + \alpha_2 u_2 + \beta_2 u_3 & \dots & \beta_{n-1} u_{n-1} + \alpha_n u_n \end{pmatrix}.$$

This implies

$$\begin{cases} Au_1 = \alpha_1 u_1 + \beta_1 u_2, \\ Au_i = \beta_{i-1} u_{i-1} + \alpha_i u_i + \beta_i u_{i+1} \quad (1 < i < n), \\ Au_n = \beta_{n-1} u_{n-1} + \alpha_n u_n. \end{cases}$$

Since $u_i^* u_j = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$, by multiplying u_i^* from left side to the i-th equation,

$$u_i^* A u_i = \beta_{i-1} u_i^* u_{i-1} + \alpha_i u_i^* u_i + \beta_i u_i^* u_{i+1} = \alpha_i.$$

Further, assuming $u_{i-1}, u_i, \alpha_i, \beta_{i-1}$ is known, by

$$v_{i+1} := \beta_i u_{i+1} = Au_i - \beta_{i-1}u_{i-1} - \alpha_i u_i, \|u_{i+1}\| = 1 \implies \beta_i = \|v_{i+1}\|.$$

Note that $\beta_1 = \|Au_1 - \alpha_1 u_1\|$ by the first equation. Lastly, by the above equation,

$$u_{i+1} = \frac{1}{\beta_i} v_{i+1}.$$

Thus, starting from some vector $u_1 \in \mathbb{C}^n$ such that $\|u_1\| = 1$, the terms u_i, α_i, β_i can be calculated one after another, and therefore, the unitary matrix U and the tri-diagonal matrix B can be obtained. Also, note that since $\beta_i u_{i+1} = Au_i - \beta_{i-1}u_{i-1} - \alpha_i u_i$, each column vector corresponds to each step of power iteration vectors, namely, they correspond to the Krylov subspace $\mathcal{K}_n = \text{span}\left\{u_1, Au_1, A^2u_1, \ldots, A^{n-1}u_1\right\}$ while they are modified to be orthogonal to each other. Furthermore, the Krylov subspace stores mid-term results of the power iteration, whereas that information is ignored in the power iteration. This enables efficient eigenvector calculation. Moreover, the iteration can be stopped in the desired step $m \leq n$. Although the algorithm does not produce tri-diagonal factorization if $m < n$, it works for finding m different useful eigenvectors. A summary of the Lanczos algorithm is shown in the algorithm 7.

Algorithm 7 Lanczos algorithm

Input: A matrix $A \in \mathbb{C}^{n\times n}$, and number of iteration $N \in \mathbb{N}$
Output: $B \in \mathbb{C}^{n\times n}$ and $U \in \mathbb{C}^{n\times n}$ where $B = U^*AU$ and B is tri-diagonal and U is unitary.

1: Initialize $u_1 \in \mathbb{C}^n$ as random vector with $\|u_1\| = 1$.
2: $\alpha_1 \leftarrow u_1^* A u_1$.
3: $v_2 \leftarrow Au_1 - \alpha_1 u_1$
4: $k \leftarrow 2$
5: **while** $k \leq N$ **do**
6: $\quad \beta_{k-1} \leftarrow \|v_k\|$
7: $\quad u_k = \frac{1}{\beta_{k-1}} v_k$
8: $\quad \alpha_k \leftarrow u_k^* A u_k$
9: $\quad v_{k+1} \leftarrow Au_k - \beta_{k-1}u_{k-1} - \alpha_k u_k$
10: **end while**
11: $U \leftarrow \begin{pmatrix} u_1 & u_2 & \ldots & u_n \end{pmatrix}$
12:
$$B = \begin{pmatrix} \alpha_1 & \beta_1 & 0 & 0 & \ldots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & 0 & \ldots & 0 \\ 0 & \beta_2 & \alpha_3 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \beta_{n-1} \\ 0 & 0 & \ldots & 0 & \beta_{n-1} & \alpha_n \end{pmatrix}$$

Arnoldi iteration The Arnoldi iteration is introduced by Arnoldi [Arn51] and is an extension of the Lanczos algorithm for the non-Hermitian matrix. As it is introduced in the DMD section, this algorithm plays a crucial role in the DMD algorithm derivation.

Consider a similar technique as the Lanczos algorithm, but let $H \in \mathbb{C}^{n\times n}$ be the upper Hessenberg matrix instead of the tri-diagonal matrix as the matrix A is non-Hermitian.

$$H = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 & \ldots & & \\ B_1 & \alpha_2 & \beta_2 & \gamma_2 & \ldots & \\ 0 & B_2 & \alpha_3 & \beta_3 & \ldots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \ldots & \beta_{n-1} \\ 0 & 0 & \ldots & 0 & B_{n-1} & \alpha_n \end{pmatrix}$$

Similar to the Lanczos algorithm, for a unitary matrix $U = \begin{pmatrix} u_1 & u_2 & \dots & u_n \end{pmatrix} \in \mathbb{C}^{n\times n}$, the relationship $AU = UB$ implies

$$\begin{cases} Au_1 = B_1 u_2 + \alpha_1 u_1 \\ Au_2 = B_2 u_3 + \alpha_2 u_2 + \beta_1 u_1 \\ \quad \vdots \\ Au_n = \alpha_n u_n + \beta_{n-1} u_{n-1} + \gamma_{n-2} u_{n-2} \dots . \end{cases}$$

Again, starting with random vector $u_1 \in \mathbb{C}^n$ with $\|u_1\| = 1$, by orthonormality of each column vectors of U, $u_1^* A u_1 = \alpha_1$, $v_2 := B_1 u_2 = Au_1 - \alpha_1 u_1$, and $B_1 = \|v_2\|$. For $k > 1$, $Au_k = B_k u_{k+1} + \alpha_k u_k + \beta_{k-1} u_{k-1} + \dots$, by considering $(B)_{j,k} = u_j A u_k$ for $1 \le j \le k$, and $v_{k+1} := B_k u_{k+1} = Au_k + \alpha_k u_k + \beta_{k-1} u_{k-1} + \dots$, $B_k = \|v_{k+1}\|$. Thus, each variable can be calculated iteratively. A summary of the Arnoldi iteration is given in algorithm 8.

Algorithm 8 Arnoldi iteration

Input: A matrix $A \in \mathbb{C}^{n\times n}$, and number of iteration $N \in \mathbb{N}$
Output: $U \in \mathbb{C}^{n\times N}$ and $H \in \mathbb{C}^{N\times N}$ where $H = Q^* A Q$ and H is upper Hessenberg.

Initialize $H \in \mathbb{C}^{N\times N}$ with zero matrix.
Initialize u_1 with random vector with $\|u_1\| = 1$.
$k \leftarrow 1$
while $k < N$ **do**
    $j \leftarrow 1$
    while $j \le k$ **do**
        $H_{j,k} \leftarrow u_j^* A u_k$
    end while
    $v_{k+1} \leftarrow Au_k - \sum_{l=1}^{k} H_{k,l} u_l$
    $H_{k+1,k} \leftarrow \|v_{k+1}\|$
    $u_{k+1} \leftarrow \frac{1}{\|v_{k+1}\|} v_{k+1}$
end while
$U \leftarrow \begin{pmatrix} u_1 & u_2 & \dots & u_n \end{pmatrix}$

QR algorithm As introduced in [Wat08] and [Wat82], the QR algorithm is one of the most useful algorithms to find eigenvalues (and eigenvectors) of a matrix. This algorithm is closely related to the power iteration (algorithm 5) or inverse iteration (algorithm 6). Note that this algorithm produces an upper triangular matrix whose diagonal elements are the eigenvalue of the original matrix. However, the column vectors of the corresponding unitary matrix are not eigenvectors. In other words, this algorithm produces Schur decomposition, not eigendecomposition. Also, note that we often refer to this algorithm later in the main part.

Before introducing the QR algorithm, we introduce QR decomposition as it is the core of the QR algorithm. The QR decomposition states that for any square matrix $A \in \mathbb{C}^{n\times n}$, it can be decomposed as

$$A = QR,$$

where Q is an unitary matrix and R is an upper triangular matrix. If A is invertible, then the factorization is unique if the diagonal elements of R is enforced to be positive. To perform such decomposition, the Gram-Schmidt process or Householder transformation (Householder reflection) can be used, and the latter is often used due to its numerical stability.

First, we introduce the QR decomposition with the Gram-Schmidt process, which is numerically unstable but intuitive. Remember that the vector projection of a onto u, denoted as $\mathrm{proj}_u(a)$, is defined as

$$\mathrm{proj}_u(a) = \frac{\langle u, a\rangle}{\langle u, u\rangle} u.$$

For a matrix $A = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} \in \mathbb{C}^{n\times n}, a_i \in \mathbb{C}^n$, consider applying Gram-Schmidt process to a_i:

$$\begin{aligned} u_1 &= a_1 & e_1 &= \frac{u_1}{\|u_1\|} \\ u_2 &= a_2 - \text{proj}_{u_1}(a_2) & e_2 &= \frac{u_2}{\|u_2\|} \\ &\vdots \\ u_n &= a_n - \sum_{k=1}^{n-1} \text{proj}_{u_k}(a_n) & e_n &= \frac{u_n}{\|u_n\|}. \end{aligned}$$

Note $\{u_i\}$ are orthogonal and $\{e_i\}$ are orthonormal. Also, note that $\langle e_i, a_i\rangle = \|u_i\|$ because $\langle e_i, a_i\rangle = \frac{1}{\|u_i\|}(\langle u_i, u_i - \sum_{k=1}^{i} \text{proj}_{u_k}(a_i)\rangle) = \frac{1}{\|u_i\|}\langle u_i, u_i\rangle = \|u_i\|$. Now, by using orthonormal basis $\{e_i\}$, a_i can be written as

$$a_i = \sum_{k=1}^{i} \langle e_k, a_i\rangle e_k.$$

Therefore,

$$A = QR, \text{ where } Q = \begin{pmatrix} e_1 & e_2 & \dots & e_n \end{pmatrix}, R = \begin{pmatrix} \langle e_1, a_1\rangle & \langle e_1, a_2\rangle & \dots & \langle e_1, a_n\rangle \\ 0 & \langle e_2, a_2\rangle & \dots & \langle e_2, a_n\rangle \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \langle e_n, a_n\rangle \end{pmatrix}.$$

However, as introduced, this algorithm is not numerically stable if the columns of the matrix A are nearly linearly dependent. For details, please refer to Stoer and Bulirsch [SB10] (p229). Therefore, QR decomposition is often done with Householder transformation thanks to its numerical stability.

Householder transformation was introduced by Householder [Hou58] and is a linear transformation of a reflection at a hyperplane. The hyperplane can be represented as the normal vector, which is a vector $w \in \mathbb{C}^n$ with $\|w\| = 1$ and orthogonal to the hyperplane. Now, the Householder matrix $H \in \mathbb{C}^{n\times n}$ is defined as

$$H := I - 2ww^*.$$

This matrix is Hermitian and unitary, since $H^* = I - 2(ww^*)^* = I - 2(ww^*) = H$ and $HH^* = H^2 = (I - 2(ww^*))^2 = I - 4ww^* + 4(ww^*)(ww^*) = I$. The idea of the QR decomposition with Householder transformation is defining the Householder matrix such that it transforms (projects) the first column of the given matrix onto the first coordinate. By repeating this procedure, since the Householder matrix is Hermitian and unitary, the resulting matrix is upper triangular with the corresponding unitary matrix.

For detail, for a vector $x = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix}^T \in \mathbb{C}^n$, consider defining the hyperplane (= the parameter w) such that

$$Hx = \alpha e_1,$$

where $\alpha \in \mathbb{C}$ and $e_1 = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}^T$. In other words, defining the hyperplane transforms the vector x onto the first coordinate. Such parameters can be determined as

$$w = \frac{x - \alpha e_1}{\|x - \alpha e_1\|}, \alpha = -e^{i\arg(x_1)}\|x\|.$$

For detail, check Stoer and Bulirsch [SB10] (p225). Therefore, define H_1 for a_1, then

$$H_1 A = \begin{pmatrix} \alpha_1 & x & \dots & x \\ 0 & x & \dots & x \\ \vdots & \vdots & \dots & \vdots \\ 0 & x & \dots & x \end{pmatrix} := \begin{pmatrix} \alpha_1 & x \\ 0 & \bar{A}_2 \end{pmatrix}.$$

Note that x just represents some number and does not mean they are the same. For $\bar{A}_2 \in \mathbb{C}^{(n-1)\times(n-1)}$, one can again consider the Householder transformation $\bar{H}_2 \in \mathbb{C}^{(n-1)\times(n-1)}$ such that

$$\bar{H}_2\bar{A}_2 = \begin{pmatrix} \alpha_2 & x & \dots & x \\ 0 & x & \dots & x \\ \vdots & \vdots & \dots & \vdots \\ 0 & x & \dots & x \end{pmatrix} := \begin{pmatrix} \alpha_2 & x \\ 0 & \bar{A}_3 \end{pmatrix}.$$

Therefore, define $H_2 := \begin{pmatrix} I_1 & 0 \\ 0 & \bar{H}_2 \end{pmatrix}$, then

$$H_2H_1A = \begin{pmatrix} \alpha_1 & x & x & \dots & x \\ 0 & \alpha_2 & x & \dots & x \\ 0 & 0 & x & \dots & x \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & x & \dots & x \end{pmatrix}.$$

By repeating procedure with $H_k := \begin{pmatrix} I_{k-1} & 0 \\ 0 & \bar{H}_k \end{pmatrix}$, $H_n \dots H_1A := R$ becomes an upper triangular matrix. Since H_k are Householder matrices, they are unitary. Thus, $Q := (H_n \dots H_1)^*$ is also unitary, and one obtains the QR decomposition $A = QR$ where Q is unitary and R is an upper triangular matrix. A summary of the QR decomposition with Householder transformation is given in algorithm 9.

Algorithm 9 QR decomposition with Householder transformation

Input: A matrix $A = \begin{pmatrix} a_1 & a_2 & \dots & a_n \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix} \in \mathbb{C}^{n\times n}, a_i \in \mathbb{C}^n$.

Output: An unitary matrix $Q \in \mathbb{C}^{n\times n}$ and an upper triangle matrix $R \in \mathbb{C}^{n\times n}$ such that $A = QR$.

$k \leftarrow 1$
while $k \leq n$ **do**
  $\bar{A}_k = \begin{pmatrix} \bar{a}_k & \bar{a}_{k+1} & \dots & \bar{a}_n \end{pmatrix} \in \mathbb{C}^{(n-k)\times(n-k)}$ where $\bar{a}_l = \begin{pmatrix} a_{k,l} & a_{k+1,l} & \dots & a_{n,l} \end{pmatrix}^T, k \leq l \leq n$
  $\alpha \leftarrow -e^{i\arg(a_{k,k})}\|\bar{a}_k\|$
  $u_k \leftarrow \bar{a}_k - \alpha\bar{e}_1$
  $w_k \leftarrow \frac{u_k}{\|u_k\|}$
  $\bar{H}_k \leftarrow I_{n-k+1} - 2ww^*$
  $H_k \leftarrow \begin{pmatrix} I_{k-1} & 0 \\ 0 & \bar{H}_k \end{pmatrix}$
  $k \leftarrow k + 1$
end while
$Q \leftarrow (H_n \dots H_1)^*$
$R \leftarrow H_n \dots H_1A$

Now, we introduce the QR algorithm. For the initial matrix A_1, define a matrix A_2 as $A_2 = R_1Q_1$ where $A_1 = Q_1R_1$ is the QR decomposition given by one of the above-mentioned algorithms. By definition, $A_2 = R_1Q_1 = Q_1^*A_1Q_1$, and as Q is unitary, A_1 and A_2 are unitary similar. Thus, they have the same eigenvalues. By repeating this procedure,

$$A_{n+1} = P_n^*A_1P_n, \text{ where } P_n = Q_1Q_2 \dots Q_n.$$

Since P_n is a product of unitary matrices, it is again a unitary matrix. Hence, A_{n+1} and A_1 are again unitary similar, and they have the same eigenvalues. Note that for $U_n = R_nR_{n-1} \dots R_1$, by $A_{n+1} = P_n^*A_1P_n \implies P_nA_{n+1} = A_1P_n$. Therefore,

$$P_nU_n = Q_1 \dots Q_{n-1}A_nR_{n-1} \dots R_1 = A_1Q_1 \dots Q_{n-1}R_{n-1} \dots R_1 = \dots = (A_1)^n.$$

Now, we introduce a brief sketch of the convergence analysis. For detailed proof, please check Stoer and Bulirsch [SB10] (p422). Assuming A_1 has n distinct eigenvalues λ_k with $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$ and has the eigendecomposition $A_1 = VDV^*$ where V is unitary and D is diagonal (and thus has inverse). The key of the proof is the LR decomposition of $LR = V^*$ with a lower triangular matrix L and an upper triangular matrix R where the diagonal element of L is 1, which introduces

$$(A_1)^i = VD^iV^* = VD^iLR = V(D^iLD^{-i})D^iR.$$

Here, the term (D^iLD^{-i}) is a multiplication of lower triangular matrices, thus again lower triangular with the values

$$(D^iLD^{-i}) = \begin{pmatrix} \left(\frac{\lambda_1}{\lambda_1}\right)^i l_{11} & 0 & \dots & 0 \\ \left(\frac{\lambda_2}{\lambda_1}\right)^i l_{21} & \left(\frac{\lambda_2}{\lambda_2}\right)^i l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \left(\frac{\lambda_n}{\lambda_1}\right)^i l_{n1} & \left(\frac{\lambda_n}{\lambda_2}\right)^i l_{n2} & \dots & \left(\frac{\lambda_n}{\lambda_n}\right)^i l_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ \left(\frac{\lambda_2}{\lambda_1}\right)^i l_{21} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \left(\frac{\lambda_n}{\lambda_1}\right)^i l_{n1} & \left(\frac{\lambda_n}{\lambda_2}\right)^i l_{n2} & \dots & \left(\frac{\lambda_n}{\lambda_{n-1}}\right)^i l_{nn-1} & 1 \end{pmatrix}.$$

By the same reason as power iteration and inverse iteration, since $|\lambda_k| > |\lambda_j|$ for $k < j$, $\left(\frac{\lambda_n}{\lambda_{n-1}}\right)^i \to 0$ as $i \to \infty$, and thus $\lim_{i\to\infty} D^iLD^{-i} = I$. Combined with the property $(A_1)^n = P_nU_n$, one can show A_i converges to the upper triangular matrix. Therefore, the convergence determined by the ratio $\frac{|\lambda_k|}{|\lambda_j|}$ is equivalent to the power iteration and inverse iteration. Note that the QR algorithm is, in fact, equivalent to the simultaneous iteration starting with $e_1, e_2, \dots, e_n$ where e_i is the i-th unit vector. Remember that simultaneous iteration is an extension of the power iteration applied to the k-dimensional subspace $\mathcal{S}_k$ and finds k eigenvectors corresponding to the k largest eigenvalues. Therefore, one can observe the QR algorithm to be an extension of power iteration. If A_1 is a symmetric (Hermitian) matrix, A_n converges to the diagonal matrix, and therefore QA_nQ^* converges to the eigendecomposition. For a non-Hermitial matrix, if A_1 is non-singular and the eigenvalues satisfy $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, it converges to the upper triangular matrix. Thus, QA_nQ^* converges to the Schur form of the matrix. Note that the QR decomposition is unique up to rescaling of the columns/row vector of Q by phase factor $\sigma = e^{i\phi}$, $|\sigma| = 1$.

Algorithm 10 QR Algorithm

Input: A matrix $A \in \mathbb{C}^{n\times n}$, eigenvalue approximation σ, and number of iteration $N \in \mathbb{N}$
Output: $\Lambda \in \mathbb{C}^{n\times n}$ and $V \in \mathbb{C}^{n\times n}$ where $V\Lambda V^* = A$ and diagonal elements of Λ are eigenvalues.

$k \leftarrow 1$
while not converge **do**
    $Q_k, R_k \leftarrow$ QR decomposition(A_k)
    $A_{k+1} \leftarrow R_kQ_k$
    $k \leftarrow k + 1$
end while
$\Lambda \leftarrow A_k$
$V = Q_kQ_{k-1}\dots Q_1$

Note, however, that this method is expensive for the dense matrix. Completing the i-th step for $n \times n$ matrix, it requires $O(n^3)$ operations. Moreover, the convergence is very slow if $|\frac{\lambda_i}{\lambda_j}| \approx 1$ by the same reason as power iteration or inverse iteration. As for the remedy of the first issue, the QR algorithm is applied only for the Hessenberg matrix or the tri-diagonal matrix for the Hessenberg matrix. For such matrices, the i-th step of the QR algorithm only takes $O(n^2)$ operations. Different algorithms are used to make the original matrix in such adequate form, such as the Arnoldi iteration and the Lanczos algorithm introduced above. For the second issue, remember that the column vectors of the QR decomposition can be interpreted as the inverse iteration, as mentioned in the convergence analysis of the QR algorithm. Therefore, the shift techniques make the convergence faster. This is the idea of an implicit shifted QR algorithm.

Implicit shifted QR algorithm Now, one considers QR decomposition for $A_i - k_i I = Q_i R_i$ and define $A_{i+1} := R_i Q_i + k_i I$ with some parameter k_i. By definition,

$$A_{i+1} = R_i Q_i + k_i I = (Q_i^* A_i - k_i Q_i^*) Q_i + k_i I = Q_i^* A_i Q_i,$$

i.e., A_{i+1} is unitary similar to A_i. Again, define $P_n = Q_1 Q_2 \dots Q_n$ and $U_n = R_n R_{n-1} \dots R_1$, then P_n is an unitary matrix and $A_{n+1} = P_n^* A_1 P_n$ implies A_{n+1} and A_1 are unitary similar having the same eigenvalues. Furthermore, by $A_{n+1} = P_n^* A_1 P_n \implies P_n A_{n+1} = A_1 P_n$, one obtains

$$\begin{aligned}
P_n U_n &= Q_1 \dots Q_{n-1} Q_n R_n R_{n-1} \dots R_1 \\
&= Q_1 \dots Q_{n-1} (A_n - k_n I) R_{n-1} \dots R_1 \\
&= A_1 Q_1 \dots Q_{n-1} R_{n-1} \dots R_1 - k_n Q_1 \dots Q_{n-1} R_{n-1} \dots R_1 \quad (\because P_{n-1} A_n U_{n-1} = A_1 P_{n-1} U_{n-1}) \\
&= (A_1 - k_n I) Q_1 \dots Q_{n-1} R_{n-1} \dots R_1 \\
&\;\;\vdots \\
&= (A_1 - k_n I)(A_1 - k_{n-1} I) \dots (A_1 - k_1 I).
\end{aligned}$$

Now, one needs to consider how the i-th shift parameter k_i should be determined. Remember, with the inverse iteration, the shift parameter is chosen as the good approximation of the eigenvalue such that $|\frac{\lambda_l - k_i}{\lambda_j - k_i}| \ll 1$ for all $l \neq j$. There are three well-known strategies to choose shift parameter k_i: (1) The Rayleigh shift (R-shift), (2) The Wilkinson shift (W-shift), and (3) The mixed shift (M-shift). Recall in the convergence analysis, the last diagonal element converges to the smallest eigenvalue as i becomes large. Therefore, the R-shift set

$$k_i := a_{n,n}^i$$

where $a_{n,n}^i$ is the last diagonal element of A_i. The W-shift also takes into account the case where a matrix having eigenvalues $|\lambda_i| \leq |\lambda_{i+1}|$ instead of strict inequality by defining k_i to be the eigenvalue of the 2×2 matrix

$$\begin{pmatrix} a_{n-1,n-1}^{(i)} & a_{n-1,n}^{(i)} \\ a_{n,n-1}^{(i)} & a_{n,n}^{(i)}, \end{pmatrix},$$

for which $|a_{n,n}^{(i)} - \lambda|$ is the smallest. The M-shift is the combination of the R-shift and the W-shift, which uses one of the shift strategies based on the parameter and the 2×2 matrix values. With the shift technique, one can prove that the convergence is cubic. For detailed proof, check Stoer and Bulirsch [SB10] (p430). A summary of the implicit shifted QR algorithm is given in algorithm 11.

Algorithm 11 implicit shifted QR Algorithm

Input: A matrix $A \in \mathbb{C}^{n \times n}$, eigenvalue approximation σ, and number of iteration $N \in \mathbb{N}$
Output: $\Lambda \in \mathbb{C}^{n \times n}$ and $V \in \mathbb{C}^{n \times n}$ where $V \Lambda V^* = A$ and diagonal elements of Λ are eigenvalues.

```
k ← 1
while not converge do
    λ_k ← Shift technique
    Q_k, R_k ← QR decomposition(A_k − λ_k I)
    A_{k+1} ← R_k Q_k + λ_k I
    k ← k + 1
end while
Λ ← A_k
V = Q_k Q_{k−1} ... Q_1
```

Implicit shifted QR algorithm for unitary Hessenberg matrix Recall that the i-th step of the QR algorithm for a Hessenberg matrix $A \in \mathbb{C}^{n \times n}$ takes $O(n^2)$ operations. Gragg [Gra86; WG02] introduced for a

unitary Hessenberg matrix, one can perform this process just in $O(n)$ operations by using the so-called Schur parametrization of A, which is based on Schur's work [Sch18].

Assume that A is transformed to upper Hessenberg matrix $U \in \mathbb{C}^{n\times n}$ with positive subdiagonal element $\{\sigma_i\}_{i=1}^n$ by, for example, the Householder transformation. Then, U can be factorized into a product of n unitary matrices called Schur parametric form of U:

$$U = U(\gamma_1, \gamma_2, \dots \gamma_n) = G_1(\gamma_1)G_2(\gamma)\dots G_3(\gamma),$$

where

$$\begin{cases} G_k := G_k(\gamma_k) = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & \begin{bmatrix} -\gamma_k & \sigma_k \\ \sigma_k & \bar{\gamma_k} \end{bmatrix} & 0 \\ 0 & 0 & I_{n-k-1} \end{pmatrix}, |\gamma_k|^2 + \sigma_k^2 = 1 \text{ for } 1 \le k < n \\ G_n := G_n(\gamma_n) = \begin{pmatrix} I_{n-1} & 0 \\ 0 & -\gamma_n \end{pmatrix}, |\gamma_n|^2 = 1. \end{cases}$$

For detailed proof, check Gragg [Gra86]. By using this parameterization, one can represent the original matrix U as

$$U = \begin{pmatrix} -\bar{\gamma}_0\gamma_1 & -\bar{\gamma}_0\sigma_1\gamma_2 & \dots & -\bar{\gamma}_0\sigma_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ \sigma_1 & -\bar{\gamma}_1\gamma_2 & \dots & -\bar{\gamma}_1\sigma_2\cdots\sigma_{n-1}\gamma_n \\ 0 & \sigma_2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & -\bar{\gamma}_{n-1}\gamma_n \end{pmatrix},$$

where $\gamma_0 := 0$ and $(U)_{kl} := u_{kl} = -\bar{\gamma}_{k-1}\sigma_k\sigma_{k+1}\cdots\sigma_{l-1}\gamma_l$ for $k < l$.

Now, consider the i-th step of the implicit shifted QR algorithm with shift λ_i, let A_i be an upper Hessenberg matrix with positive subdiagonal element, and consider the QR decomposition $H_i - \lambda_i I = Q_i R_i$ with unitary matrix Q_i and upper triangular matrix R. Recall the i-th step of the shifted QR algorithm is $A_{i+1} = Q_i^* A_i Q_i$. By the relationship and upper triangularity of R_i, the unitary matrix Q_i is also an upper Hessenberg matrix with positive subdiagonal elements. Thus, it can be represented in the Schur parametric form. Therefore, A_{i+1} is also an upper Hessenberg matrix with positive subdiagonal elements except the last subdiagonal element, which can be zero. In both cases, the resulting matrix A_{i+1} can be represented in the Schur parametric form, and Gragg showed their parameters can be determined by the Schur parameters of A_i and the shift parameter λ_i in $O(n)$ operation.

3 Exponential algorithms for mpEDMD

Summary of the settings of this thesis Now, we introduce our approach. Again, the main idea of our approach is integrating the proposition 2.1 of the Koopman operator into eigensolver for the Koopman matrix representation, especially for DMD-type algorithms. Recall this proposition allows us to obtain eigenpair just by taking the product $(\lambda_1\lambda_2, \varphi_1\varphi_2)$ or exponential $(\lambda_1^p, \varphi_1^p)$ with $p \in \mathbb{R}_+$ (or $p \in \mathbb{R}$ if the eigenfunction vanish nowhere) of some already found eigenpairs (λ_1, φ_1), (λ_2, φ_2) of the Koopman operator U. Note that this proposition is valid for the eigenpairs (eigenvalues and eigenfunctions) of the original Koopman operator U but it might not be true for the eigenpairs (eigenvalue and eigenvectors) of the matrix representation because the eigenvector, which corresponds to the eigenfunction φ^p, might not be on the projected space.

We consider the ergodic systems, where the time average equals the space average. The ergodic system is denoted as $(\mathcal{M}, \mathfrak{B}, \mu, T)$ and is defined by the state space $\mathcal{M}$ which is measurable with a σ-algebra $\mathfrak{B}$, the normalized invariant measure μ s.t. $\mu(\mathcal{M}) = 1$, and the evolution function $T : \mathcal{M} \to \mathcal{M}$ which is measure-preserving and we also assume T is invertible. Furthermore, the system is discrete, i.e., the state evolution $\{p_n\}_{n\in\mathbb{N}}, p_n \in \mathcal{M}$ is written in the form

$$p_{n+1} = T(p_n).$$

For using the DMD-type algorithm, we assume that we are given an ergodically sampled state evolution dataset through some vector-valued observable function

$$F : \mathcal{M} \to \mathbb{C}^m, F(x) = \begin{pmatrix} f_1(x) & f_2(x) & \dots & f_m(x) \end{pmatrix}^T, f_i \in \mathcal{F} := L^2(\mathcal{M}, \mu),$$

and define the Koopman operator U as

$$U : \mathcal{F} \to \mathcal{F}, [UF](x) := F(T(x)) = \begin{pmatrix} [Uf_1](x) & [Uf_2](x) & \dots & [Uf_m](x) \end{pmatrix}^T,$$

i.e., for an initial state $p_0 \in \mathcal{M}$, the dataset X is

$$\begin{aligned} X &= \left\{F(p_0), F(T(p_0)), \dots, F(T^{n-1}(p_0))\right\} = \left\{F(p_0), [UF](p_0), \dots, [U^{n-1}F](p_0)\right\} \\ &= \begin{pmatrix} f_1(p_0) & [Uf_1](p_0) & \dots & [U^{n-1}f_1](p_0) \\ f_2(p_0) & [Uf_2](p_0) & \dots & [U^{n-1}f_2](p_0) \\ \vdots & \vdots & \ddots & \vdots \\ f_m(p_0) & [Uf_m](p_0) & \dots & [U^{n-1}f_m](p_0) \end{pmatrix}. \end{aligned}$$

The set of observable functions $\mathcal{F}$ is defined as the Hilbert space

$$\mathcal{F} = L^2(\mathcal{M}, \mu) := \left\{ f : \mathcal{M} \to \mathbb{C} \mid \int_{\mathcal{M}} |f|^2 \, d\mu < \infty \right\},$$

whose inner product is defined by $\langle f_1, f_2 \rangle := \int_{\mathcal{M}} f_1(x)\overline{f_2(x)}d\mu(x)$. Recall under the measure-preserving setting with invertible evolution function T, the Koopman operator U is unitary, and therefore, the eigenvalues of U lie on the unit circle $\mathbb{T} := \{z \in \mathbb{C} \mid |z| = 1\}$.

Since the dynamical system is measure-preserving, we choose mpEDMD to obtain the Koopman operator's matrix representation from datasets. Remember that in mpEDMD, we assume the dataset to be the pair of snapshots of the state

$$X = (\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_M}), Y = (\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_M}), \mathbf{y_k} = T(\mathbf{x_k}),$$

and dictionaries of observables

$$\mathcal{D} = \{f_1, f_2, \cdots, f_K\}, f_k \in \mathcal{F}.$$

Note since we assumed the dataset is ergodically sampled, we define X, Y as

$$X = (\mathbf{x_0}, \mathbf{x_1}, \cdots, \mathbf{x_{M-1}}), Y = (\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_M}).$$

The output of the mpEDMD algorithm is the matrix representation K of the underlying Koopman operator U, the eigenvalues Λ, and eigenvectors V of the matrix. Further, remember that we reconstruct the eigenpair (λ_j, φ_j) of the Koopman operator by

$$\varphi_j(x) = \sum_{i=1}^{r} v_{ij}\phi_i(x),$$

where (λ_j, v_j) is the j-th eigenpair of the matrix representation $K \in \mathbb{C}^{r\times r}$ given by the algorithm. One knows that the eigenvalues and eigenfunctions converge to the real eigenpair as $M, K \to \infty$. As the algorithm for finding eigendecomposition, mpEDMD uses Schur decomposition due to its numerical stability. Note, however, that the Schur decomposition is not exactly the same as the eigendecomposition. For input matrix A, the Schur decomposition produces a unitary matrix $Z \in \mathbb{C}^{n\times n}$ and upper triangular matrix $T \in \mathbb{C}^{n\times n}$ such that $A = ZTZ^*$ whereas the eigendecomposition produces a unitary matrix $V \in \mathbb{C}^{n\times n}$ and diagonal matrix $D \in \mathbb{C}^{n\times n}$ such that $A = VDV^*$. Thus, the diagonal elements of both T and D are the eigenvalues of A, and the column vectors of V are the eigenvectors of A, but the column vectors of Z are not eigenvectors. If one wants to calculate the "true" eigenvectors with respect to the eigenvalues, as explained by Stoer and Bulirsch [SB10](p435), one should use an eigenvector algorithm, such as inverse iteration (algorithm 6) after the mpEDMD algorithm. The algorithm for this case is summarized as the algorithm 12.

Algorithm 12 Eigenfunction reconstruction with inverse iteration

Require: The matrix representation $K \in \mathbb{C}^{r\times r}$ and eigenvalues $\Lambda = \begin{pmatrix}\lambda_1 & \lambda_2 & \dots & \lambda_r\end{pmatrix}$ given by mpEDMD with the number of dictionaries r.
Ensure: Eigenfunctions $\Phi := \{\varphi_1, \varphi_2, \dots, \varphi_r\}$ where (λ_k, φ_k) is the eigenpair of U.
  $k \leftarrow 1$
  while $k \le r$ **do**
    $v \leftarrow$ Inverse iteration(K, λ_k)
    $\varphi_k \leftarrow \sum_{i=1}^{r}(v)_i\phi_i$
    $k \leftarrow 1$
  end while
  $\Phi \leftarrow \{\varphi_1, \varphi_2, \dots, \varphi_r\}$

So the goal of this section is to derive algorithms to find eigenvectors from the mpEDMD matrix representation, which can be represented as the power or combination of some other already found eigenvectors and possibly numerically show the ratio of the number of such vectors in the matrix.

3.1 Real-exponential algorithm

Now, assume eigenfunction φ is given by the mpEDMD algorithm, i.e., $\varphi = \sum_{i=1}^{r} v_i\phi_i$ with eigenvector v of the matrix representation and the dictionaries of observables $\{\phi_i\}_{i=1}^{r}$ where $r \in \mathbb{N}$ is the number of dictionaries. By using this vector $v \in \mathbb{C}^r$, we consider obtaining the vector $v^{(p)} \in \mathbb{C}^r$ which corresponds to the vector of the eigenfunction φ^p, i.e.,

$$\varphi^p = \sum_{i=1}^{r} v_i^{(p)}\phi_i.$$

Note the eigenfunction φ^p is obtained by

$$\varphi^p(x) := \left(\sum_{i=1}^{r} v_i\phi_i(x)\right)^p.$$

Note that by the proposition 2.1, one can take $p \in \mathbb{R}_+$ in general, and $p \in \mathbb{R}$ if the (eigen)function φ vanishes nowhere. For convenience, we call $v^{(p)}$ and φ^p as "power vector with p", "power function with p" or simply "power vector", "power function" if the power value p is clear, respectively.

Recall from functional analysis, for orthonormal basis $\{e_k\}_{k\in B}$ of the Hilbert space $\mathcal{F} = L^2(\mathcal{M}, \mu)$, any element $f \in \mathcal{F}$ can be expressed as

$$f = \sum_{k\in B} \langle f, e_k \rangle e_k,$$

where the inner product is defined by $\langle f_1, f_2 \rangle := \int_{\mathcal{M}} f_1(x)\overline{f_2(x)}d\mu(x)$. Therefore, if the dictionaries of observables ϕ_i is an orthonormal basis of $L^2(\mathcal{M}, \mu)$, one obtains

$$\varphi^p = \sum_i \langle \varphi^p, \phi_i \rangle \phi_i.$$

By comparing with $\varphi^p = \sum_{i=1}^r v_i^{(p)} \phi_i$, $v^{(p)}$ can be expressed as

$$v^{(p)} = \begin{pmatrix} \langle \varphi^p, \phi_1 \rangle \\ \langle \varphi^p, \phi_2 \rangle \\ \vdots \\ \langle \varphi^p, \phi_r \rangle \end{pmatrix}.$$

Note that the dictionaries of the observables are finite, whereas the equation holds for the infinite number of basis elements. Therefore, there is a reconstruction error, and increasing the degree of dictionaries will reduce such errors. Also, recall that an example of such dictionary of the observables is the Fourier basis $\{e_k : [-\pi, \pi] \to \mathbb{C} \mid e_k(x) = \frac{1}{\sqrt{2\pi}} e^{ikx}\}_{k\in\mathbb{Z}}$ which is an orthonormal basis of $L^2([-\pi, \pi], \mu)$ since

$$\langle e_k, e_l \rangle = \int_{-\pi}^{\pi} e_k(x)\overline{e_l(x)}dx = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i(k-l)x}dx = \begin{cases} 1 & \text{if } k = l \\ \frac{1}{2\pi}\left[\frac{1}{i(k-l)} e^{i(k-l)x}\right]_{-\pi}^{\pi} = 0 & \text{if } k \neq l \end{cases}.$$

Also, recall that since the functional inner product is defined as integral, one can use a numerical integration method, such as the quadrature rule [SB10], for calculating them. As a summary, we obtain a lemma 3.1 and an algorithm 13.

Lemma 3.1. *Assume the given observables $\{\phi\}_k$ are the orthonormal basis of $\mathcal{F}$, then all eigenfunction $\varphi \in \mathcal{F}$ can be represented as $\varphi = \sum_k \langle \varphi, \phi_k \rangle \phi_k$. Especially, for eigenfunction $\varphi_{v_i} = \sum_k (v_i)_k \phi_k$, $(v_i)_k = \langle \varphi_{v_i}, \phi_k \rangle$.*

Algorithm 13 obtain power vector from an eigenvector

Require: An eigenvector $v \in \mathbb{C}^r$ of the Koopman matrix K, the dictionaries $\{\phi_i\}_{i=1}^r$, and a power $p \in \mathbb{R}_+$ (or $p \in \mathbb{R}$).

Ensure: The power vector $v^{(p)}$ such that $\varphi^p = \sum_{i=1}^r v_i^{(p)} \phi_i$.

$\varphi^p(\cdot) \leftarrow \left(\sum_{i=1}^r v_i \phi_i(\cdot)\right)^p$.

$$v^{(p)} \leftarrow \begin{pmatrix} \langle \varphi^p, \phi_1 \rangle \\ \langle \varphi^p, \phi_2 \rangle \\ \vdots \\ \langle \varphi^p, \phi_r \rangle \end{pmatrix}.$$

3.1.1 Choice of power values

Now, one needs to consider what p to choose. Note that since the eigenfunction approximation $\varphi = \sum_{i=1}^r v_i \phi_i$ includes some error, for large p value, the error for power function φ^p increases respectively. Also, note that we want to find the power vector $v^{(p)}$ which is not only $\varphi^p = \sum_{i=1}^r v_i^{(p)} \phi_i$ is an eigenfunction of U but also is an eigenvector of the matrix representation K. Thus, we first consider the necessary conditions for the power p so that the power vector with p is again an eigenvalue of the matrix representation K.

Here, remember that the exponential of the complex number is defined by

$$(e^x)^y := e^{y\log(e^x)}, x, y \in \mathbb{C},$$

where the complex logarithm of non-zero complex number z, denoted as $\log(z)$, is defined by

$$e^{\log(z)} := z \in \mathbb{C}.$$

For details, check Lang [Lan93], for example. Note if z is given by the polar form $z = re^{i\theta}$ with $r > 0$ and $\theta \in \mathbb{R}$, then the complex logarithm is of the form

$$\log(z) = \ln(r) + i(\theta + 2\pi k), k \in \mathbb{Z},$$

where $\ln(\cdot)$ is the natural logarithm, i.e., $\ln(\cdot) := \log_e(\cdot)$. Also remember the principal value of the $\log(z)$ is defined as the logarithm whose imaginary part lies in the $(-\pi, \pi]$, i.e., the principal value is $\ln(r) + i\theta'$ such that $\theta' = \theta + 2\pi k \in (-\pi, \pi], k \in \mathbb{Z}$. The $\log(z)$ is generally meant for the principal value without any specification. Also note that in NumPy [Har+20], which we use for numerical experiments, $\log(z)$ returns the principal value. (See numpy.log documentation for details.) Also, in this thesis, $\log(z)$ indicates the principal value from now on.

Now, assume there exists an eigenpair (λ, φ) of the Koopman operator U where the eigenvalue λ is not equal to 1. Note that since the eigenvalues lie on the unit circle, they can be written in the form $\lambda = e^{ix}$ where $x \in (-\pi, \pi] \setminus \{0\}$. For an arbitrary $\lambda' \in \mathcal{S}^1$, one can write it as e^{iy} where $y \in (-\pi, \pi]$, and one can consider the power $p := \frac{y}{x} \in \mathbb{R}$ so that

$$\lambda^p = e^{p\log(\lambda)} = e^{ipx} = e^{i\frac{y}{x}x} = e^{iy} = \lambda'.$$

Thus, by proposition 2.1, $(\lambda^p, \varphi^p) = (\lambda', \varphi^p)$ is also an eigenpair. In other words, any value on the unit circle $\mathcal{S}^1$ is an eigenvalue of the Koopman operator U. Conversely, remember that the Koopman operator is unitary in the ergodic settings. Therefore, from functional analysis results, the eigenvalues lie on the unit circle $\mathcal{S}^1$. Thus, if one obtains an eigenvalue not equal to 1, then simply taking the above-mentioned power will produce a new eigenpair.

Lemma 3.2. *In the ergodic settings, if there exists an eigenvalue of the Koopman operator U which is not equal to 1, then λ is an eigenvalue of U if and only if $\lambda \in \mathcal{S}^1$.*

Now, we know that any values on the unit circle are eigenvalues of the Koopman operator U. Since the Koopman operator U is unitary, especially normal, recall that the eigenfunctions of a normal operator T have the following properties (Lemma 3.3) from Muscat [Mus14].

Lemma 3.3. *For a normal operator T,*

1. *If (λ, φ) is an eigenpair of T, then $(\bar{\lambda}, \varphi)$ is also an eigenpair of T^*.*
2. *If the eigenpairs $(\lambda_1, \varphi_1), (\lambda_2, \varphi_2)$ satisfies $\lambda_1 \neq \lambda_2$, then $\langle \varphi_1, \varphi_2 \rangle = 0$.*

Proof. 1). Since T is normal, $\|Tx\| = \|T^*x\|$ for all x, and $(T - \lambda I)$ is also normal. For an eigenpair (λ, φ), it follows that

$$0 = \|(T - \lambda I)\varphi\| = \|(T - \lambda I)^*\varphi\| = \|(T^* - \bar{\lambda} I)\varphi\|.$$

2). Without loss of generality, assume $\lambda_1 \neq 0$ (since one of them should be non-zero. Otherwise, the equation clearly holds),

$$\langle \varphi_1, \varphi_2 \rangle = \frac{1}{\lambda_1}\langle \lambda_1\varphi_1, \varphi_2 \rangle = \frac{1}{\lambda_1}\langle T\varphi_1, \varphi_2 \rangle = \frac{1}{\lambda_1}\langle \varphi_1, T^*\varphi_2 \rangle = \frac{1}{\lambda_1}\langle \varphi_1, \bar{\lambda_2}\varphi_2 \rangle = \frac{\lambda_2}{\lambda_1}\langle \varphi_1, \varphi_2 \rangle$$

Since $\lambda_1 \neq \lambda_2$,

$$\langle \varphi_1, \varphi_2 \rangle = 0.$$

□

Furthermore, we have the following property for the functional inner product of the eigenfunction reconstructed by the mpEDMD and the corresponding eigenvector.

Lemma 3.4. *For eigenvector v, w of the matrix representation K of the Koopman operator U and eigenfunction reconstruction $\varphi_v = \sum v_i\phi_i, \varphi_w = \sum w_i\phi_i$ by mpEDMD where $\{\phi_i\}$ are the basis dictionary, one has*

$$\langle \varphi_v, \varphi_w \rangle = \langle v, w \rangle.$$

Proof.

$$\langle \varphi_v, \varphi_w \rangle = \langle \sum_i v_i\phi_i, \sum_j w_j\phi_j \rangle = \sum_i \sum_j v_i\bar{w}_j\langle \phi_i, \phi_j \rangle = \sum_i v_i\bar{w}_i = \langle v, w \rangle.$$

Therefore

$$\langle \varphi_v, \varphi_w \rangle = 0 \Leftrightarrow \langle v, w \rangle = 0.$$

□

Since, in our setting, the Koopman operator U is unitary, especially normal, the Lemma 3.3 ensures the eigenspaces are orthogonal to each other. Combined with the lemma 3.4, the eigenvectors, which correspond to the different eigenvalues, are orthogonal to each other. Thus, if power eigenvalue λ^p does not exactly coincide with any of the eigenvalues of matrix representation K, then the power vector $v^{(p)}$ is not the eigenvector of the K and orthogonal to all of the eigenvectors of K. In other words, the condition that λ^p is an eigenvalue of K is a necessary condition for the power vector $v^{(p)}$ to be an eigenvector of K.

Theorem 3.1. *(Necessary condition for power vector to be an eigenvector) For an eigenpair (λ, v) of the matrix representation K, the power vector $v^{(p)}$ is also an eigenvector of K only if λ^p is the eigenvalue of K.*

Remember that by lemma 3.2, all values on the unit circle $\mathcal{S}^1$ are the eigenvalues of U, i.e., there are infinitely many eigenvalues of U. Therefore, the simple integer choice of power $p = 2, 3, 4 \ldots$ may not find the desired eigenvalues because it is almost impossible to obtain the same value as the eigenvalue of K just by taking integer power of one eigenvalue. Therefore, except for some special case, the power vectors $v^{(p)}$ with integer power is almost always not eigenvector of K and orthogonal to them. Note, however, due to approximation error of the eigenfunctions, it is often true that $\langle \varphi_v, \varphi_w \rangle = \langle v, w \rangle \neq 0$ even if they correspond to the different eigenvalues.

Hence, we need to find a way to obtain powers such that the exponentials of an eigenvalue with the powers are exactly the eigenvalues of the matrix representation K, and consequently, orthogonalizing eigenvectors to the power vectors for accelerating eigendecomposition process. Recall, that in the eigenvalue algorithms, such as the QR algorithm, they ensure the orthogonality of eigenvectors by applying the Gram-Schmidt-like process with the previously found vectors in each step. However, by theorem 3.1, we need to obtain the exact power value such that λ^p is an eigenvalue of K, which, without any knowledge of eigenvalues of K, is difficult to achieve during the eigendecomposition process. Therefore, calculating power vectors in between the eigendecomposition process will just increase the number of computations or even delete some meaningful information. Thus, we consider a post-processing algorithm that is applied after Schur decomposition by QR algorithm, which is performed in the original mpEDMD procedure as well. Namely, we will further assume all eigenvalues and an eigenvector is given. Recall the Schur decomposition transforms K to be

$$K = ZTZ^*$$

where $Z \in \mathbb{C}^{r\times r}$ is unitary and $T \in \mathbb{C}^{r\times r}$ is upper triangular. Note the diagonal elements of the T are the eigenvalues of K. By denoting

$$Z = \begin{pmatrix} z_1 & z_2 & \ldots & z_r \end{pmatrix}, z_i \in \mathbb{C}^r, T = \begin{pmatrix} t_{11} & t_{12} & \ldots & t_{1r} \\ 0 & t_{22} & \ldots & t_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & t_{rr} \end{pmatrix},$$

one obtains

$$KZ = ZT \implies K\begin{pmatrix} z_1 & z_2 & \dots & z_r \end{pmatrix} = \begin{pmatrix} t_{11}z_1 & t_{12}z_2 + t_{22}z_2 & \dots & \sum_{k=1}^{r} t_{k2}z_k \end{pmatrix}.$$

Therefore, $Kz_1 = t_{11}z_1$, meaning (t_{11}, z_1) is an eigenpair of K. Hence, we can obtain all eigenvalues and an eigenvector by Schur decomposition.

Now, assume we have an eigenpair (λ_1, v_1) with $\lambda_1 \neq 1$ and the eigenvalues $\lambda_2, \lambda_3, \dots \lambda_r$. Remember the eigenvalues lies on the unit circle $\mathcal{S}^1$ and therefore can represented as $\lambda_k = e^{i\theta_k}, \theta_k \in (-\pi, \pi]$. Consider determining the powers $p_2, p_3, \dots$, such that

$$\lambda_1^{p_k} = \lambda_k, 2 \leq k \leq r.$$

Here, because $\theta_k \in (-\pi, \pi]$, the principal value of the eigenvalue $\lambda_k = e^{i\theta_k}$ is given by $\log(\lambda_k) = i\theta_k$. Therefore by defining $p_k := \frac{\theta_k}{\theta_1}$, one can obtain

$$\lambda_1^{p_k} = e^{p_k \log(\lambda_1)} = e^{ip_k\theta_1} = e^{i\theta_1 \frac{\theta_k}{\theta_1}} = e^{i\theta_k} = \lambda_k.$$

Note $\theta_1 \neq 0$ since $\lambda_1 \neq 1$.

In this way, we can obtain the powers such that the exponential of an eigenvalue with the powers are truly the eigenvalues of matrix representation K. Remember that obtaining the same eigenvalues is a necessary condition for power vector $v^{(p)}$ to be an eigenvector of matrix representation K (Lemma 3.1). In other words, there might be the case where for eigenpair (λ_1, φ_1) of the Koopman operator U, the powers are given by the above way and the $(\lambda_1^{p_k}, \varphi_1^{p_k})$ is another eigenpair of U, but the corresponding eigenvector $v_1^{p_k}$ is not eigenvector of the matrix representation K. Therefore, we now need to check if the power vectors $v_1^{p_k}$ with $p_k = \frac{\theta_k}{\theta_1}$ provided by the algorithm 13 are truly the eigenvectors.

Here, through an analysis of the ergodic system, as explained by, for instance, Cornfeld, Fomin, and Sinai [CFS12], it exhibits the following properties (Lemma 3.5).

Lemma 3.5. *Assume for Koopman operator U of an ergodic system, there exists eigenfunctions φ_i, φ_j with the same eigenvalue λ i.e. $(\lambda, \varphi_i), (\lambda, \varphi_j)$ are both eigenpair of U. Assume further φ_j vanish nowhere, then*

$$\varphi_i = \alpha\varphi_j \quad a.e. \quad \text{where } \alpha \in \mathbb{C}.$$

In other words, the eigenvalues are of multiplicity 1. Especially if eigenfunctions are given by the mpEDMD algorithm, then

$$\alpha = \langle \varphi_j, \varphi_i \rangle = \langle v_j, v_i \rangle, |\alpha| = 1.$$

Proof. Recall under the ergodic setting, the eigenfunction of the Koopman operator U corresponding to the eigenvalue 1 is constant almost everywhere [CFS12]. Here, assuming φ_j vanishes nowhere, $\frac{\varphi_i}{\varphi_j}$ is well-defined and an eigenfunction with eigenvalue 1 because

$$U\frac{\varphi_i}{\varphi_j} = \frac{U\varphi_i}{U\varphi_j} = \frac{\lambda\varphi_i}{\lambda\varphi_j} = \frac{\varphi_i}{\varphi_j}.$$

Thus, $\frac{\varphi_i}{\varphi_j} = \alpha \quad a.e.$ for some constant $\alpha \in \mathbb{C}$ and hence

$$\varphi_i = \alpha\varphi_j \quad a.e.$$

Especially, in mpEDMD, one defines $\varphi_i = \sum_{k=1}^{r}(v_i)_k\phi_k$ where $v_i \in \mathbb{C}^r$ is the corresponding eigenvector with $\|v_i\| = 1$ and $\{\phi_k\}_k$ are the dictionaries of observables which is orthonormal. Thus,

$$\alpha = \alpha\langle \varphi_i, \varphi_i \rangle = \langle \varphi_j, \varphi_i \rangle = \langle v_j, v_i \rangle.$$

and

$$\sum_k (v_i)_k\phi_k = \alpha \sum_k (v_j)_k\phi_k = \sum_k (\alpha v_j)_k\phi_k \quad a.e.$$
$$\therefore v_i = \alpha v_j, \implies 1 = \|v_i\| = |\alpha|\|v_j\| = |\alpha|$$

□

The lemma 3.5 ensures the eigenvalues are of multiplicity 1, the eigenfunctions are unique up to scalar multiplication, and the corresponding eigenvectors are also unique up to scalar multiplication. Especially, for eigenpairs $(\lambda_1 = e^{i\theta_1}, \varphi_1), (\lambda_2 = e^{i\theta_2}, \varphi_2)$ of the Koopman operator U and corresponding eigenvectors v_1, v_2 of the matrix representation K, respectively, by defining $p := \frac{\theta_2}{\theta_1}$, $\lambda_1^p = \lambda_2$ and $(\lambda_1^p, \varphi_1^p) = (\lambda_2, \varphi_1^p)$ is an eigenpair. Thus, there exists some constant $\alpha \in \mathbb{C}$ such that

$$\varphi_1^p = \alpha\varphi_2 \quad a.e.,$$

which ensures the power vector $v_1^{(p)}$ provided by the algorithm 13 to be exactly eigenvectors, since

$$v_1^{(p)} = \begin{pmatrix} \langle \varphi_1^p, \phi_1 \rangle \\ \langle \varphi_1^p, \phi_2 \rangle \\ \vdots \\ \langle \varphi_1^p, \phi_r \rangle \end{pmatrix} = \begin{pmatrix} \langle \alpha\varphi_2, \phi_1 \rangle \\ \langle \alpha\varphi_2, \phi_2 \rangle \\ \vdots \\ \langle \alpha\varphi_2, \phi_r \rangle \end{pmatrix} = \alpha v_2 \implies Kv_1^{(p)} = \alpha K v_2 = \alpha\lambda_2 v_2 = \lambda_2 v_1^{(p)}.$$

Note that v_2 is the eigenvector of K corresponding to the eigenvalue λ_2. As a summary, we obtain the sufficient condition for the power vector $v_1^{(p)}$ to be an eigenvector of the matrix representation K (theorem 3.2).

Theorem 3.2. *Assume (λ_1, φ_1) is an eigenpair of the Koopman operator U and $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ are the eigenvalues of the matrix representation K. Then,*

$$v^{(p_k)} := \begin{pmatrix} \langle \varphi_1^{p_k}, \phi_1 \rangle \\ \langle \varphi_1^{p_k}, \phi_2 \rangle \\ \vdots \\ \langle \varphi_1^{p_k}, \phi_r \rangle \end{pmatrix}, p_k := \frac{\theta_k}{\theta_1}$$

is the eigenvector of K corresponding to the eigenvlaue λ_k for all $1 \le k \le n$.

In total, by combining this theorem 3.2 with the algorithm 13 for obtaining power vector, we obtain the following algorithm 14, which we call the real-exponential algorithm.

Algorithm 14 Eigensolver with real-exponential

Require: $K \in \mathbb{C}^{r\times r}$ is the matrix representation of Koopman operator U on the basis $\{\phi_i\}_{i=1}^r$, some small number ε (= 10^{-9} by default), number of dictionaries r.

Ensure: Eigenvalues Λ and Eigenvectors V.

$(T, Z) \leftarrow$ QRAlgorithm(K) such that $K = ZTZ^*$.

$\Lambda = \begin{pmatrix} \lambda_1 & \lambda_2 & \ldots & \lambda_r \end{pmatrix} \leftarrow \mathrm{diag}(T)$

$v_1 \leftarrow z_1$.

$k_0 \leftarrow 1$

$k \leftarrow 2$

if $|\arg(\lambda_1)| < \varepsilon$ **then** ▷ Rarely happens, but if the first eigenvalue is close to 1, run inverse iteration

$v_2 \leftarrow$ Inverse iteration(K, α) ▷ $\alpha \in \mathbb{C}$ is chosen s.t. $|\alpha| = 1$ and $|\arg(\alpha)| \gg 0$

$k_0 \leftarrow 2$

$k \leftarrow k + 1$

end if

while k<=N **do**

$p \leftarrow \arg(\lambda_k)/\arg(\lambda_{k_0})$ ▷ $\arg(e^{ix}) := x$ for $x \in (-\pi, \pi]$

$v_k \leftarrow v_{k_0}^p = \begin{pmatrix} \langle \varphi^p, \phi_1 \rangle \\ \langle \varphi^p, \phi_2 \rangle \\ \vdots \\ \langle \varphi^p, \phi_r \rangle \end{pmatrix}$ where $\varphi = \sum_{j=1}^r (v_i)_j \phi_j$

$k \leftarrow k + 1$

end while

$V \leftarrow \begin{pmatrix} v_1, v_2, \ldots, v_r \end{pmatrix}$

Note that if the angle of eigenvalue λ_1 is 0 or too small, then the powers can not be defined or become too large. Therefore, in the algorithm 14, we add additional inverse iteration if the angle of the eigenvalue is too small.

Now, consider the complexity of the algorithm 14. For obtaining the power vector, we take the power function and calculate the inner product by the quadrature rule

$$\int_a^b f(x)dx = \frac{b-a}{N}\sum_{k=1}^{N} f(\frac{b-a}{N}k),$$

thus, the total calculation for getting power vector from mpEDMD results is $O(Nr)$ where r is the number of dictionaries given as input and N is the number of subdomains for the quadrature rule. Note that, usually, if one wants to obtain eigenvectors, one uses inverse iteration for each column, which requires $O(r^3)$ iteration for calculating the inverse of the matrix with eigenvalue shift, and $O(rN')$ iteration for finding vector where N' is the parameter of the number of iteration, which is given by the user. Thus, the new algorithm is faster than the combination of mpEDMD and inverse iteration if $N < r^2 + N'$ because of no inverse calculation while returning the same eigenfunctions (up to scalar multiplication), which are often better approximation of the true eigenfunction than reconstruction with Schur vectors.

Also note that the uniqueness of eigenfunctions with respect to an eigenvalue implies if one is only interested in obtaining the eigenfunctions, one can use the following algorithm 15.

Algorithm 15 Eigenfunction reconstruction with real-exponential

Require: $K \in \mathbb{C}^{r\times r}$ is the matrix representation of Koopman operator U on the basis $\{\phi_i\}_{i=1}^r$, some small number ε (= 10^{-9} by default), number of dictionaries r.
Ensure: Eigenfunctions $\Phi := \{\varphi^{p_1}, \varphi^{p_2}, \dots\}$.
  $(T, Z) \leftarrow$ QRAlgorithm(K) such that $K = ZTZ^*$.
  $\Lambda = \begin{pmatrix}\lambda_1 & \lambda_2 & \dots & \lambda_r\end{pmatrix} \leftarrow \mathrm{diag}(T)$
  $v_1 \leftarrow z_1$.
  $k_0 \leftarrow 1$
  if $|\arg(\lambda_1)| < \varepsilon$ **then** ▷ Rarely happens, but if the first eigenvalue is close to 1, run inverse iteration
    $v_2 \leftarrow$ Inverse iteration(K, α) ▷ $\alpha \in \mathbb{C}$ is chosen s.t. $|\alpha| = 1$ and $|\arg(\alpha)| \gg 0$
    $k_0 \leftarrow 2$
    $k \leftarrow k + 1$
  end if
  $k \leftarrow 1$
  while $1 \le k \le r$ **do**
    $p_k \leftarrow \arg(\lambda_k)/\arg(\lambda_{k_0})$
    $k \leftarrow k + 1$
  end while
  $\varphi \leftarrow \sum_{i=1}^r (v_{k_0})_i \phi_i$
  $\Phi \leftarrow \{\varphi^{p_1}, \varphi^{p_2}, \dots\}$

3.1.2 Issue of real-exponential algorithm

However, this approach does not work well in numerics. One reason for this is the real exponential of the complex number. Recall that the real-exponential of a complex number is defined for $\alpha \in \mathbb{R}$ and $z = re^{i\theta} \in \mathbb{C}, r, \theta \in \mathbb{R}$ as

$$z^\alpha := e^{\alpha \log(z)}, \log(z) = \ln(r) + i(\theta + 2\pi n), n \in \mathbb{Z}.$$

Take $(e^{ikx})^\alpha$, for example, the logarithm (principle value) of e^{ikx} is $\log(e^{ikx}) = i(kx+2n\pi)$, where $kx+2n\pi \in (-\pi, \pi]$, thus

$$(e^{ikx})^\alpha = e^{i\alpha(kx+2n\pi)}.$$

However, if $n \neq 0$ and $\alpha \notin \mathbb{Z}$, then there is no guarantee that $e^{i\alpha kx} = e^{i\alpha(kx+2n\pi)} = (e^{ikx})^\alpha$.

Now, we show a small numerical experiment for calculating $e^{ik\alpha x}$, where i is the imaginary unit, $k \in \mathbb{Z}$, and $\alpha \in \mathbb{R} \setminus \mathbb{Z}$. As an example, consider for $k = 4 \in \mathbb{Z}$ and $\alpha = 0.5 \in \mathbb{R} \setminus \mathbb{Z}$, define the following three functions in Python (1) original function $e_1(x) = e^{ik\alpha x}$, (2) integer powered function $e_2(x) = (e^{i\alpha x})^k$, and (3) real powered function $e_3(x) = (e^{ikx})^\alpha$. Figure 1 shows the real part of these function values within the domain $x \in [-2\pi, 2\pi]$, and the red vertical dot lines represent the x-values where $e^{ikx} = -1$, i.e., $x = \frac{\pi+2\pi m}{k}, m \in \mathbb{Z}$. One can observe the function $e_3(x)$, which does real power later, takes a different value than the other two functions $e_1(x)$ and $e_2(x)$. Note that the functions $e_1(x)$ and $e_2(x)$ take the same values.

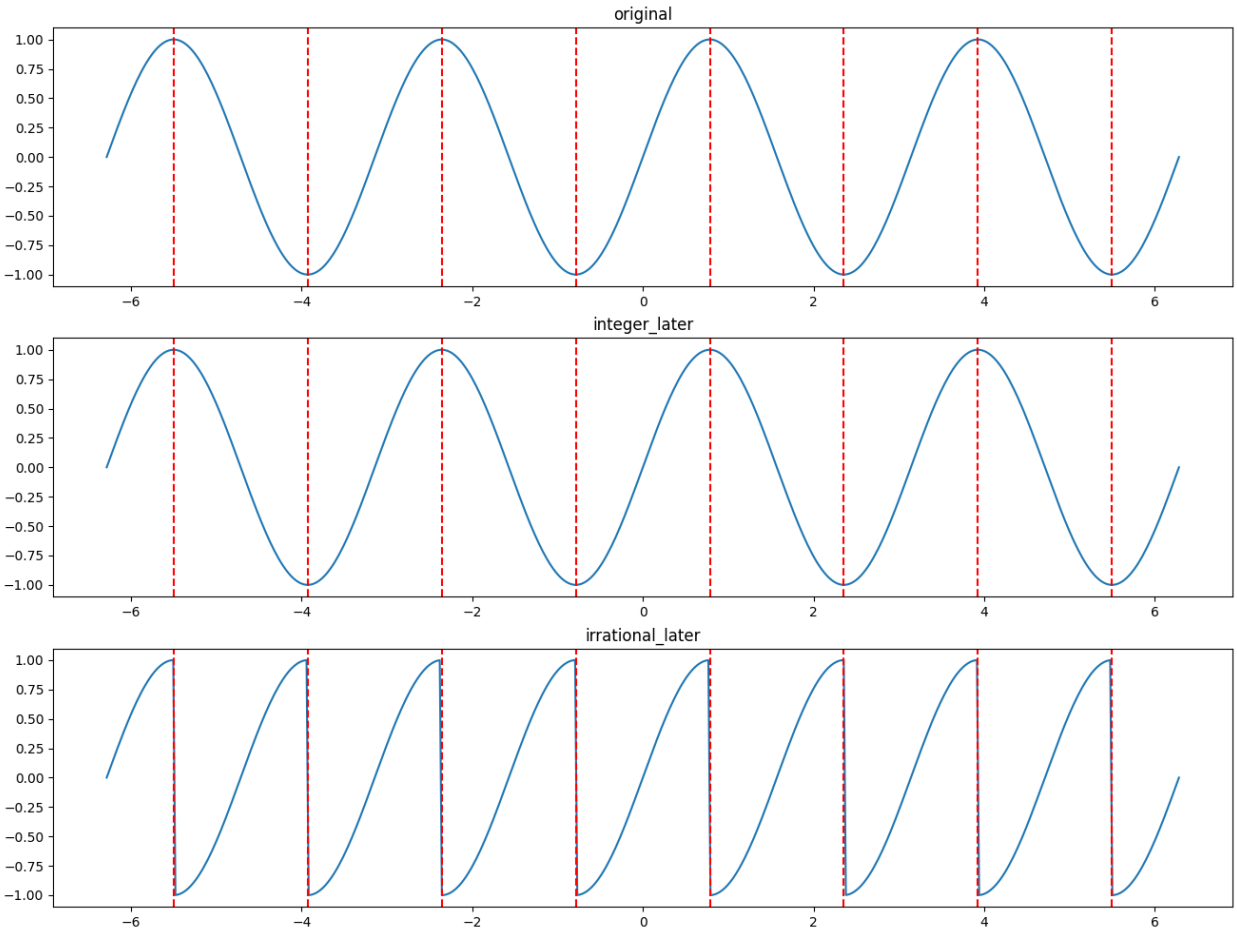


Figure 1 Exponential order difference with $e_1(x)$, $e_2(x)$, and $e_3(x)$.

Furthermore, consider the function $f_\alpha(x) = e_3(x) = (e^{ikx})^\alpha$ with small $\alpha > 0$ and positive integer $k \in \mathbb{N}$. Since e^{ikx} is $\frac{2\pi}{k}$ periodic function (with respect to x), $f_\alpha(x)$ is also $\frac{2\pi}{k}$ periodic. By definition, $f_\alpha(x) = e^{\alpha \log(e^{ikx})} = e^{i\alpha(kx+2n\pi)}$ with $n \in \mathbb{Z}, kx + 2n\pi \in (\pi, \pi]$. Here, restrict our attention for the $f_\alpha(x)$ values in one period, i.e., for some $m \in \mathbb{Z}$, consider the $f_\alpha(x)$ value within $\frac{2\pi}{k}m < x \leq \frac{2\pi}{k}(m+1)$, then

$$\begin{aligned}
\frac{2\pi}{k}m < x \leq \frac{2\pi}{k}(m+1) \\
\therefore 2\pi m < kx \leq 2\pi(m+1) \\
\therefore 2\pi(m+n) < kx + 2\pi n \leq 2\pi(m+n+1) \\
\therefore 2\pi(m+n)\alpha < (kx + 2\pi n)\alpha \leq 2\pi(m+n+1)\alpha
\end{aligned}$$

Here, for small $\alpha > 0$, both $2\pi(m+n)\alpha$ and $2\pi(m+n+1)\alpha$ are close to 0, and thus,

$$0 = \lim_{\alpha\to 0} 2\pi(m+n)\alpha < \lim_{\alpha\to 0}(kx + 2\pi n)\alpha \leq \lim_{\alpha\to 0} 2\pi(m+n+1)\alpha = 0.$$

Note that for negative $k \in \mathbb{Z}$ and $\alpha \in \mathbb{R}$, the same holds while only the direction of the inequality is changed. Hence, for small α, the angle of $f_\alpha(x)$, $(kx + 2\pi n)\alpha$, is close to 0, and therefore whose real part $\mathrm{Re}(f_\alpha(x))$ is almost 1 and the imaginary part $\mathrm{Im}(f_\alpha(x))$ is almost 0 for all values in one period, i.e.,

$$\mathrm{Re}(f_\alpha(x)) \approx 1, \mathrm{Im}(f_\alpha(x)) \approx 0, \forall x \in [\frac{2\pi}{k}m, \frac{2\pi}{k}(m+1)].$$

Figure 2 is the numerical example of this phenomena with α, $\frac{\alpha}{10}$, $\frac{\alpha}{100}$ where $\alpha = \sqrt{50}$. Note that the x-axis is the values in $[-2\pi, 2\pi]$ and y-axis is the real part of $f_\alpha(x) = (e^{ikx})^\alpha$, and the red vertical dot line represent the x-values where $e^{ikx} = -1$, i.e., $x = \frac{\pi+2\pi m}{k}, m \in \mathbb{Z}$. One can observe that for small power $\frac{\alpha}{100}$, the function $f_\alpha(x)$ takes values close to 1 for almost all x.

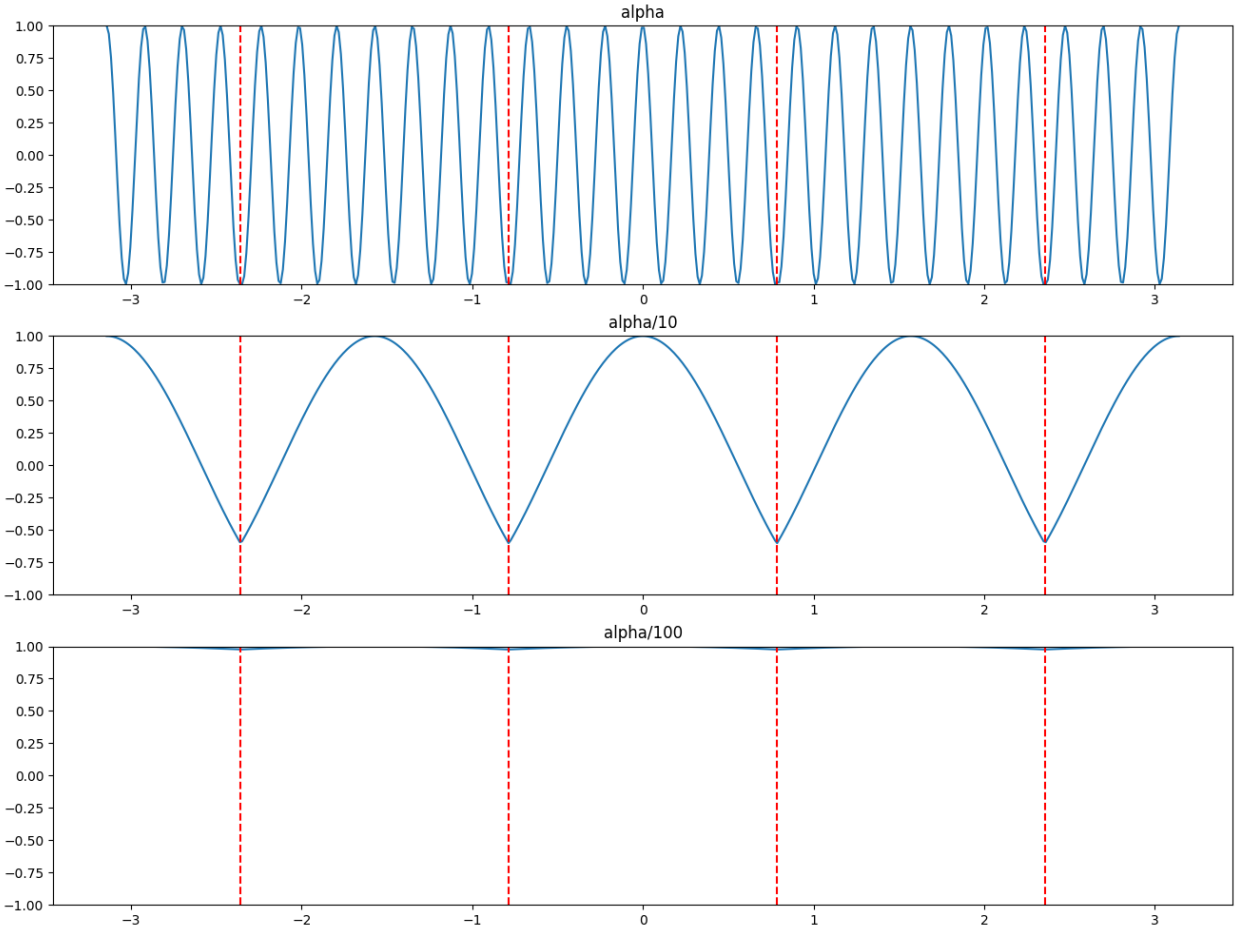


Figure 2 Real exponential with small values α, $\frac{\alpha}{10}$, $\frac{\alpha}{100}$ where $\alpha = \sqrt{50}$.

3.2 Integer-exponential algorithm

To fix real-exponential errors, one can only consider the integer power. Note that in the case α is integer, remember that the principal value is $\log(e^{ikx}) = i(kx + 2\pi n)$ with some $n \in \mathbb{Z}$, and the exponential $(e^{ikx})^\alpha, \alpha \in \mathbb{Z}$ is calculated by

$$(e^{ikx})^\alpha = e^{\alpha \log(e^{ikx})} = e^{i\alpha(kx+2\pi n)} = e^{i(\alpha kx + 2\pi n\alpha)} = e^{i\alpha kx} e^{i2\pi n\alpha} = e^{i\alpha kx}.$$

Thus, there is no exponential order difference like the real-exponential case, i.e., the case where $\alpha \in \mathbb{R} \setminus \mathbb{Z}$. To see this, for $k = 4$ and $\alpha = 3, 2, 1, \frac{1}{2}$, we show a numerical example in figure 3 where, again, the x-axis is the values in the domain $[-2\pi, 2\pi]$ and the y-axis is the value of $f_\alpha(x) = (e^{ikx})^\alpha$, and the red vertical dot line represent the x-values where $e^{ikx} = -1$, i.e., $x = \frac{\pi+2\pi m}{k}, m \in \mathbb{Z}$. In this figure, one can observe if α is an

integer, the function $f_\alpha(x)$ takes 1 or -1 on the red line (where $e^{ikx} = -1$), and the value is continuous in the sense of derivative. One can also observe that if $\alpha = \frac{1}{2}$, then the function takes value 0 on the red line, as in the period is $2\pi(m+n)\alpha \leq (kx+2\pi n)\alpha < 2\pi(m+n+1)\alpha$, therefore $\pi(m+n) \leq (kx+2\pi n)\frac{1}{2} < \pi(m+n+1)$ and $f_\alpha(\pi(m+n)) = f_\alpha(\pi(m+n+1)) = i$, i.e., $\mathrm{Re}(f_\alpha(\pi(m+n))) = \mathrm{Re}(f_\alpha(\pi(m+n+1))) = 0$.

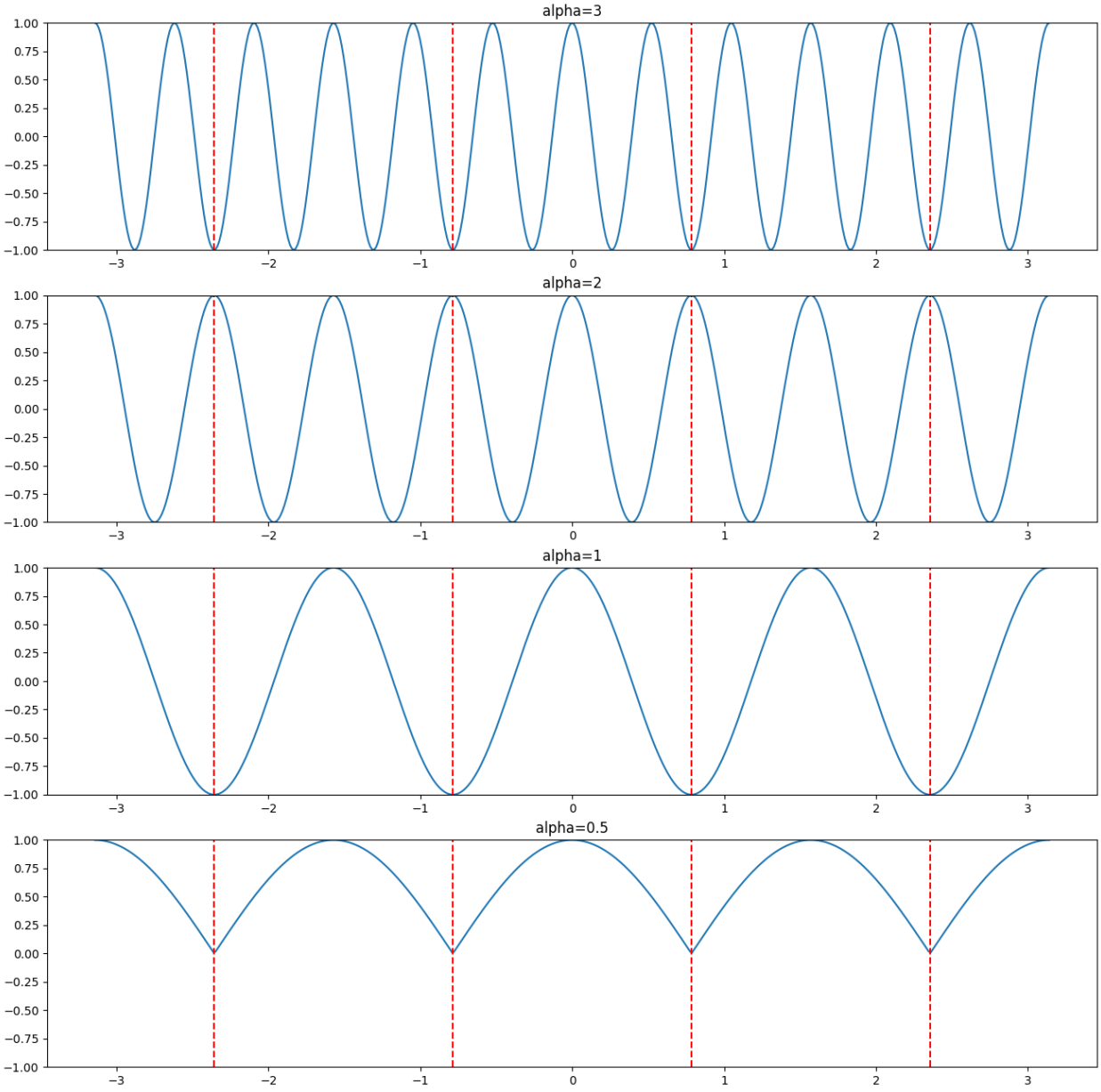


Figure 3 integer power property

Therefore, we now consider taking integer power only. Recall the set of the integer powers of an irrational angled value on the unit circle $\mathcal{S}^1$ form a dense subset of the unit circle $\mathcal{S}^1$ (Lemma 3.6).

Lemma 3.6. *For irrational number $\gamma \in \mathbb{R} \setminus \mathbb{Q}$, if $k \neq l, k, l \in \{0, 1, 2, \dots\}$, then $e^{ik\gamma} \neq e^{il\gamma}$. Also, the set $\mathcal{P}_\gamma := \{e^{ik\gamma} | k \in \{0, 1, 2, \dots\}\}$ is dense in $\mathcal{S}^1$.*

By the lemma 3.2, λ is an eigenvalue if and only if it is on the unit circle, by lemma 3.6, one can get a dense subset of the unit circle by taking integer power of an irrational eigenvalue where the integer powered eigenvalues are all different, and by lemma 3.5 the eigenfunctions are unique up to scalar multiplication. Hence, one can find one eigenpair $(e^{i\gamma}, \varphi_\gamma)$ with irrational angle γ and take integer power of them

$$\{(e^{ik\gamma\pi}, \varphi_\gamma^k) \mid k \in \{0, 1, 2, \dots\}\}$$

to get infinitely many eigenfunctions while their corresponding eigenvalues are all different, and therefore they are orthogonal to each other by lemma 3.3. Furthermore, they form a dense subset of all eigenvalues. Since all eigenvalues have a multiplicity of one, an integer-powered eigenpair represents the corresponding eigenspace. Note that the found eigenpairs are often not the eigenpair of the matrix representation, but it is still an eigenpair of the original Koopman operator, allowing us to obtain the eigenpair outside the projected space. Further, note, in a computer, that the irrational numbers are truncated and represented as rational numbers. Thus, the above dense subset produced by the integer powers of an irrational angled eigenvalue will not exactly produce a dense subset. Still, if the denominator of the rational representation is large enough, then the number of elements in the above set will be large enough. In detail, for an eigenvalue $e^{i2\pi\alpha}$ of the matrix representation, assume α is approximated by a rational number $q = \frac{n}{m}, m, n \in \mathbb{Z}$ where m, n are co-prime and $m \neq 0$, then one can obtain $m-1$ different eigenfunctions by taking powers $1, 2, \ldots, m-1$ since for $k \in \{1, 2, \ldots, m-1\}$, $qk \notin \mathbb{Z}$. (If $qk \in \mathbb{Z}$, then it contradicts that m, n are co-prime.) Since, in our setting, constant functions are always eigenfunctions, one can also take the power 0. As a summary, we obtain the following algorithm 16, which we call the integer-exponential algorithm.

Algorithm 16 Eigenfunction reconstruction with integer-exponential

Require: $K \in \mathbb{C}^{r\times r}$ is the matrix representation of Koopman operator U on the basis $\{\phi_i\}_{i=1}^r$, some small number ε (= 10^{-9} by default), number of dictionaries r, maximum power p_{max}.
Ensure: Eigenfunctions $\Phi := \{1, \varphi, \varphi^2, \ldots, \varphi^{p_{max}}\}$.
  $v \leftarrow$ Inverse iteration(K, α)
  $k \leftarrow 1$
  $\varphi \leftarrow \sum_{i=1}^r (v)_i \phi_i$
  $\Phi \leftarrow \{1, \varphi, \varphi^2, \ldots, \varphi^{p_{max}}\}$

Furthermore, remember that the products of the eigenfunctions are also eigenfunctions, and it is natural to guess that having more "true" eigenvectors will produce more eigenfunctions with small errors. Thus, we also developed the algorithm 17, which we call the integer-combination algorithm.

Algorithm 17 Eigenfunction reconstruction with integer-combinations

Require: $K \in \mathbb{C}^{r\times r}$ is the matrix representation of Koopman operator U on the basis $\{\phi_i\}_{i=1}^r$, some small number ε (= 10^{-9} by default), number of dictionaries r.
Ensure: Eigenfunctions $\Phi := \{\varphi^{p_1}, \varphi^{p_2}, \ldots\}$.
  $k \leftarrow 1$
  while $k \leq N$ **do**
    $v \leftarrow$ Inverse iteration(K, α_k)
    $\varphi_k \leftarrow \sum_{i=1}^r (v)_i \phi_i$
    $k \leftarrow k + 1$
  end while
  $\Phi \leftarrow \{1, \varphi_1, \varphi_2, \ldots, \varphi_N, (\varphi_1\varphi_2), \ldots, (\varphi_1^{p_1}\varphi_2^{p_2}\ldots\varphi_N^{p_N}), \ldots\}$

Note that in this case, one should set α_k to be not only irrational but also rationally independent, i.e., $\frac{\alpha_k}{\alpha_j} \in \mathbb{R} \setminus \mathbb{Q}$ so that the combination of the eigenvalues does not coincide with the already existing eigenvalues to avoid unnecessary effort. Also note that for large power, the error increases drastically.

Also, recall that in most cases, the integer power eigenvalues are not eigenvalues of the matrix representation, as infinitely many eigenvalues exist that are orthogonal to each other. The integer exponential algorithm will speed up the eigensolver only if there exists some eigenvalues that are integer exponential of the other eigenvalue. An example of such a system is, for instance, given by Bevanda, Sosnowski, and Hirche [BSH21]. For the state space $\mathcal{M} := \mathbb{R}^2$, define the evolution function as

$$T(\mathbf{x}) = \begin{pmatrix} ax_1 \\ bx_2 + (b-a^2)x_1^2 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathcal{M},$$

where $a, b \in [0,1]$. As is introduced in the paper, take the following dictionaries of observables $\phi_1(\mathbf{x}) = x_1, \phi_2(\mathbf{x}) = x_2 + x_1^2, \phi_3(\mathbf{x}) = x_1^2 = \phi_1^2(\mathbf{x})$ and define vector-valued observables $F = \begin{pmatrix}\phi_1 & \phi_2 & \phi_3\end{pmatrix}^T$. Then,

$$[UF](\mathbf{x}) = \begin{pmatrix}[U\phi_1](\mathbf{x}) \\ [U\phi_2](\mathbf{x}) \\ [U\phi_3](\mathbf{x})\end{pmatrix} = \begin{pmatrix}a\phi_1(\mathbf{x}) \\ b\phi_2(\mathbf{x}) \\ a^2\phi_3(\mathbf{x})\end{pmatrix} = \begin{pmatrix}a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & a^2\end{pmatrix} F(\mathbf{x}).$$

Thus, ϕ_1, ϕ_2, ϕ_3 are all the eigenfunctions of this system with corresponding eigenvalues a, b, a^2, respectively. Therefore, in this case, the integer-exponential algorithm can be used for eigensolver, and it requires 2 inverse iteration, not 3. Furthermore, based on these results, one can consider a special setting where the dictionaries of observables are all eigenfunctions and the integer power of an eigenfunction, i.e., the dictionaries of observables is $\{\varphi, \varphi^2, \varphi^3, \dots\}$ with corresponding eigenvalues $\{\lambda, \lambda^2, \lambda^3, \dots\}$. Then, the eigenvalues of the matrix representation are exactly $\{\lambda, \lambda^2, \lambda^3, \dots\}$, thus the integer-exponential algorithm for eigensolver requires just one inverse iteration and the rest can be obtained through integer-exponential of an eigenfunction. Note, however, that this is a special case. Therefore, we generally consider using the integer-exponential algorithm not for the eigensolver of DMD-type algorithms but for finding eigenfunctions whose corresponding power vectors are not eigenvectors of the matrix representation K.

3.3 Error calculation criteria

Before we come to the numerical experiments part, we consider the way to measure the reconstruction error of our algorithms. For the real-exponential algorithm (Algorithm 14), which is designed for replacing the eigenvectors reconstruction, we check if the power vector $v_1^{(p_k)}$ is truly an eigenvector corresponds to the same eigenvalue or not. Note that we normalize the eigenvectors and thus $\left\|v_1^{(p_k)}\right\| = \|v_k\| = 1$. Also recall by Lemma 3.5, the eigenvectors corresponding to different eigenvalues are all orthogonal to each other, and eigenvectors corresponding to the same eigenvalues are unique up to complex scalar multiplication. In other words, for $v_1^{(p_k)} = \alpha v_k$ for $\alpha \in \mathbb{C}$, if the reconstruction is correct, then $|\alpha| = 1$ and if not, $\alpha = 0$. Thus, the absolute value of

$$\langle v_1^{(p_k)}, v_k\rangle = \langle \alpha v_k, v_k\rangle = \alpha\langle v_k, v_k\rangle = \alpha$$

should be close to 1 if the reconstruction is correct and be close to 0 if they are orthogonal. Therefore, we can calculate the reconstruction error by the following criterion (lemma 3.7), which we call the power vector error of the real-exponential algorithm and denote it as E_1.

Lemma 3.7. *(real-power vector error of real-exponential algorithm) For eigenvector reconstruction $v_1^{(p_k)}$ by the real-exponential algorithm (algorithm 14) and the true eigenvector v_k both corresponding to the same eigenvalue, define*

$$E_1(v_1^{(p_k)}, v_k) := 1 - |\langle v_1^{(p_k)}, v_k\rangle| \in [0,1],$$

then which measures the error of the power vector given by the real-exponential algorithm.

If one has direct access to the explicit form of the eigenfunction with respect to an eigenvalue, for the algorithms of eigenfunction reconstruction (algorithm 15 and algorithm 16), we can check the norm of the functional difference between the reconstructed eigenfunction $\varphi_1^{p_k}$ and the true eigenfunction φ_k, i.e.,

$$\|\varphi_1^{p_k} - \varphi_k\|_2^2 = \int_{\mathcal{M}} (\varphi_1^{p_k}(x) - \varphi_k(x))\overline{(\varphi_1^{p_k}(x) - \varphi_k(x))} d\mu(x).$$

To numerically calculate the functional inner products, we can consider, for example, the Newton-Cotes formulas,

$$\int_a^b f(x)dx = \frac{b-a}{n}\sum_{k=1}^{n} f(\frac{b-a}{n}k).$$

However, since reconstructed eigenfunction may include complex scalar multiplication α, one should instead consider

$$\|\varphi_1^{p_k} - \alpha\varphi_k\|_2^2.$$

One way to determine α is by taking the average of the ratio

$$\alpha := \frac{1}{N}\sum_{i=1}^{N}\frac{\varphi_1^{p_k}(x_i)}{\varphi_k(x_i)},$$

with some x_i such that $\varphi_k(x_i) \neq 0$. Thus, if we have direct access to the explicit form of the eigenfunction with respect to an eigenvalue, we can calculate the eigenfunction reconstruction error by the following criterion (lemma 3.8), which we call power function reconstruction error and denote it as E_2.

Lemma 3.8. *(power function reconstruction error criterion) For eigenfunction reconstruction $f_1^{(p_k)}$ by the algorithm 15 or the algorithm 16, and the true eigenfunction φ_k both corresponding to the same eigenvalue, define*

$$E_2(f_1^{(p_k)}, f_k) := \|\varphi_1^{p_k} - \alpha\varphi_k\|_2^2$$

with

$$\alpha = \frac{1}{N}\sum_{i=1}^{N}\frac{\varphi_1^{p_k}(x_i)}{\varphi_k(x_i)},$$

then which measures the error of the power function.

However, we can not always have direct access to the explicit form of the eigenfunction. Remember that the power functions given by the algorithm 15 and 16 should be an eigenfunction of the Koopman operator U. Therefore, we can consider another criterion of power function, which checks the eigenfunction property

$$Uf = \lambda f \Leftrightarrow (U - \lambda I)f = 0.$$

Since we are given the snapshot of the state information dataset X, Y with $y_i = T(x_i)$ for $y_i \in Y, x_i \in X$, we have $[Uf](x_i) = f(T(x_i)) = f(y_i)$. Thus, we can calculate the average error of it by

$$\frac{1}{N}\sum_{k=1}^{N}\|[Uf](x_k) - \lambda f(x_k)\|_2^2 = \frac{1}{N}\sum_{k=1}^{N}\|f(y_k) - \lambda f(x_k)\|_2^2.$$

Thus, we have the following criterion (lemma 3.9), which we call eigenfunction property error and denote as E_3.

Lemma 3.9. *(eigenfunction property error) For eigenfunction reconstruction $f_1^{(p_k)}$ by the algorithm 15 or the algorithm 16 and the corresponding eigenvalue λ_k, define*

$$E_3(f_1^{(p_k)}) := \frac{1}{N}\sum_{k=1}^{N}\|f_1^{(p_k)}(y_k) - \lambda_k f_1^{(p_k)}(x_k)\|_2^2,$$

then which measures the error of the eigenfunction property of the power function.

Lastly, recall the purpose of obtaining eigenvalues, eigenfunctions, and Koopman modes is obtaining the underlying dynamical system's information, especially reconstructing the trajectory by the identity

$$[U^n F](x) = \begin{pmatrix}[U^n f_1](x)\\ \vdots \\ [U^n f_K](x)\end{pmatrix} = \begin{pmatrix}\sum_j \lambda_j^n v_{j1}\phi_j(x)\\ \vdots \\ \sum_j \lambda_j^n v_{jK}\phi_j(x)\end{pmatrix} = \sum_j \lambda_j^n \phi_j(x)\begin{pmatrix}v_{j1}\\ \vdots \\ v_{jK}\end{pmatrix}.$$

Recall in mpEDMD, given inputs

$$X = (\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_M}), Y = (\mathbf{y_1}, \mathbf{y_2}, \cdots, \mathbf{y_M}), \mathbf{y_k} = T(\mathbf{x_k}), \mathcal{D} = \{f_1, f_2, \cdots, f_N\}, f_k \in \mathcal{F},$$

define

$$\Phi(x) := \begin{pmatrix} f_1(x) & \dots & f_N(x) \end{pmatrix} \in \mathbb{C}^{1\times N}, \Phi_X := \begin{pmatrix} \Phi(x_1) \\ \vdots \\ \Phi(x_M) \end{pmatrix} \in \mathbb{C}^{M\times N}, \Phi_Y := \begin{pmatrix} \Phi(y_1) \\ \vdots \\ \Phi(y_M) \end{pmatrix} \in \mathbb{C}^{M\times N},$$

and the approximation of for $g \in L^2(\mathcal{M}, \mu)$ and states $x^{(0)}, x^{(n)} \in \mathcal{M}, x^{(n)} = T^n(x^{(0)})$ is given by

$$g(x^{(n)}) \approx [\Phi(x^{(0)})V]\Lambda^n[V^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T],$$

where the first term $[\Phi(x^{(0)})V]$ is the approximation of eigenfunction, the second term Λ^n is the eigenvalues, and the last term $V^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T$ is the Koopman modes. Since the power vectors of the real-exponential algorithm (algorithm 14) approximate the corresponding eigenvectors, define

$$V_p := \begin{pmatrix} v_1^{(p_1)} & v_1^{(p_2)} & \dots & v_1^{(p_r)} \end{pmatrix},$$

and replace the eigenvector matrix V of the $g(x_n)$ approximation with mpEDMD by the real-power vector matrix V_p,

$$g(x^{(n)}) \approx [\Phi(x^{(0)})V_p]\Lambda^n[V_p^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T].$$

This suggests us to consider the function reconstruction error

$$\left\| g(x^{(n)}) - [\Phi(x^{(0)})V_p]\Lambda^n[V_p^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T] \right\|_2.$$

One can further consider averaging the reconstruction error along the trajectory to have stable results. Thus, for input $I_K = \{(x_1^{(0)}, n_1, g_1), (x_2^{(0)}, n_2, g_2), \dots, (x_K^{(0)}, n_K, g_K)\}$ where $x_k^{(0)} \in \mathcal{M}, n_k \in \mathbb{N}, g_k \in L^2(\Omega, \omega)$, define the reconstruction error as

$$\frac{1}{K}\sum_{k=1}^{K}\left\| g_k(x_k^{(n_k)}) - [\Phi(x_k^{(0)})V_p]\Lambda^{n_k}[V_p^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T] \right\|_2.$$

Thus, we obtain the following criterion (lemma 3.10), which we call function trajectory reconstruction error and denote as E_4. Note that this criterion can be used for not only the real-exponential algorithm (algorithm 14) but also the integer-exponential algorithm (algorithm 16) where the power vectors are given by the same way as the real-exponential algorithm, i.e., by algorithm 13. Also note that the function $g \in L^2(\mathcal{M}, \mu)$ is often set as the identity function $g(x) = x$ in EDMD and mpEDMD to obtain the original trajectory X, Y as they are given as the inputs.

Lemma 3.10. *(Function trajectory reconstruction error) For power vectors* $V_p = \begin{pmatrix} v_1^{(p_1)} & v_1^{(p_2)} & \dots & v_1^{(p_r)} \end{pmatrix}$ *and state, power, and function tuples* $I_K = \{(x_1^{(0)}, n_1, g_1), (x_2^{(0)}, n_2, g_2), \dots, (x_K^{(0)}, n_K, g_K)\}$*, define*

$$E_4(I_K, V_p) := \frac{1}{K}\sum_{k=1}^{K}\left\| g_k(x_k^{(n_k)}) - [\Phi(x_k^{(0)})V_p]\Lambda^{n_k}[V_p^{-1}(\sqrt{W}\Phi_X)^\dagger\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M) \end{pmatrix}^T] \right\|_2.$$

then which measures the Koopman mode decomposition error of function $g \in L^2(\Omega, \omega)$ *with the power vectors. This criterion can be used for power vectors that are not eigenvectors of matrix representation, but the corresponding power function is the eigenfunction of the Koopman operator* U*.*

3.4 Numerical examples

Now, we show the numerical experimental results of our algorithms. Note the algorithms are implemented by using Python 3.9 with main libraries NumPy 1.21.6 [Har+20] and SciPy 1.7.3 [Vir+20]. As explained, we use mpEDMD to obtain the matrix representation of the Koopman operator. Note that the original

mpEDMD is implemented with Matlab (see Colbrook [Col23] and their GitHub repository `https://github.com/MColbrook/Measure-preserving-Extended-Dynamic-Mode-Decomposition`). Therefore, there are some differences in how functions work. The following is a list of important differences I noticed:

- Eigendecomposition function in NumPy (numpy.linalg.eig) returns a one-dimensional array of eigenvalues, whereas in Matlab (eig) returns a matrix (two dimensional) whose diagonal elements are eigenvalues and 0 elsewhere.
- Singular Value Decomposition (SVD) function in NumPy (numpy.linalg.svd) returns U, S, V_h where USV_h equals to the given matrix (one does not need to take the conjugate transpose of V_h), whereas in Matlab (svd) returns U, S, V where USV^* is equals to the given matrix (one needs to take the conjugate transponse of V to reconstruct the original matrix).
- There are some operand differences. For example, to get the conjugate transpose, NumPy needs to call the $.T$ operator and the $.conj()$ operator, whereas Matlab has its operand $'$ for a matrix. Also, there are some other syntax differences between these languages, but they can be interpreted one by one.

The original mpEDMD accepts having different weights W for the quadrature rule. However, we set them as an identity matrix and treat them equally. Also, they calculate the inverse of eigenvectors of the Hermitian matrix G by the conjugate transpose as it should be unitary. Note that in this setting, G is defined as

$$G := \frac{1}{M} \sum_{m=1}^{M} \Psi(x_m)^* \Psi(x_m).$$

However, this calculation shows worse performance than simply using *numpy.linalg.inv* function, in the sense of if eigenvalues lie on the unit circle. Therefore, we use *numpy.linalg.inv* function for obtaining the inverse of G. For more detailed code, check my GitHub repository `https://github.com/skyhajime/master_thesis`.

Now, for reconstructing the matrix representation of the Koopman operator from data with mpEDMD, we assume we obtain the snapshot of the state evolution, i.e., the ergodic sampling of the states such that

$$x_{i+1} = T(x_i) = y_i, x_i \in X, y_i \in Y.$$

To numerically calculate the inner products, especially integral, we use the Newton-Cotes formulas with the same weight for all values:

$$\int_a^b f(x)dx = \frac{b-a}{n} \sum_{k=1}^{n} f(\frac{b-a}{n}k).$$

3.4.1 Irrational rotation on unit circle

One of the most simple examples of the ergodic system is the irrational rotation on the unit circle where the state space $\mathcal{M}$ and evolution function $T : \mathcal{M} \to \mathcal{M}$ are defined as

$$\begin{aligned} &\mathcal{M} := [0, 1), \\ &T(x) = x + w \pmod 1, w \in \mathbb{R} \setminus \mathbb{Q}. \end{aligned}$$

For details, check, for example, Budišić, Mohr, and Mezić [MB12]. Note that the evolution function T is invertible with the inverse $T^{-1}(x) = x - w \pmod 1$. As it is shown in the figure 4, one can observe the state $x \in \mathcal{M}$ as the point on the unit circle $\mathbb{T} = \mathbb{R}/2\pi\mathbb{Z}$ with an observable $e^{i2\pi x}$ where i is the imaginary unit. (Also 2π-periodic functions on $\mathbb{R}$.) It is well known that if w is rational, then every initial condition is periodic, and if it is irrational, then the trajectory starting from any initial condition densely fills $\mathcal{M}$. Note that this dynamical system preserves the Lebesgue measure and is ergodic.

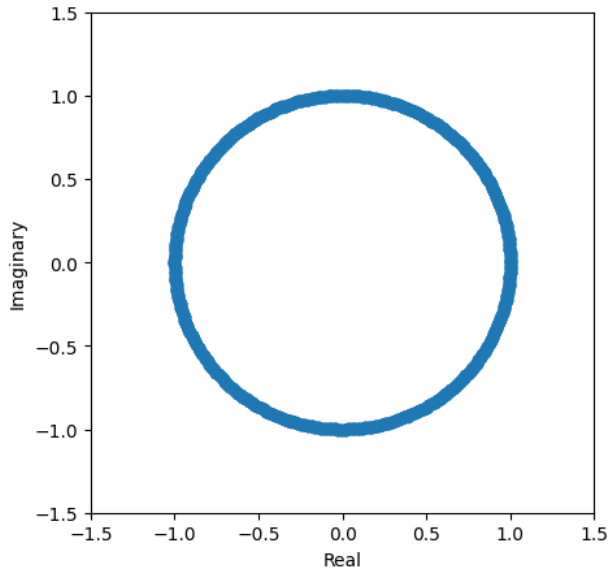


Figure 4 Irrational rotation on unit circle through an observable $e^{i2\pi x}$.

Now, let $\mathcal{F} := L^1(\mathcal{M}) = \left\{f : \mathcal{M} \to \mathbb{C} \mid \int_{\mathcal{M}} |f| d\mu < \infty\right\}$, the space of Lebesgue integrable $\mathbb{C}$-valued functions on $\mathcal{M}$, and consider observable $\varphi_n(x) = e^{i2\pi nx}$ with $n \in \mathbb{Z}$. Then

$$[U\varphi_n](x) = \varphi_n(T(x)) = e^{i2\pi n(x+w)} = e^{i2\pi nw} e^{i2\pi nx} = e^{i2\pi nw} \varphi_n(x).$$

Therefore, φ_n is an eigenfunction with eigenvalue $e^{i2\pi nw}$. Note that for $f_k(x) \in L^1(\mathbb{T})$, by Fourier transform, one can write $f_k(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k(n) e^{i2\pi nx}$ with Fourier coefficients $\hat{f}_k(n)$ and thus

$$[Uf_k](x) = \sum_{n\in\mathbb{Z}} \hat{f}_k(n) e^{i2\pi nT(x)} = \sum_{n\in\mathbb{Z}} \hat{f}_k(n) e^{i2\pi nw} \varphi_n(x).$$

This implies that the Fourier coefficients are the Koopman mode of the system with the Fourier basis $\{\varphi_k\}$. Remember that the Fourier basis is an orthonormal basis of $L^2([-\pi, \pi])$. Therefore, we set the Fourier basis as the dictionaries of the observables for the mpEDMD with degree d, i.e.,

$$\mathcal{D} := \{e_k : [-\pi, \pi] \to \mathbb{C} \mid e_k(x) = \frac{1}{\sqrt{2\pi}} e^{ikx}\}_{k\in\{\mathbb{Z}||k|\le d\}}.$$

Now, consider applying our algorithm to this setting. As for the parameters, we defined $w = \sqrt{2}/10$ as the irrational number, the initial state is $0 \in \mathcal{M}$, the dictionaries of the observables $\mathcal{D}$ are the finite length of the Fourier basis with maximum degree $d = 100$ which implies the number of dictionaries $r = 2d+1 = 201$, and the number of dataset $M = 10^4$. By applying mpEDMD, we obtain the unitary matrix representation $K \in \mathbb{C}^{r\times r}$ of the Koopman operator U, eigenvalues $\Lambda \in \mathbb{C}^r$ that are all lie on the unit circle, and the eigenvectors $V \in \mathbb{C}^{r\times r}$ which is approximated by Schur vectors.

power vector error E_1 of real-exponential algorithm First, we compare the eigenvector reconstruction error of the algorithm 14 by using the error $E_1(v_1^{(p_k)}, v_k)$. As it is introduced in Lemma 3.7, E_1 is defined as

$$E_1(v_1^{(p_k)}, v_k) := 1 - |\langle v_1^{(p_k)}, v_k\rangle| \in [0, 1].$$

Note that the algorithm 14 produces the power vectors $V^p \in \mathbb{C}^{r\times r}$, whose k-th column vector corresponds to $v_1^{(p_k)}$, which should coincide with the eigenvectors v_k calculated by the inverse iteration. Figure 5 is the scatter plot of the results of this experiment, where the x-axis represents the power value p_k and the y-axis represents the error $E_1(v_1^{(p_k)}, v_k)$.

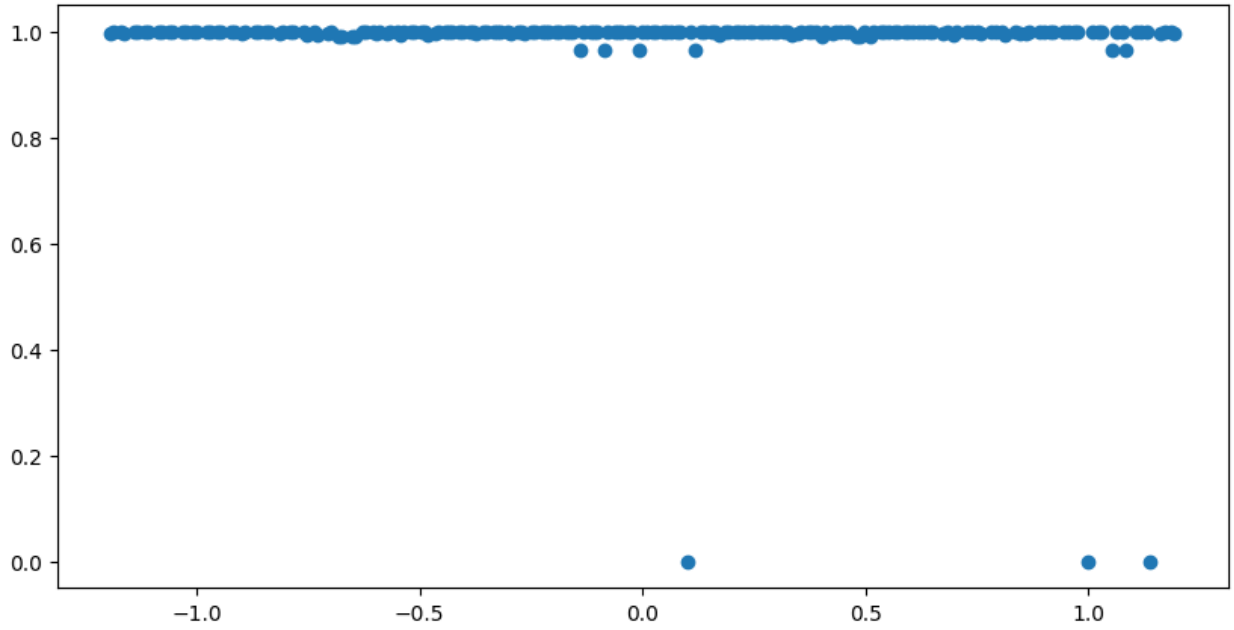


Figure 5 The E_1 error of the real-exponential vectors and the eigenvectors in the y-axis, where the x-axis is the corresponding power value.

One can observe the $E_1(v_1^{(p_k)}, v_k)$ errors are almost always 1, which means the power vector $v_1^{(p_k)}$ is almost always orthogonal to the desired eigenvector. It means that the algorithm 14 is failing to obtain the "true" eigenvector. One can hypothesize that the reason for this is the truncation of an irrational number in a computer. Recall that given an eigenvalue λ_1 (with corresponding eigenvector v_1) the k-th power p_k is defined so that $\lambda_1^{p_k} = \lambda_k$ holds. However, since the real number $p_k \in \mathbb{R}$ is represented as the truncated number $p'_k \in \mathbb{Q}$ in computer, they are not exactly the same, i.e., $p_k \neq p'_k$. Also, remember from Lemma 3.3 that the eigenfunctions of a normal operator are orthogonal to each other if the corresponding eigenvalues are different. Therefore, the truncation of the power produces a different eigenvalue from the desired value, and therefore, the power function is orthogonal to the desired eigenfunction, and projecting them onto the vector space will keep the orthogonality as it is introduced in the Lemma 3.4. As a clue of this, one can observe the power 1 produces the E_1 error 0 since the power 1 is for the first vector $\frac{\theta_1}{\theta_1}$ which is the first column vector of Schur decomposition and therefore is a "true" eigenvector. For the other two points having zero errors, we do not find a promising explanation because the index of these vectors is $[0, 82, 199]$ and the power values are $[1.0, 1.2442258436554148, -0.0019138144317270129]$. They are not integers nor any frequency ($\sqrt{2}/10 \approx 0.1414213562373095$) related values. Therefore, we further conducted the same experiment with different settings, especially different frequencies $\sqrt{3}/10, \sqrt{5}/20, \sqrt{7}/20 \in [0, 1)$. These experiments showed the E_1 error is zero with the powers $[1.0, -0.1268923926307155, -0.9199383957078294]$, $[1.0, -0.9322282970998922, -0.5941234612333683]$, and $[1.0, -0.6491015381236813, 0.14910128206605464]$, respectively. Since the error always becomes 0 with two other powers except 1, it seems there exists some rule for it. However, these powers are not integer or frequency-related values. Therefore, we could not find any promising explanations for this phenomenon.

Eigenfunction reconstruction error E_2 of real-exponential algorithm Now, we check the eigenfunction reconstruction error of the real-exponential algorithm. For this, we compare the four functions, (1) the true eigenfunction given by the above formula, i.e., given eigenvalue $\lambda = e^{i2\pi nw}$, the corresponding eigenfunction φ is $\varphi(x) = e^{i2\pi nx} = (e^{i2\pi nw})^{x/w} = \lambda^{x/w}$, (2) the eigenfunction reconstruction by mpEDMD, i.e., eigenfunction reconstruction with Schur vectors, (3) the eigenfunction reconstruction with eigenvectors by inverse iteration (Algorithm 12), and (4) the eigenfunction reconstruction by the algorithm 15, i.e.,

the real-exponential from one eigenvector. The error is calculated by $E_2(f_1^{(p_k)}, f_k)$. As it is introduced in Lemma 3.8, which is defined as

$$E_2(f_1^{(p_k)}, f_k) := \|\varphi_1^{p_k} - \alpha\varphi_k\|_2^2, \alpha := \frac{1}{N}\sum_{i=1}^{N}\frac{\varphi_1^{p_k}(x_i)}{\varphi_k(x_i)}.$$

Note that we defined the N to be simply the length of the dataset, i.e., $N = M$. Also, in the unit circle rotation setting, the eigenfunction has the form $\varphi_k(x) = e^{i2\pi kx}$ and the eigenfunction is unique up to some complex scalar $\alpha \in \mathbb{C}$, which is normalized, i.e., $|\alpha| = 1$. Therefore, the scalar can be expressed as $\alpha = e^{i2\pi a}, a \in \mathbb{R}$, and it works as the translation of the eigenfunction, i.e.,

$$\alpha\varphi_k(x) = e^{i2\pi(kx+a)}.$$

Thus, one can focus on the frequency of the reconstructed eigenfunction when checking the plot of the functions.

We first show the plot of the function values. Figure 6 is the first 15 values of the four functions plotted on the unit circle because the "true" eigenfunction $\varphi_n(x) = e^{i2\pi nx}$ only takes values on the unit circle. Figure 7 is the four function value's evolution plotted with the domain $[-0.5, 0.5]$ because some of the eigenfunction's frequency is so high that the plot is difficult to see with the whole domain $[-\pi, \pi]$. Note that for both figures, each row is the four function values corresponding to the same eigenvalue. Also note that each function is called (1) "True", (2) "mpEDMD", (3) "QR+EIG", and (4) "QR+Power", respectively. In both figures, we can observe the eigenfunction reconstructed by the true eigenvector ("QR + EIG") often performs better than mpEDMD eigenfunction reconstruction ("mpEDMD") in the sense of frequency. The reason would be that there is little error for the eigenvectors with ("QR + EIG"), whereas the mpEDMD uses Schur vectors as eigenvectors. Also, the frequencies of the eigenfunctions by the real-exponential algorithm ("QR + Power") are similar to the true eigenfunction ("True"). However, their amplitudes decrease as the power closes to 0, as explained in the theoretical section. (Check figure 2.)

Figure 8 shows the E_2 errors of each eigenpair, where the eigenfunction reconstruction is given by (1) mpEDMD (Schur decomposition), (2) inverse iteration (Eigendecomposition), and (3) the real-exponential algorithm (algorithm 15). One can observe that the real-exponential algorithm, however, has a larger E_2 error than the other two algorithms, which means the real-exponential algorithm is, again, not able to obtain a more accurate approximation of the true eigenfunction than mpEDMD or QR algorithm + inverse iteration.

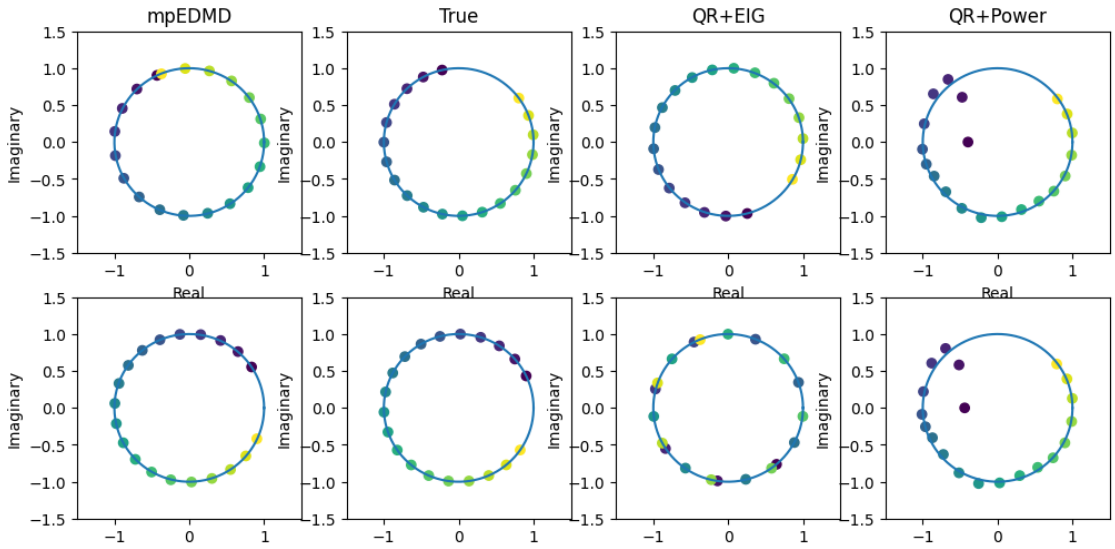


Figure 6 The function values on the unit circle where the functions are the reconstruction given by the formula ("True"), mpEDMD ("mpEDMD"), QR algorithm + inverse iteration ("QR+EIG"), and our real-exponential algorithm ("QR+Power").

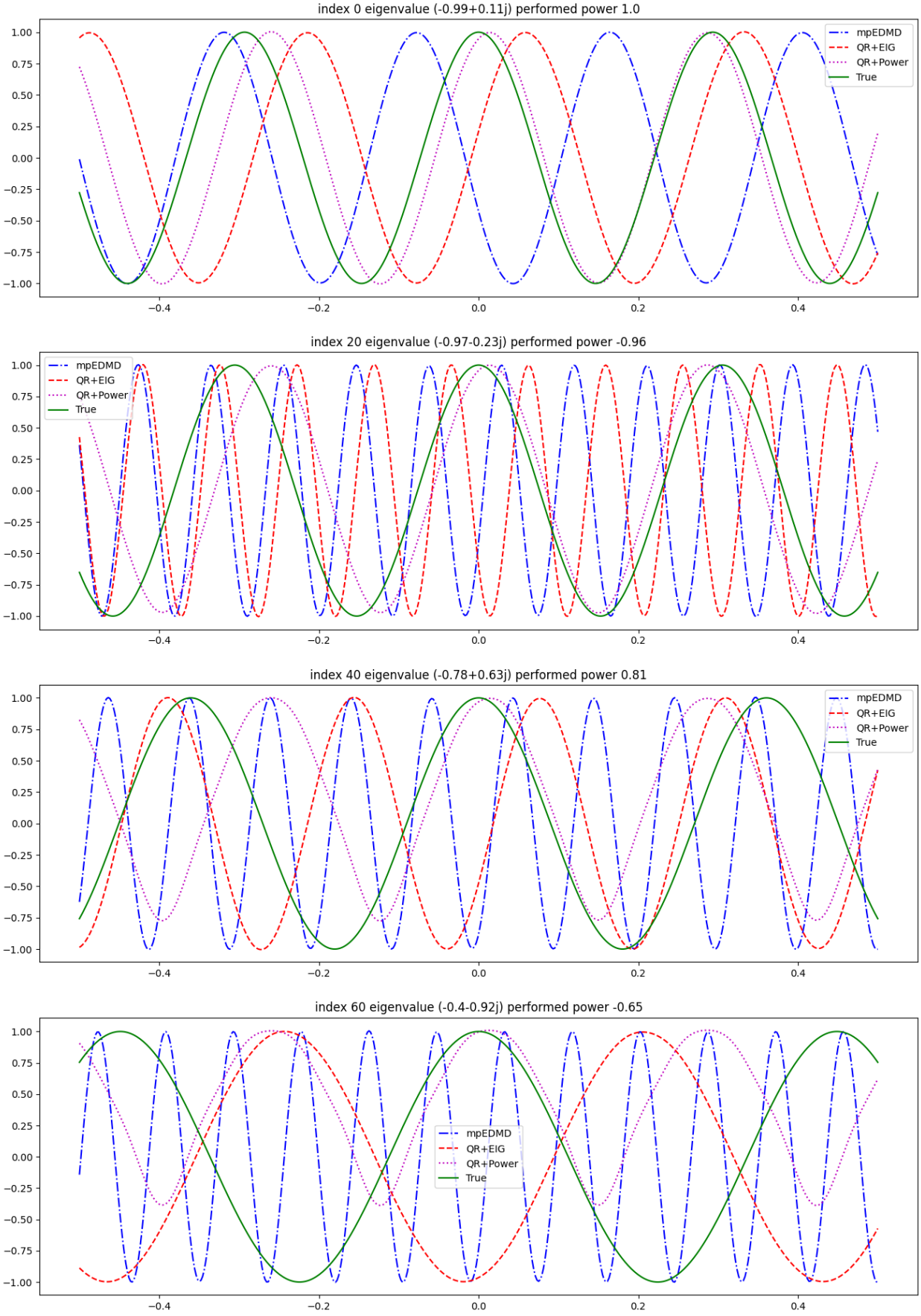


Figure 7 The function values evolution where the functions are the reconstruction given by the formula ("True"), mpEDMD ("mpEDMD"), QR algorithm + inverse iteration ("QR+EIG"), and our real-exponential algorithm ("QR+Power").

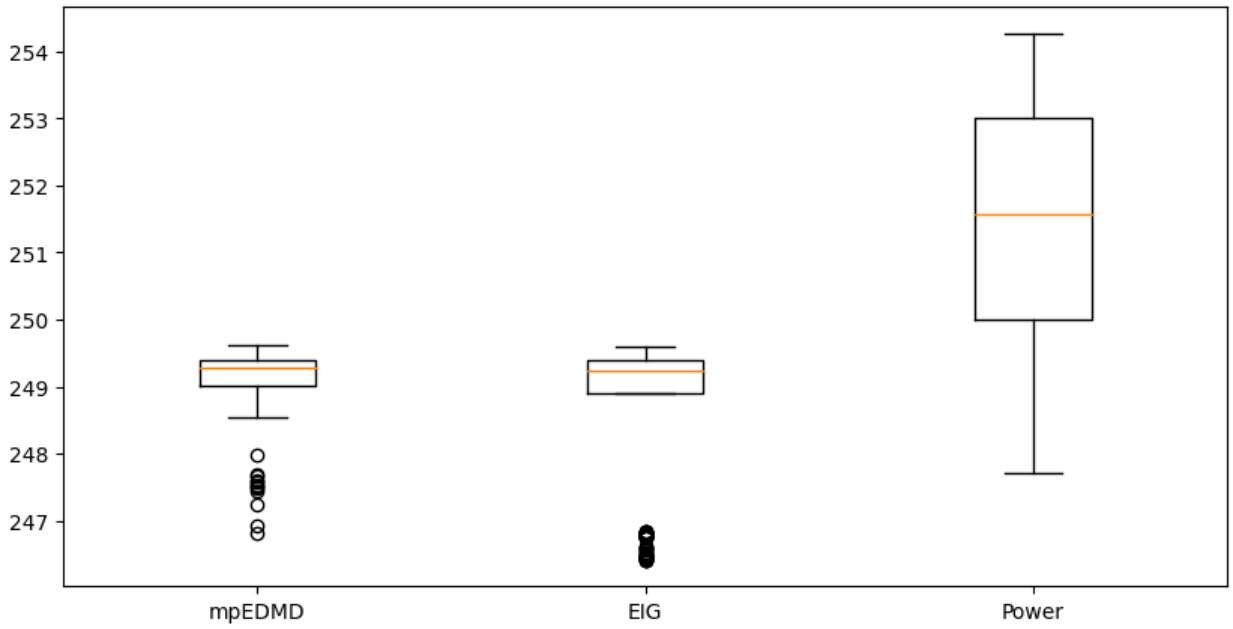


Figure 8 The E_2 error of the eigenfunction reconstruction given by mpEDMD ("mpEDMD"), QR algorithm + inverse iteration ("EIG"), and our real-exponential algorithm ("Power").

Eigenproperty error E_3 of integer-exponential algorithm Now, check if the power functions given by the integer-exponential algorithm (algorithm 16) satisfy the eigenfunction property by the error $E_3(f_1^{(p_k)})$. Recall from Lemma 3.9, the error $E_3(f_1^{(p_k)})$ is defined as

$$E_3(f) := \frac{1}{N}\sum_{k=1}^{N} \|f(y_k) - \lambda_k f(x_k)\|_2, y_k = T(x_k).$$

In this experiment, we simply defined N as the number of trajectory datasets, i.e., $N = M$ and E_3 is the average error over the whole trajectory.

The table 1 shows if the power eigenvalue $\lambda^p, p \in \mathbb{N}$ is an eigenvalue of the matrix representation K (the column "is eigenvalue of matrix"), and the E_3 error of the power function φ_1^p (the column "$E_3(\varphi_1^p)$ error") with corresponding power values. One can observe that the smaller the power, the smaller the E_3 error. The error is 0 if the power is 0. The reason for this is the constant function $1(x) := 1$ is an eigenfunction with eigenvalue 1 in this setting. Note that the power 1 is an eigenfunction reconstruction produced by the mpEDMD and has an error 0.000024. Moreover, even for the large power 20, the E_3 error is 0.000478, which is small, i.e., the power function given by the integer-exponential algorithm almost satisfies the eigenfunction property with eigenvalue λ^p. Also note that the figure 9 shows that E_3 error is almost proportional to the power, i.e., if power increases by 1, then the E_3 error increases by around 0.000024. Furthermore, the table 1 shows λ^p is not the eigenvalue of the matrix representation when $p \neq 1$. Note that the λ^p is considered as the eigenvalue of the matrix representation if there is an eigenvalue λ' of matrix representation that is given by the QR algorithm, such that the difference $|\lambda^p - \lambda'|$ is less than 10^{-5}, which is the default value of numpy.allclose function. This coincides with the necessary condition of the power vector to be an eigenvector of the matrix representation from theoretical results (Lemma 3.1) as it rarely happens that the integer-powered eigenvalues are also an eigenvalue of the matrix representation even allowing machine epsilon or pre-defined small precision difference.

In figure 10, we also plot the eigenfunction given by the algorithm 16 and compare it with the true eigenfunction and integer exponential of the true eigenfunction. Again, the plot domain is restricted to $[-0.1, 0.1]$ due to the high frequency. One can observe that even the original reconstruction "QR+Power" with power 1 does not coincide with the true eigenfunction due to reconstruction errors, which are the same for all mpEDMD, QR + inverse iteration, and our exponential algorithms. Also, even for the power of

the true eigenfunction, it does not coincide with the true eigenfunction. This will be due to the numerical error of the true eigenfunction calculation, as it includes irrational numbers, while the computer truncates irrational numbers to rational numbers, as already explained.

Table 1 The results of E_3 error of the power function given by integer-exponential algorithm (algorithm 16). The eigenvalue λ^p given by the integer-exponential algorithm is considered to be the eigenvalue of the matrix if there exists an eigenvalue λ' of the matrix s.t. $|\lambda^p - \lambda'| < 10^{-5}$.

power value	is eigenvalue of matrix	E_3 error
0	FALSE	0.00E+00
1	TRUE	2.67E-04
2	FALSE	5.33E-04
3	FALSE	7.98E-04
4	FALSE	1.06E-03
5	FALSE	1.33E-03
6	FALSE	1.59E-03
7	FALSE	1.85E-03
8	FALSE	2.12E-03
9	FALSE	2.38E-03
10	FALSE	2.64E-03
11	FALSE	2.90E-03
12	FALSE	3.16E-03
13	FALSE	3.42E-03
14	FALSE	3.68E-03
15	FALSE	3.94E-03
16	FALSE	4.20E-03
17	FALSE	4.46E-03
18	FALSE	4.72E-03
19	FALSE	4.98E-03
20	FALSE	5.24E-03

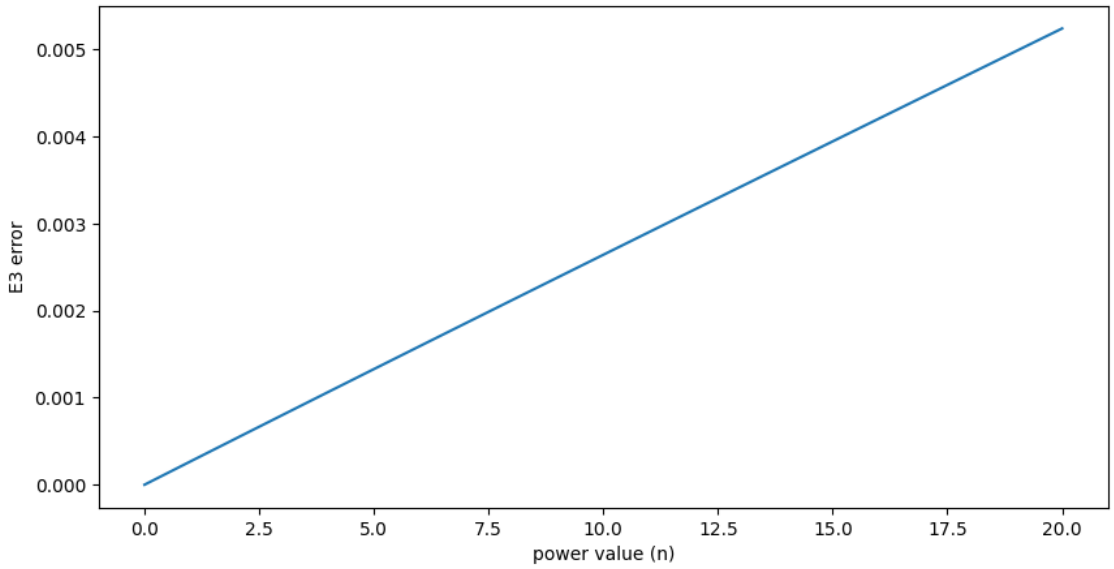


Figure 9 The plot of E_3 error of the power function given by integer-exponential algorithm, where the x-axis represents the power values and the y-axis represents the E_3 values.

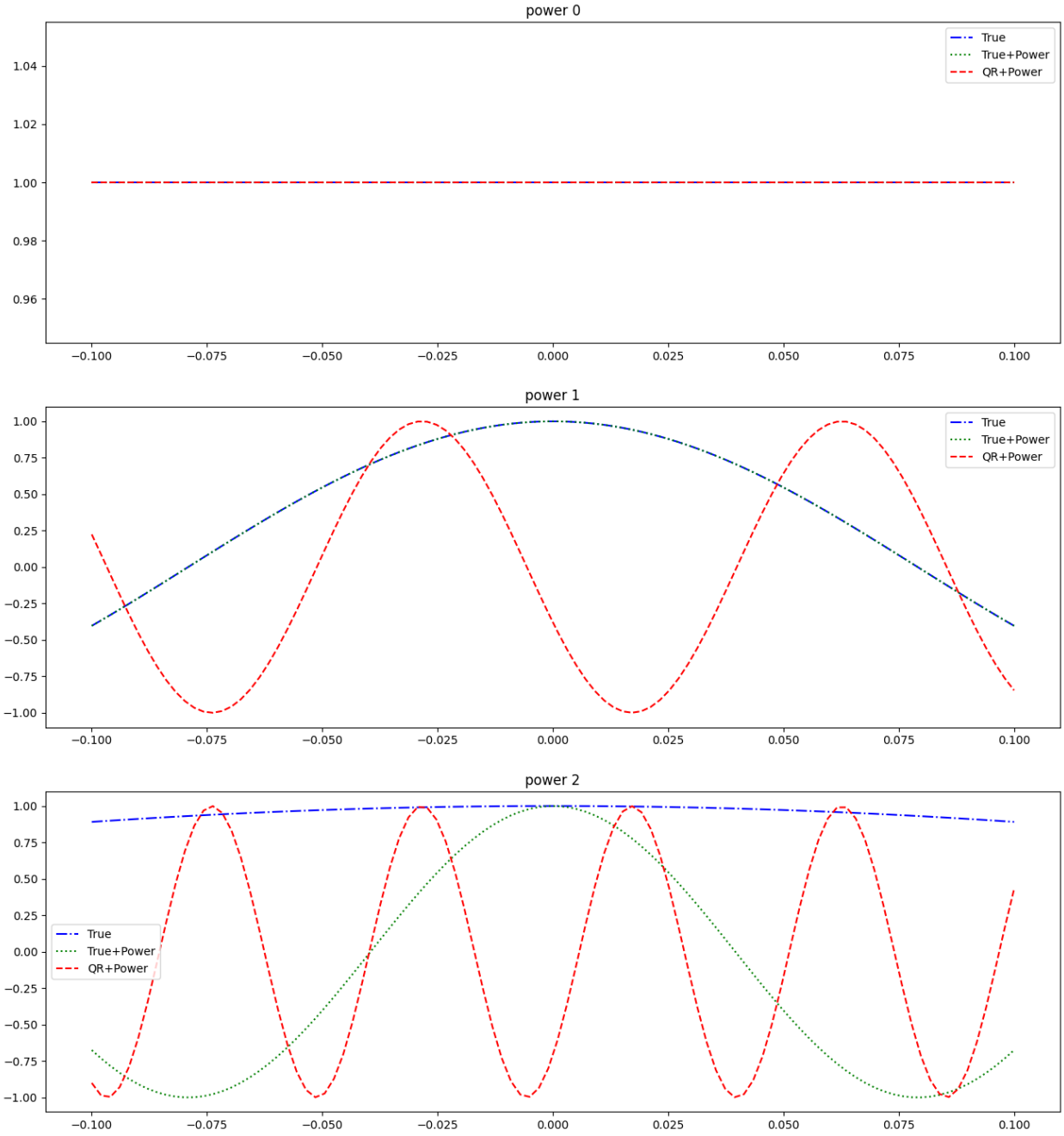


Figure 10 The function values evolution where each row corresponds to an eigenvalue with shown power. "True" represents the eigenfunction given by the formula, "True+Power" represents the function given by taking the power of the true eigenfunction of the original eigenvalue, and "QR+Power" represents the function given by the integer-exponential algorithm.

Original trajectory reconstruction error E_4 by integer-exponential algorithm Lastly, we check the original trajectory reconstruction error of the integer-exponential algorithm 16 by using Lemma 3.10. Recall that the reconstruction error of function $g \in L^2(\Omega, \omega)$ is defined as

$$E_4(I_K, V_p) := \frac{1}{K}\sum_{k=1}^{K}\left\| g_k(x_k^{(n_k)}) - [\Phi(x_k^{(0)})V_p]\Lambda^{n_k}[V_p^{-1}(\sqrt{W}\Phi_X)^{\dagger}\sqrt{W}\begin{pmatrix} g(x_1) & \dots & g(x_M)\end{pmatrix}^T]\right\|_2,$$

where $I_K = \{(x_1^{(0)}, n_1, g_1), (x_2^{(0)}, n_2, g_2), \dots, (x_K^{(0)}, n_K, g_K)\}$ and $V_p = \begin{pmatrix} v_1^{(p_1)} & v_1^{(p_2)} & \dots & v_1^{(p_r)}\end{pmatrix}$. In this numerical experiment, we only consider reconstructing the original trajectory given as the inputs, and there-

fore, we define the state as the input data $x_k^{(0)} = x_1$, the function is full state observable $g_k(x) := x$. The power values are simply the integer power $n_k = k, \forall k \in \{1, 2, \ldots, K\}, K = M$. In total, we calculate the E_4 error as

$$E_4(V_p) := \frac{1}{M} \sum_{k=1}^{M} \left\| x_k - [\Phi(x_1)V_p]\Lambda^k [V_p^{-1}\Phi_X^\dagger X^T] \right\|_2 .$$

Table 2 shows the E_4 error of the integer exponential algorithm, and figure 11 shows the plot of the E_4 error with respect to the number of inverse iterations conducted during the integer-exponential algorithm (algorithm 17). Note that for a large number of iterations, the E_4 error should decrease as they have more access to the "true" eigenvectors. One can observe that there are cases where the error is larger even if the number of inverse iterations is higher, for example, where the number of inverse iterations is 21 or 181. Note that in the case where the number of inverse iterations is 201, this algorithm coincides with the eigenvalue decomposition with inverse iteration. Also, note that none of the cases outperform the mpEDMD reconstruction, nor are they even not competitive with the mpEDMD unless it coincides with the QR algorithm + inverse iteration.

Table 2 The results of E_4 errors on unit circle setting with respect to some number of inverse iterations.

number of inverse iteration	E_4 error
1	102368.5
21	455982.23
41	170.59
61	633.69
81	5752.7
101	3793.65
121	1237.7
141	245.29
161	204.02
181	13806.42
201	0.03
mpEDMD	1.5E-03

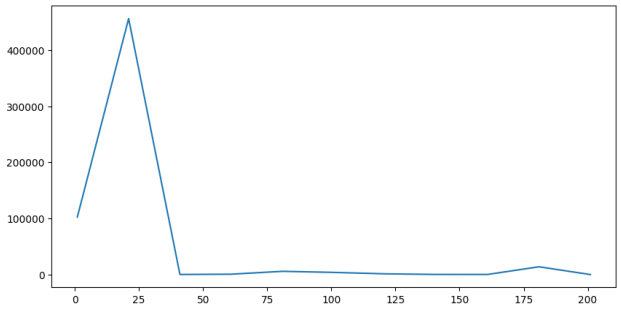


Figure 11 The plot of E_4 errors with tunit circle settings. The x-axis is the number of inverse iterations, and the y-axis is the E_4 errors.

3.4.2 Irrational flow on n-dimensional torus

Irrational flow on the n-dimensional flat torus is also an easy example of the ergodic system [CFS12]. For the case $n = 2$, consider for the state space $\mathcal{M} = [0, 1)^2$ and an irrational number $w_1, w_2 \in \mathbb{R} \setminus \mathbb{Z}$, the evolution function $T : \mathcal{M} \to \mathcal{M}$ as

$$T(\theta_1, \theta_2) = (\theta_1 + w_1 t, \theta_2 + w_2 t)^T \mod 1, t \in \mathbb{R}.$$

Again, the evolution function T is invertible with the inverse $T^{-1}(\theta_1, \theta_2) = (\theta_1 - w_1 t, \theta_2 - w_2 t)^T \mod 1$, and the system is ergodic. The evolution of the irrational flow on the flat torus can be viewed in figure 12 where the color indicates the time step of the data, i.e., the darker color plots are the n-th data with small n and the brighter color plots are n-th data with large n. Note that the states visit and explore the entire state space $\mathcal{M} = [0, 1)^2$, which represents the ergodicity of the system. Also, note that the two-dimensional flat torus can be viewed through the three-dimensional observable $f : \mathcal{M} = [0, 1)^2 \to \mathbb{R}^3$ defined as

$$f(\theta_1, \theta_2) = \begin{pmatrix} (1 + R\cos(2\pi\theta_2))\cos(2\pi\theta_1) & (1 + R\cos(2\pi\theta_2))\sin(2\pi\theta_1) & \sin(2\pi\theta_1) \end{pmatrix}^T .$$

Figure 13 shows the irrational flow on a two-dimensional torus through a three-dimensional observable.

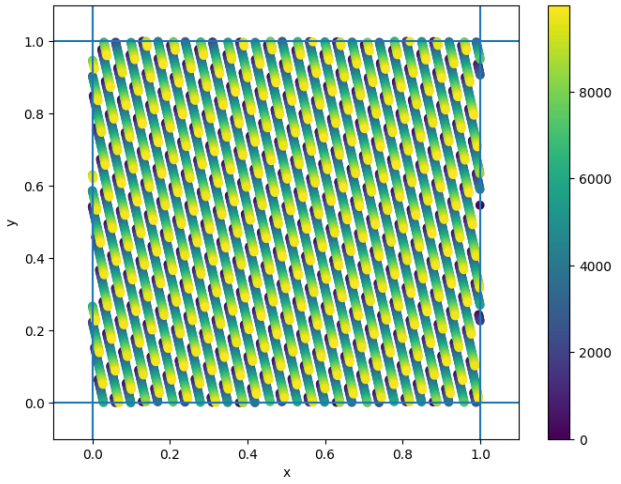


Figure 12 Irrational flow on 2D flat torus where the color represents the time step of the data.

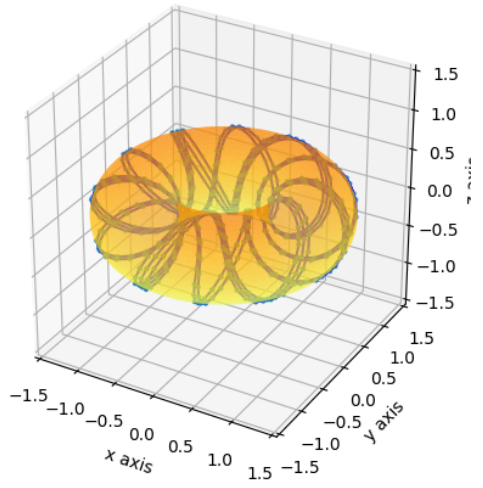


Figure 13 Irrational flow on 2D torus through 3D observable.

Now, for $k = (k_1, k_2), k_i \in \mathbb{Z}$ and $\theta = (\theta_1, \theta_2), \theta_i \in \mathbb{R}$, consider, similar to the unit circle case, that an observable $\varphi_k(\theta) := e^{i2\pi(k_1\theta_1 + k_2\theta_2)}$, then

$$\begin{aligned}[U\varphi_k](\theta) &= \varphi_k(T(\theta)) = \varphi_{k_1,k_2}(\theta_1 + w_1 t, \theta_2 + w_2 t) = e^{i2\pi(k_1(\theta_1+w_1 t)+k_2(\theta_2+w_2 t))} \\ &= e^{i2\pi(k_1 w_1 + k_2 w_2)t} e^{i2\pi(k_1\theta_1+k_2\theta_2)} = e^{i2\pi(k_1 w_1+k_2 w_2)t}\varphi_k(\theta).\end{aligned}$$

Thus, φ_k is an eigenfunction with eigenvalue $e^{i2\pi(k_1 w_1+k_2 w_2)t}$. This can be extended to the n-dimensional case where the state space $\mathcal{M} := [0,1)^n$, the frequencies $w := (w_1, w_2, \ldots, w_n), w_i \in \mathbb{R} \setminus \mathbb{Q}$, the evolution function for $\theta := (\theta_1, \theta_2, \ldots, \theta_n), \theta_i \in \mathbb{R}$,

$$T(\theta) := (\theta_1 + w_1 t, \theta_2 + w_2 t, \ldots, \theta_n + w_n t) \mod 1, t \in \mathbb{R},$$

with the inverse $T^{-1}(\theta) = (\theta_1 - w_1 t, \theta_2 - w_2 t, \ldots, \theta_n - w_n t) \mod 1$, and for $k := (k_1, k_2, \ldots, k_n)$, the observable function

$$\varphi_k(\theta) := e^{i2\pi \sum_{i=1}^n k_i\theta_i},$$

is an eigenfunction with eigenvalue

$$\lambda_k = e^{i2\pi \sum_{i=1}^n k_i w_i t}.$$

Now, consider applying our algorithms to this setting. In the following numerical examples, as the dictionaries of the observables, we again take the Fourier basis

$$\{e_k : [-\pi, \pi]^n \to \mathbb{C} \mid e_k(x) = \frac{1}{\sqrt{2\pi}} e^{2\pi i \sum_j^n k_j x_j}\},$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathcal{M}, k = (k_1, k_2, \ldots, k_n), k_j \in \mathbb{Z}, |k_j| \leq d$ with maximal degree d. Note that since we take all possible combinations of $|k_j| \leq d$, with maximal degree d, the number of dictionaries is $(2d+1)^n$. Thus, even for $n = 3$ and $d = 10$, it is $11^3 = 1331$, which is already large enough. Also note that the number of dictionaries r coincides with the size of the matrix representation K, and inverse iteration requires $O(r^3)$ operation for calculating the inverse of the matrix. Therefore, we only consider the case where $n = 2$. We defined the number of data is $N = 10^5$ and the maximum degree is $d = 10$, which implies the number of dictionaries $r = (2d+1)^n = 21^2 = 441$. The initial state $p_0 \in \mathcal{M}$ and frequencies w_1, w_2 are randomly chosen and defined as $p_0 = (0.06908747, 0.65315924)$ and $w_1, w_2 = 0.77080411, 0.9886656$. Note that the ratio $\frac{w_1}{w_2}$ is $\frac{0.77080411}{0.9886656} = 0.7796408714938601$ and "as_integer_ratio()" function of Python returns the integer ratio is $0.7796408714938601 \approx \frac{722063162674695}{562949953421312}$, which is almost irrational and almost satisfies our requirements.

power vector error E_1 of the real-exponential algorithm In the same way as the unit circle case, we first compare the eigenvector reconstruction error of the real-exponential algorithm (algorithm 14) by using E_1 error (Lemma 3.7). The results are shown in the figure 14.

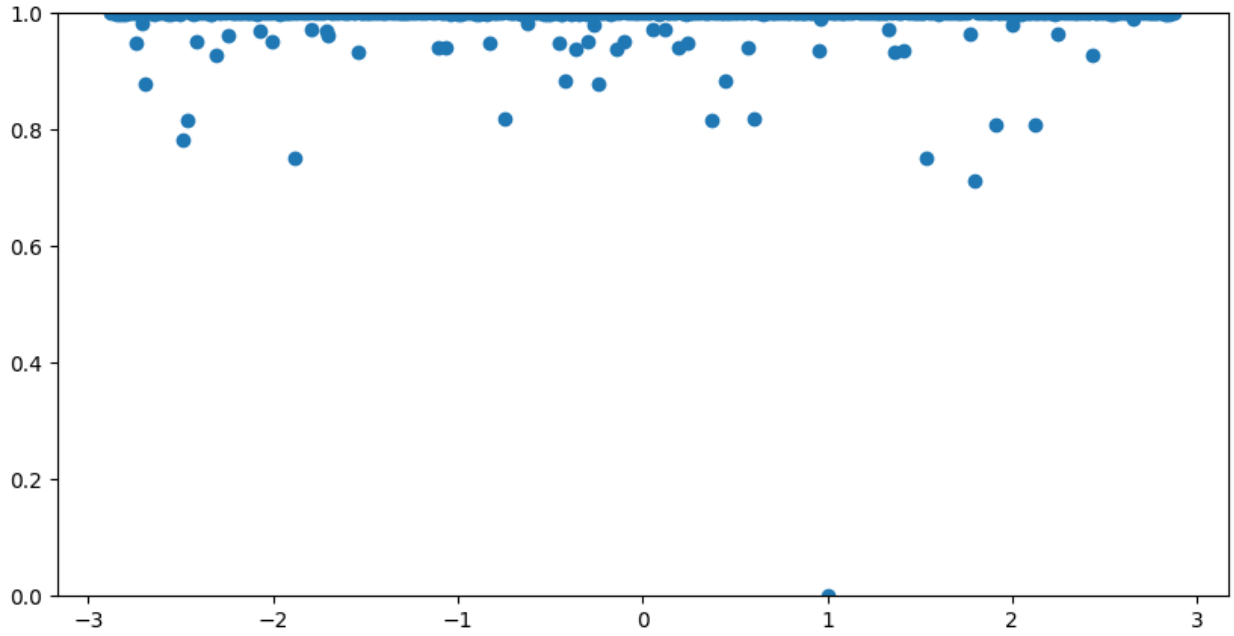


Figure 14 E_1 error of the irrational flow on 2D torus. The x-axis represents the power values, and the y-axis is the E_1 error.

On the one hand, similar to the one-dimensional case, the E_1 errors are almost 1 for all powers, except the case where power is one, which corresponds to the original vector given by the mpEDMD. This, again, indicates the power vectors given by the real-exponential algorithm are almost always orthogonal to the desired eigenvector, and it fails to obtain the true eigenvector. On the other hand, unlike the unit circle case, there are some cases that the E_1 error is between $[0, 1]$, especially around 0.8, and there is no other case where the E_1 error is zero except the power is one. This result supports the results of the one-dimensional case that the real-exponential algorithm cannot obtain the desired eigenfunction and corresponding eigenvector due to the rational approximation of irrational numbers on the computer and the orthogonality of the eigenfunctions with respect to the different eigenvalues. Also, we can hypothesize the results of the one-dimensional case, in which there exists three powers with E_1 error become zero, might be just a coincidence. The values, not zero nor one, can be understood as the limitation of the finite-dimensional approximation of the infinite-dimensional eigenfunction. In the infinite-dimensional case, the E_1 error should be 0 or 1. However, in the finite-dimensional case, it fails to obtain some information, which cancels some of the information in the projected space.

Eigenfunction reconstruction error E_2 of real-exponential algorithm To calculate the eigenfunction reconstruction error E_2, we need to calculate the true eigenfunction given eigenvalues. However, it is difficult to obtain such eigenfunction from eigenvalues due to the number of unknown variables and the combination of irrational numbers. Recall for the eigenvalue eigenvalue $e^{i2\pi(k_1w_1+k_2w_2)t}$, the corresponding eigenfunction is given by $\varphi_k(\theta) := e^{i2\pi(k_1\theta_1+k_2\theta_2)}$. Thus, given only an eigenvalue $\lambda \in \mathbb{T}$, one can not determine the $k = (k_1, k_2)$, and consequently, the eigenfunction φ_k. Therefore, we just plot the eigenfunction reconstruction evolution by (1) mpEDMD ("mpEDMD"), (2) QR algorithm and inverse iteration ("QR+EIG"), and (3) real-exponential algorithm (algorithm 15) ("Power") in figure 15 and ignore calculating "True" eigenfunctions and E_2 error of two-dimensional case. Similar to the one-dimensional case, the real-exponential algorithm finds eigenfunctions that are similar to the eigenfunction reconstruction given by

mpEDMD and QR algorithm + inverse iteration. However, it contains a real-exponential error of complex numbers, especially if the power is far from 1.

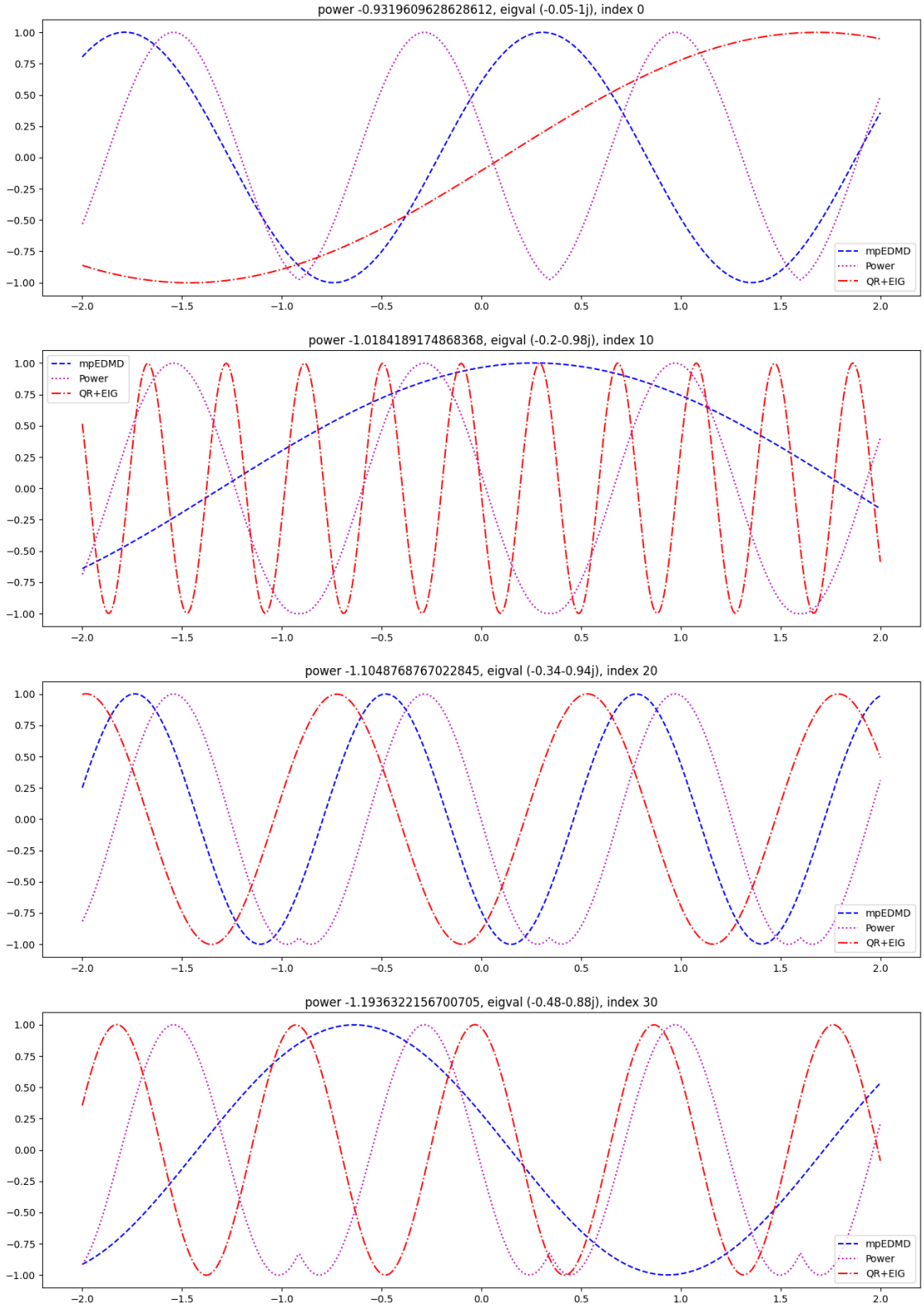


Figure 15 The eigenfunction reconstruction by the real-exponential algorithm (algorithm 15) of irrational flow on 2D torus.

Eigenfunction property error E_3 of integer-exponential algorithm We now check the eigenfunction property error E_3 (Lemma 3.9) of the integer-exponential algorithm (algorithm 16). E_3 errors are shown in the table 3 and the plot is in the figure 16. Even in the two-dimensional case, the integer-exponential algorithm is able to obtain the power functions that satisfy the eigenfunction property while the corresponding eigenvalues are not eigenvalues of the matrix representation. As shown in the table 3, the E_3 error is less than 0.001 (= 1.00E-03) even if the power is large (20). Also, one can observe in figure 16 that E_3 error increases linearly with respect to the power value. This means that once one obtains an eigenfunction with some error, one can obtain several eigenfunctions with eigenvalues while the E_3 error is easily predicted.

Table 3 The results of E_3 errors with two-dimensional setting.

power value	is eigenvalue of matrix	E_3 error
0	FALSE	0.00E+00
1	TRUE	4.49E-05
2	FALSE	8.97E-05
3	FALSE	1.35E-04
4	FALSE	1.79E-04
5	FALSE	2.24E-04
6	FALSE	2.69E-04
7	FALSE	3.14E-04
8	FALSE	3.59E-04
9	FALSE	4.04E-04
10	FALSE	4.49E-04
11	FALSE	4.94E-04
12	FALSE	5.38E-04
13	FALSE	5.83E-04
14	FALSE	6.28E-04
15	FALSE	6.73E-04
16	FALSE	7.18E-04
17	FALSE	7.63E-04
18	FALSE	8.08E-04
19	FALSE	8.53E-04
20	FALSE	8.98E-04

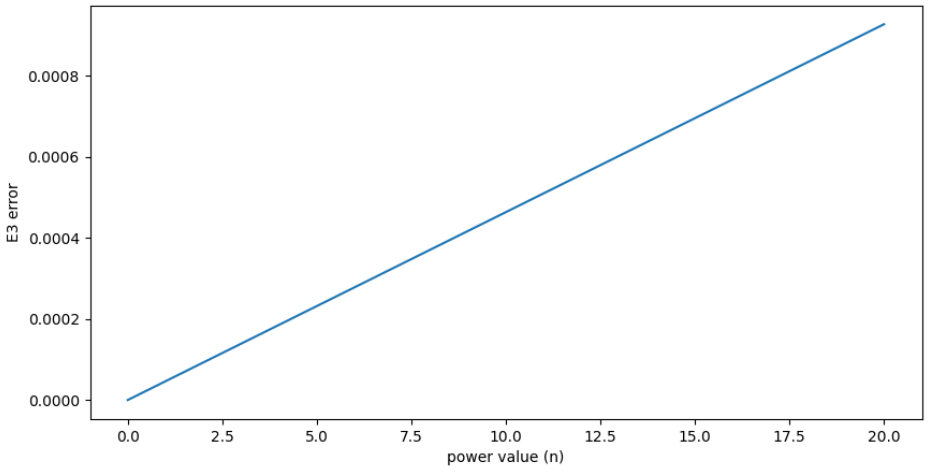


Figure 16 The plot of E_3 errors with two-dimensional setting. The x-axis is the power value (n in the table), and the y-axis represents the E_3 errors.

Original trajectory reconstruction error E_4 of integer-exponential algorithm Lastly, the original trajectory reconstruction error E_4 is shown in the table 4 and whose plot is in the figure 17. Note the number of iterations is simply chosen such that the total plot is 10, i.e., number of inverse iterations of the n-th plot is $1 + n \times r//10 = 1 + 44n$. Also note that in the case the number of inverse iterations is 441, it is the same as performing the QR algorithm and inverse iteration for all column vectors (algorithm 12).

Figure 17 shows that the E_4 error decreases as the number of inverse iterations increases although there are some irregular increases in between. This trend is natural as they have more access to the true eigenvectors. Note, however, that even if the number of inverse iteration is 397, which is quite close to performing inverse iteration for all column vectors, the E_4 error is high (2.6×10^{12}). This effect can be understood as due to their corresponding vectors' information not existing in the projected finite-dimensional space, which causes inaccurate eigenvectors reconstruction and the corresponding Koopman modes.

Table 4 The results of E_4 errors with two-dimensional setting.

number of inverse iteration	E_4 error
1	NaN
45	3.58E+15
89	2.58E+15
133	5.15E+15
177	1.48E+15
221	3.83E+15
265	6.93E+14
309	8.73E+14
353	5.23E+13
397	2.60E+12
441	1.13E-02

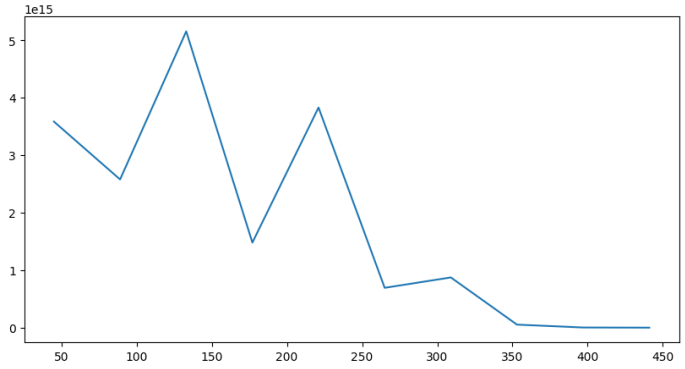


Figure 17 The plot of E_4 errors with two-dimensional setting. The x-axis is the number of inverse iterations and the y-axis is the E_4 errors.

4 Conclusion

Summary of our approach and results We introduced two algorithms: the real-exponential algorithm for finding eigenvectors of the matrix representation of the Koopman operator obtained by DMD-type algorithms and the integer-exponential algorithm for avoiding the real-exponential error of complex numbers and finding eigenfunctions of a Koopman operator outside the projected finite-dimensional space. We assumed the underlying dynamical system is ergodic and used the unique properties of the Koopman operator and ergodic systems, especially the property 2.1, which states that the combinations and exponentiations of some eigenfunctions form yet another eigenfunction.

The real-exponential algorithm is developed by the combination of the property 2.1 and the multiplicity of eigenvalues on the ergodic system for obtaining eigenvectors of the matrix representation of the Koopman operator and for accelerating the eigendecomposition process performed during the DMD-type algorithms. The power values $\{p_1, p_2, \ldots, p_n\}$ are defined such that the powered eigenvalues coincide with the true eigenvalues so that the power vectors with $\{p_1, p_2, \ldots, p_n\}$ satisfy the necessary condition of being the eigenvectors of the matrix representation. The multiplicity of eigenvalues of the Koopman operator ensured the eigenfunction corresponding to an eigenvalue is unique up to complex scalar multiplication, and consequently, the power vectors with $\{p_1, p_2, \ldots, p_n\}$ are truly eigenvectors of the matrix representation of the Koopman operator. However, the theoretical and numerical results showed that the real-exponential algorithms perform worse than the original mpEDMD or the true eigendecomposition due to the real-exponential error of complex numbers and the rationally truncated representation of irrational numbers on a computer.

The integer-exponential algorithm is introduced to avoid the real-exponential error of complex numbers. The set $\mathcal{P}_\gamma = \left\{e^{i\gamma n} \mid n \in \mathbb{N}\right\}$, which consists integer-powered values of an irrational angled value $e^{i\gamma}, \gamma \in \mathbb{R} \setminus \mathbb{Q}$, is dense subset of the unit circle (Lemma 3.6) and all values are different. This ensures the integer-exponential algorithm produces different eigenfunctions of the Koopman operator, and they are orthogonal to each other. The numerical study showed the integer-exponential algorithm successfully finds eigenfunctions that have small eigenfunction property errors (E_3 error) while the corresponding power vectors are not the eigenvectors of the matrix representation. The eigenfunction property error was small with a small integer power, and the error increases as the power value increases almost linearly. Moreover, even for relatively large power 20, the numerical results showed the error is still small (under 0.001). However, once we project such eigenfunctions onto the finite-dimensional space where the matrix representation is considered, the reconstruction error of the original trajectory (E_4 error) is large even if half of the eigenvectors are provided by the inverse iteration.

Discussion and outlook Based on the numerical and theoretical results, there will be a small room for improvement of the real-exponential algorithm (algorithm 14) since the property 2.1 of Koopman operator will not precisely work for complex case due to real power of complex numbers, and the relationship between the eigenvalues of the matrix representation will often be real power not integer. Furthermore, the real-exponential algorithm will not work if there exists any small number of differences due to the orthogonality of eigenfunctions corresponding to different eigenvalues (Lemma 3.3). Since the real numbers are truncated in the computer, there will almost always be some truncation error of reconstructed eigenvalues. Note, however, that if the observable function f is defined to be a real-valued function, namely, $f : \mathcal{M} \to \mathbb{R}$, then there is no real-exponential error, and the algorithm might work up to truncation error of the irrational number. Also, if it is possible to obtain the eigenfunction corresponding to an eigenvalue without any error, then the ergodicity assures the power functions are truly eigenfunctions and the power vector given by algorithm 13 will provide the corresponding eigenvector of the matrix representation.

On the contrary, the integer-exponential algorithm showed potential to be further investigated since there is no exponential error like the real-exponential case. The results showed the power functions given by the integer-exponential algorithm almost satisfy the eigenfunction property, i.e., E_3 errors were small. Although the trajectory reconstruction error (E_4 error) was large even with a large number of inverse iterations, it is likely that only the Koopman modes reconstruction is not working well since the other terms, eigenvalues and eigenfunctions, were ensured satisfying the eigenproperty correctly by the small E_3

errors. We interpret this phenomenon as natural since the Koopman modes are reconstructed by using the power vectors projected onto the finite-dimensional space where the information of these power functions (=eigenfunctions) is neglected.

Hence, one possible future research will be studying the integer-exponential algorithm for reconstructing the original trajectory. Since the cause of the large E_4 error of the integer-exponential algorithm might be the Koopman modes reconstruction error due to projecting the obtained functions onto the finite-dimensional space, one can consider an algorithm for obtaining the Koopman modes without projecting back to the finite-dimensional space. Since the Koopman modes are defined as the coefficients of the (full-state observable) function, one can obtain such value, for example, by using the inner product of the function and eigenfunctions obtained by the integer-exponential algorithm. If this is achieved, then one can approximate the information of the underlying dynamical system just by one (or a small number of) inverse iterations. Another possibility will be considering the case where the observable functions are real-valued. In this case, the real-exponential algorithm will work without suffering from the real-exponential error of complex numbers and possibly fasten the eigendecomposition process performed during the DMD-type algorithms. Lastly, the numerical results of E_1 error with unit circle showed there are some cases where the real-exponential algorithm finds "true" eigenvectors while the power is not 1 (figure 5). This phenomena is not fully understood yet since this did not happen with two-dimensional torus experiments (figure 14). Therefore, it can be further investigated.

In conclusion, the discoveries and contributions of this thesis highlighted both the potential and the limitations of the exponential algorithms concerning the enhancement of efficiency and quantity in computing the spectral information of the Koopman operator. Through continued exploration of the aforementioned future research topics and others, it is conceivable that the exponential algorithms may play a pivotal role in accelerating the eigendecomposition process or augmenting the quantities of spectral information associated with the Koopman operator.

References

[AM17] H. Arbabi and I. Mezić. "Ergodic Theory, Dynamic Mode Decomposition, and Computation of Spectral Properties of the Koopman Operator". In: *SIAM Journal on Applied Dynamical Systems* 16.4 (2017), pp. 2096–2126. eprint: https://doi.org/10.1137/17M1125236.

[Arn89] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. 2nd ed. Springer New York, NY, 1989.

[Arn51] W. Arnoldi. "The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem". In: *Quarterly of Applied Mathematics* 9 (1951), pp. 17–29.

[BSH21] P. Bevanda, S. Sosnowski, and S. Hirche. "Koopman operator dynamical models: Learning, analysis and control". In: *Annual Reviews in Control* 52 (2021), pp. 197–212. ISSN: 1367-5788.

[Bir31] G. D. Birkhoff. "Proof of the ergodic theorem". English. In: *Proc. Natl. Acad. Sci. USA* 17 (1931), pp. 656–660. ISSN: 0027-8424.

[BCS14] G. I. Bischi, C. Chiarella, and I. Sushko. *Global Analysis of Dynamic Models in Economics and Finance*. 1st ed. Springer Berlin, Heidelberg, 2014.

[Bor20] D. Borthwick. *Spectral Theory*. 1st ed. Springer Cham, 2020.

[BS83] A. Böttcher and B. Silbermann. "The finite section method for TOEPLITZ operators on the quarter-plane with piecewise continuous symbols". In: *Mathematische Nachrichten* 110.1 (1983), pp. 279–291. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mana.19831100120.

[BCC14] F. Brauer and C. Castillo-Chavez. *Mathematical Models in Population Biology and Epidemiology*. 2nd ed. Springer New York, NY, 2014.

[CTR12] K. K. Chen, J. H. Tu, and C. W. Rowley. "Variants of Dynamic Mode Decomposition: Boundary Condition, Koopman, and Fourier Analyses". In: *Journal of Nonlinear Science* 22.6 (2012), pp. 887–915.

[Chu08] M. T. Chu. "Linear algebra algorithms as dynamical systems". In: *Acta Numerica* 17 (2008), 1–86.

[Col23] 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.

[CFS12] I. P. Cornfeld, S. V. Fomin, and Y. G. Sinai. *Ergodic Theory*. 1st ed. Springer New York, NY, 2012.

[DTK20] F. Dietrich, T. N. Thiem, and I. G. Kevrekidis. "On the Koopman Operator of Algorithms". In: *SIAM Journal on Applied Dynamical Systems* 19.2 (2020), pp. 860–885. eprint: https://doi.org/10.1137/19M1277059.

[DFN85] B. Dubrovin, A. Fomenko, and S. Novikov. *Modern Geometry — Methods and Applications. Part II: The Geometry and Topology of Manifolds*. 1st ed. Springer New York, NY, 1985.

[ER85] J. P. Eckmann and D. Ruelle. "Ergodic theory of chaos and strange attractors". In: *Rev. Mod. Phys.* 57 (3 1985), pp. 617–656.

[GV00] G. H. Golub and H. A. van der Vorst. "Eigenvalue computation in the 20th century". In: *Journal of Computational and Applied Mathematics* 123.1 (2000), pp. 35–65.

[Gra86] W. B. Gragg. "The QR algorithm for unitary Hessenberg matrices". In: *Journal of Computational and Applied Mathematics* 16.1 (1986), pp. 1–8. ISSN: 0377-0427.

[Han22] A. Hannachi. *Patterns Identification and Data Mining in Weather and Climate*. 1st ed. Springer Cham, 2022.

[Har+20] C. R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.

[Hou58] A. S. Householder. “Unitary Triangularization of a Nonsymmetric Matrix”. In: *J. ACM* 5.4 (1958), 339–342. ISSN: 0004-5411.

[Jon01] C. K. R. T. Jones. “Whither Applied Nonlinear Dynamics?” In: *Mathematics Unlimited — 2001 and Beyond*. Ed. by B. Engquist and W. Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 631–645. ISBN: 978-3-642-56478-9.

[Koo31] B. O. Koopman. “Hamiltonian Systems and Transformation in Hilbert Space”. In: *Proceedings of the National Academy of Sciences* 17.5 (1931), pp. 315–318. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.17.5.315.

[KN32] B. O. Koopman and J. v. Neumann. “Dynamical Systems of Continuous Spectra”. In: *Proceedings of the National Academy of Sciences* 18.3 (1932), pp. 255–263. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.18.3.255.

[KPM20] M. Korda, M. Putinar, and I. Mezić. “Data-driven spectral analysis of the Koopman operator”. In: *Applied and Computational Harmonic Analysis* 48.2 (2020), pp. 599–629. ISSN: 1063-5203.

[Lan50] C. Lanczos. “An iteration method for the solution of the eigenvalue problem of linear differential and integral operators”. In: *Journal of Research of the National Bureau of Standards* (1950).

[Lan93] S. Lang. *Complex Analysis*. 3rd ed. Springer Berlin, Heidelberg, 1993.

[MK16] J. Mann and J. N. Kutz. “Dynamic mode decomposition for financial trading strategies”. In: *Quantitative Finance* 16.11 (2016), pp. 1643–1655. eprint: https://doi.org/10.1080/14697688.2016.1170194.

[MB12] I. M. Marko Budišić Ryan M. Mohr. “Applied Koopmanism”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 22 (Dec. 2012), p. 047510. ISSN: 1054-1500.

[Mez05] I. Mezić. “Spectral Properties of Dynamical Systems, Model Reduction and Decompositions”. In: *Nonlinear Dynamics* 41.1 (2005), pp. 309–325.

[Mus14] J. Muscat. *Functional Analysis*. 1st ed. Springer Cham, 2014.

[Par+23] H. Partamian et al. “Analysis of task-related MEG functional brain networks using dynamic mode decomposition”. In: *Journal of Neural Engineering* 20.1 (2023), p. 016011.

[ROW+09] C. W. ROWLEY et al. “Spectral analysis of nonlinear flows”. In: *Journal of Fluid Mechanics* 641 (2009), 115–127.

[SCH10] P. J. SCHMID. “Dynamic mode decomposition of numerical and experimental data”. In: *Journal of Fluid Mechanics* 656 (2010), 5–28.

[Sch66] P. H. Schönemann. “A generalized solution of the orthogonal procrustes problem”. In: *Psychometrika* 31.1 (1966), pp. 1–10.

[Sch18] J. Schur. “Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind.” In: *Journal für die reine und angewandte Mathematik* 148 (1918), pp. 122–145.

[SB10] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. 3rd ed. Springer New York, NY, 2010.

[Str15] S. H. Strogatz. *Nonlinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry, and Engineering*. 2nd ed. CRC Press, 2015.

[SN+10] B. Sz.-Nagy et al. *Harmonic Analysis of Operators on Hilbert Space*. 2nd ed. Springer New York, NY, 2010.

[Ugb80] O. O. Ugbebor. “Ergodic properties of Brownian Motion”. In: *Proceedings of the Edinburgh Mathematical Society* 23.3 (1980), 331–340.

[Via01] M. Viana. “Dynamical Systems: Moving into the Next Century”. In: *Mathematics Unlimited — 2001 and Beyond*. Ed. by B. Engquist and W. Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 1167–1178. ISBN: 978-3-642-56478-9.

[Vir+20] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272.

[WG02] T.-L. Wang and W. B. Gragg. “Convergence of the Shifted QR Algorithm for Unitary Hessenberg Matrices”. In: *Mathematics of Computation* 71.240 (2002), pp. 1473–1496. ISSN: 00255718, 10886842.

[Wat08] D. S. Watkins. “The QR Algorithm Revisited”. In: *SIAM Review* 50.1 (2008), pp. 133–145. ISSN: 00361445.

[Wat82] D. S. Watkins. “Understanding the QR Algorithm”. In: *SIAM Review* 24.4 (1982), pp. 427–440. ISSN: 00361445.

[WKR15] 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 (2015), pp. 1307–1346.