Optimal construction of Koopman eigenfunctions for prediction and control

Milan Korda, Igor Mezić

Optimal construction of Koopman eigenfunctions for prediction and control

Milan Korda[1,2], Igor Mezić[3]

September 4, 2019

Abstract

This work presents a novel data-driven framework for constructing eigenfunctions of the Koopman operator geared toward prediction and control. The method leverages the richness of the spectrum of the Koopman operator away from attractors to construct a rich set of eigenfunctions such that the state (or any other observable quantity of interest) is in the span of these eigenfunctions and hence predictable in a linear fashion. The eigenfunction construction is optimization-based with no dictionary selection required. Once a predictor for the uncontrolled part of the system is obtained in this way, the incorporation of control is done through a *multi-step* prediction error minimization, carried out by a simple linear least-squares regression. The predictor so obtained is in the form of a linear controlled dynamical system and can be readily applied within the Koopman model predictive control framework of [12] to control *nonlinear* dynamical systems using *linear* model predictive control tools. The method is entirely data-driven and based purely on convex optimization, with no reliance on neural networks or other non-convex machine learning tools. The novel eigenfunction construction method is also analyzed theoretically, proving rigorously that the family of eigenfunctions obtained is rich enough to span the space of all continuous functions. In addition, the method is extended to construct *generalized eigenfunctions* that also give rise Koopman invariant subspaces and hence can be used for linear prediction. Detailed numerical examples demonstrate the approach, both for prediction and feedback control.

Keywords: Koopman operator, eigenfunctions, model predictive control, data-driven methods

1 Introduction

The Koopman operator framework is becoming an increasingly popular tool for data-driven analysis of dynamical systems. In this framework, a nonlinear system is represented by an infinite dimensional *linear* operator, thereby allowing for spectral analysis of the nonlinear

[1]CNRS, Laboratory for Analysis and Architecture of Systems (LAAS), Toulouse, France. korda@laas.fr.

[2]Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Control Engineering, Prague, Czech Republic.

[3]University of California, Santa Barbara. mezic@ucsb.edu.

system akin to the classical spectral theory of linear systems or Fourier analysis. The theoretical foundations of this approach were laid out by Koopman in [11] but it was not until the early 2000's that the practical potential of these methods was realized in [20] and [18]. The framework became especially popular with the realization that the Dynamic Mode Decomposition (DMD) algorithm [26] developed in fluid mechanics constructs an approximation of the Koopman operator, thereby allowing for theoretical analysis and extensions of the algorithm (e.g., [33, 2, 13]). This has spurred an array of applications in fluid mechanics [25], power grids [24], neurodynamics [5], energy efficiency [8], or molecular physics [35], to name just a few.

Besides descriptive analysis of nonlinear systems, the Koopman operator approach was also utilized to develop systematic frameworks for control [12] (with earlier attempts in, e.g., [23, 32]), state estimation [29, 30] and system identification [17] of nonlinear systems. All these works rely crucially on the concept of *embedding* (or lifting) of the original state-space to a higher dimensional space where the dynamics can be accurately predicted by a linear system. In order for such prediction to be accurate over an extended time period, the embedding mapping must span an *invariant subspace* of the Koopman operator, i.e., the embedding mapping must consist of the (generalized) eigenfunctions of the Koopman operator (or linear combinations thereof).

It is therefore of paramout importance to construct accurate approximations of the Koopman eigenfunctions. The leading data-driven algorithms are either based on the Dynamic Mode Decomposition (e.g., [26, 33]) or the Generalized Laplace Averages (GLA) algorithm [21]. The DMD-type methods can be seen as finite section operator approximation methods, which do not exploit the particular Koopman operator structure and enjoy only weak spectral convergence guarantees [13]. On the other hand, the GLA method does exploit the Koopman operator structure and ergodic theory and comes with spectral convergence guarantees, but suffers from numerical instabilities for eigenvalues that do not lie on the unit circle (discrete time) or the imaginary axis (continuous time). Among the plethora of more recently introduced variations of the (extended) dynamic mode decomposition algorithm, let us mention the variational approach [34], the sparsity-based method [10] or the neural-networks-based method [31].

In this work, we propose a new algorithm for construction of the Koopman eigenfunctions from data. The method is geared toward transient, off-attractor, dynamics where the spectrum of the Koopman operator is *extremely rich*. In particular, provided that a *non-recurrent* surface exists in the state-space, *any* complex number is an eigenvalue of the Koopman operator with an associated continuous (or even smooth if so desired) eigenfunction, defined everywhere on the image of this non-recurrent surface through the flow of the dynamical system. What is more, the associated eigenspace is infinite-dimensional, parametrized by functions defined on the boundary of the non-recurrent surface. We leverage this richness to obtain a large number of eigenfunctions in order to ensure that the observable quantity of interest (e.g., the state itself) lies *within the span of the eigenfunctions* (and hence within an invariant subspace of the Koopman operator) and is therefore predictable in a *linear* fashion. The requirement that the embedding mapping spans an invariant subspace *and* the quantity of interest belongs to this subspace are crucial for practical applications: they imply both a *linear time evolution* in the embedding space as well the possibility *reconstruct* the quantity of interest in a *linear fashion*. On the other hand, having only a nonlinear reconstruction

mapping from the embedding space may lead to comparatively low-dimensional embeddings but may not buy us much practically since in that case we are replacing one nonlinear problem with another.

In addition to eigenfunctions, the proposed method can be extended to construct *generalized eigenfunctions* that also give rise to Koopman invariant subspaces and can hence be used for linear prediction; this further enriches the class of embedding mappings constructible using the proposed method.

On an algorithmic level, given a set of initial conditions lying on distinct trajectories, a set of complex numbers (the eigenvalues) and a set of continuous functions, the proposed method constructs eigenfunctions by simply "flowing" the values of the continuous functions forward in time according the eigenfunction equation, starting from the values of the continuous functions defined on the set of initial conditions. Provided the trajectories are non-periodic, this consistently and uniquely defines the eigenfunctions on the entire data set. These eigenfunctions are then extended to the entire state-space by interpolation or approximation. We prove that such extension is possible (i.e., there exist continuous eigenfunctions taking the computed values on the data set) provided that there is a non-recurrent surface passing through the initial conditions of the trajectories and we prove that such surface always exists provided the flow is rectifiable in the considered time interval. We also prove that the eigenfunctions constructed in this way span the space of all continuous functions in the limit as the number of boundary function-eigenvalue pairs tends to infinity. This implies that in the limit any continuous observable can be arbitrarily accurately approximated by *linear* combinations of the eigenfunctions, a crucial requirement for practical applications.

Importantly, both the values of the boundary functions and the eigenvalues can be selected using numerical optimization. The minimized objective is simply the projection error of the observables of interest onto the span of the eigenfunctions. For the problem of boundary function selection, we derive a convex reformulation, leading to a linear least-squares problem. The problem of eigenvalue selection appears to be intrinsically non-convex but is fortunately relatively low-dimensional, thereby amenable to a wide array of local and global non-convex optimization techniques. This is all that is required to construct the linear predictors in the uncontrolled setting.

In the controlled setting, we follow a two-step procedure. First, we construct a predictor for the uncontrolled part of the system (i.e., with the control being zero or any other fixed value). Next, using a second data set generated with control we minimize a *multi-step* prediction error in order to obtain the input matrix for the linear predictor. Crucially, the multi-step error minimization boils down to a simple *linear* least-squares problem; this is due to the fact that the dynamics and output matrices are already identified. This is a distinctive feature of the approach, compared to (E)DMD-based methods (e.g., [13, 23]) where only a one-step prediction error can be minimized in a convex fashion.

The predictors obtained in this way are then applied within the Koopman model predictive control (Koopman MPC) framework of [12], which we briefly review in this work. A core component of any MPC controller is the minimization of an objective functional over a multi-step prediction horizon; this is the primary reason for using a multi-step prediction error minimization during the predictor construction. However, the eigenfunction and linear predictor construction methods are completely general and immediately applicable, for

example, in the state estimation setting [29, 30].

The predictors obtained can then applied within the Koopman model predictive control (Koopman MPC) framework (see [12] for a general theory and [1, 15] for applications in fluid mechanics and power grid control). A core component of any MPC controller is the minimization of an objective functional over a multi-step prediction horizon; this is the primary reason for using a multi-step prediction error minimization during the predictor construction. However, the eigenfunction and linear predictor construction methods are completely general and immediately applicable, for example, in the state estimation setting [29, 28].

The fact that the spectrum of the Koopman operator is very rich in the space of continuous functions is a well known fact in the Koopman operator community; see, e.g., [16, Theorem 3.0.2]. In particular, the fact that, away from singularities, eigenfunctions corresponding to *arbitrary* eigenvalues can be constructed was noticed in [19] where these were termed *open eigenfunctions* and they were subsequently used in [4] to find conjugacies between dynamical systems. This work is, to the best of our knowledge, the first one to exploit the richness of the spectrum for prediction and control using *linear* predictors and to provide a theoretical analysis of the set of eigenfunctions obtained in this way. On the other hand, the spectrum of the Koopman operator "on attractor", in a post-transient regime, is much more structured and can be analyzed numerically in a great level of detail (see, e.g., [14, 9]).

Notation The set of real numbers is denoted by $\mathbb{R}$, the set of complex numbers by $\mathbb{C}$ and $\mathbb{N} = \{0, 1, \ldots\}$ denotes the set of natural numbers. The space of continuous complex-valued functions defined on a set $X \subset \mathbb{R}^n$ is denoted by $\mathcal{C}(X)$ or $\mathcal{C}(X; \mathbb{C})$, whenever we want to emphasize that the codomain is complex. The symbol $\circ$ denotes the pointwise composition of two functions, i.e., $(g \circ f)(x) = g(f(x))$. The Moore-Penrose pseudoinverse of a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is denoted by $\mathbf{A}^\dagger$, the transpose by $\mathbf{A}^\top$ and the conjugate (Hermitian) transpose by $\mathbf{A}^\mathsf{H}$. The identity matrix is denoted by $\mathbf{I}$. The symbol $\mathrm{diag}(\cdot, \ldots, \cdot)$ denotes a (block-) diagonal matrix composed of the arguments.

2 Koopman operator

We first develop our framework for uncontrolled dynamical systems and generalize it to controlled systems in Section 5. Consider therefore the nonlinear dynamical system

$$\dot{x} = f(x) \tag{1}$$

with the state $x \in X \subset \mathbb{R}^n$ and f Lipschitz continuous on X. The flow of this dynamical system is denoted by $S_t(x)$, i.e.,

$$\frac{d}{dt} S_t(x) = f(S_t(x)), \tag{2}$$

which we assume to be well defined for all $x \in X$ and all $t \geq 0$. The *Koopman operator semigroup* $(\mathcal{K}_t)_{t \geq 0}$ is defined by

$$\mathcal{K}_t g = g \circ S_t$$

for all $g \in \mathcal{C}(X)$. Since the flow of a dynamical system with Lipschitz vector field is also Lipschitz, it follows that

$$\mathcal{K}_t : \mathcal{C}(X) \to \mathcal{C}(X),$$

i.e., each element of the Koopman semigroup maps continuous functions to continuous functions. Crucially for us, each $\mathcal{K}_t$ is a *linear* operator[1].

With a slight abuse of language, from here on, we will refer to the Koopman operator semigroup simply as the *Koopman operator*.

Eigenfunctions An *eigenfunction* of the Koopman operator associated to an eigenvalue $\lambda \in \mathbb{C}$ is any function $\phi \in \mathcal{C}(X; \mathbb{C})$ satisfying

$$(\mathcal{K}_t \phi)(x) = e^{\lambda t} \phi(x), \tag{3}$$

which is equivalent to

$$\phi(S_t(x)) = e^{\lambda t} \phi(x). \tag{4}$$

Therefore, any such eigenfunction defines a coordinate evolving *linearly* along the flow of (1) and satisfying the *linear* ordinary differential equation (ODE)

$$\frac{d}{dt} \phi(S_t(x)) = \lambda \phi(S_t(x)). \tag{5}$$

2.1 Linear predictors from eigenfunctions

Since the eigenfunctions define linear coordinates, they can be readily used to construct *linear* predictors for the nonlinear dynamical system (1). The goal is to predict the evolution of a quantity of interest $\boldsymbol{h}(x)$ (often referred to as "observable" or an "output" of the system) along the trajectories of (1). The function

$$\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^{n_{\boldsymbol{h}}}$$

often represents the state itself, i.e., $\boldsymbol{h}(x) = x$ or an output of the system (e.g., the attitude of a vehicle or the kinetic energy of a fluid) or the cost function to be minimized within an optimal control problem or a nonlinear constraint on the state of the system (see Section 5.1 for concrete examples). The distinctive feature of this work is the requirement that the predictor constructed be a *linear* dynamical system. This facilitates the use of *linear* tools for state estimation and control, thereby greatly simplifying the design procedure as well as drastically reducing computational and deployment costs (see [29] for applications of this idea to state estimation and [12] for model predictive control).

Let $\phi_1, \ldots, \phi_N$ be eigenfunctions of the Koopman operator with the associated (not necessarily distinct) eigenvalues $\lambda_1, \ldots, \lambda_N$. Then we can construct a linear predictor of the form

$$\dot{z} = Az \tag{6a}$$

$$z_0 = \boldsymbol{\phi}(x_0), \tag{6b}$$

$$\hat{y} = Cz, \tag{6c}$$

[1]To see the linearity of $\mathcal{K}_t$ consider $g_1 \in \mathcal{C}(X)$, $g_2 \in \mathcal{C}(X)$ and $\alpha \in \mathbb{C}$. Then we have $\mathcal{K}_t(\alpha g_1 + g_2) = (\alpha g_1 + g_2) \circ S_t = \alpha g_1 \circ S_t + g_2 \circ S_t = \alpha \mathcal{K}_t g_1 + \mathcal{K}_t g_2$.

where

$$A = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}, \qquad \boldsymbol{\phi} = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}, \tag{7}$$

and where $\hat{y}$ is the prediction of $\boldsymbol{h}(x)$. To be more precise, the prediction of $\boldsymbol{h}(x(t)) = \boldsymbol{h}(S_t(x_0))$ is given by

$$\boldsymbol{h}(x(t)) \approx \hat{y}(t) = Ce^{At}z_0 = C \begin{bmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_N t} \end{bmatrix} z_0.$$

The matrix C is chosen such that the projection of $\boldsymbol{h}$ onto $\operatorname{span}\{\phi_1, \ldots, \phi_N\}$ is minimized, i.e., C solves the optimization problem

$$\min_{C \in \mathbb{C}^{n_{\boldsymbol{h}} \times N}} \|\boldsymbol{h} - C\boldsymbol{\phi}\|, \tag{8}$$

where $\|\cdot\|$ is a norm on the space of continuous functions (e.g., the sup-norm or the L_2 norm).

Prediction error Since $\phi_1, \ldots, \phi_N$ are the eigenfunctions of the Koopman operator, the prediction of the evolution of the eigenfunctions along the trajectory of (1) is error-free, i.e.,

$$z(t) = \boldsymbol{\phi}(x(t)).$$

Therefore, the *sole source* of the prediction error

$$\|\boldsymbol{h}(x(t)) - \hat{y}(t)\| \tag{9}$$

is the error in the projection of $\boldsymbol{h}$ onto $\operatorname{span}\{\phi_1, \ldots, \phi_N\}$, quantified by Eq. (8). In particular, if $\boldsymbol{h} \in \operatorname{span}\{\phi_1, \ldots, \phi_N\}$ (with the inclusion understood componentwise), we have

$$\|\boldsymbol{h}(x(t)) - \hat{y}(t)\| = 0 \quad \forall\, t \geq 0.$$

This observation[2] will be crucial for defining a meaningful objective function when learning the eigenfunctions.

Goals The primary goal of this paper is a data-driven construction of a set of eigenfunctions $\{\phi_1, \ldots, \phi_N\}$ such that the error (8) is minimized. In doing so, we introduce and theoretically analyze a novel eigenfunction construction procedure (Section 3) from which a data-driven optimization-based algorithm is derived in Section 4. A secondary goal of this paper is to use the constructed eigenfunctions for prediction in a controlled setting (Section 5) and subsequently for model predictive control design (Section 5.1).

[2]To see this, notice that $\boldsymbol{h}$ belonging to the span of $\boldsymbol{\phi}$ is equivalent to the existence of a matrix C such that $\boldsymbol{h} = C\boldsymbol{\phi}$. Therefore, $\boldsymbol{h}(x(t)) = \boldsymbol{h} \circ S_t = C\boldsymbol{\phi} \circ S_t = Ce^{At}\boldsymbol{\phi} = \hat{y}(t)$.

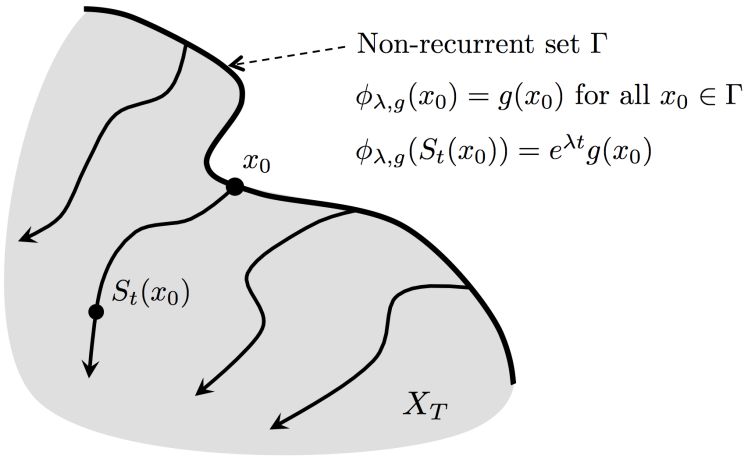

Figure 1: Construction of eigenfunctions using a non-recurrent set Γ and a continuous function g defined on Γ.

3 Non-recurrent sets and eigenfunctions

In this section we show how non-recurrent sets naturally give rise to eigenfunctions. Let time $T \in (0, \infty)$ be given. A set $\Gamma \subset X$ is called non-recurrent if

$$x \in \Gamma \Longrightarrow S_t(x) \notin \Gamma \quad \forall t \in (0, T].$$

Given *any* function $g \in \mathcal{C}(\Gamma)$ and *any* $\lambda \in \mathbb{C}$, we can *construct* an eigenfunction of the Koopman operator by simply solving the defining ODE (5) "initial condition by initial condition" for all initial conditions $x_0 \in \Gamma$. Mathematically, we define for all $x_0 \in \Gamma$

$$\phi_{\lambda,g}(S_t(x_0)) = e^{\lambda t} g(x_0), \tag{10}$$

which defines the eigenfunction on the entire image X_T of Γ by the flow $S_t(\cdot)$ for $t \in [0, T]$. Written explicitly, this image is

$$X_T = \bigcup_{t \in [0,T]} S_t(\Gamma) = \bigcup_{t \in [0,T]} \{S_t(x_0) \mid x_0 \in \Gamma\}.$$

See Figure 1 for an illustration. To get an explicit expression for $\phi_{\lambda,g}(x)$ we flow backward in time until we hit the non-recurrent set Γ, obtaining

$$\phi_{\lambda,g}(x) = e^{-\lambda \tau(x)} g(S_{\tau(x)}(x)) \tag{11}$$

for all $x \in X_T$, where

$$\tau(x) = \inf_{t \in \mathbb{R}} \{t \mid S_t(x) \in \Gamma\}$$

is the first time that the trajectory of (1) hits Γ starting from x. By construction, for $x \in X_T$ we have

$$\tau(x) \in [-T, 0].$$

The results of this discussion are summarized in the following theorem:

Theorem 1 *Let Γ be a non-recurrent set, $g \in \mathcal{C}(\Gamma)$ and $\lambda \in \mathbb{C}$. Then $\phi_{\lambda,g}$ defined by (11) is an eigenfunction of the Koopman operator on X_T. In particular, $\phi_{\lambda,g}$ satisfies (4) and (5) for all $x \in X_T$ and all t such that $S_t(x) \in X_T$. In addition, if g is Lipschitz continuous, then also*

$$\nabla \phi_{\lambda,g} \cdot f = \lambda \phi_{\lambda,g} \tag{12}$$

almost everywhere in X_T (and everywhere in X_T if g is differentiable).

Proof: The result follows by construction. Since Γ is non-recurrent, the definition (10) is consistent for all $t \in [0, T]$ and equivalent to (11). Since $S_0(x_0) = x_0$, we have $\phi_{\lambda,g}(x_0) = g(x_0)$ for all $x_0 \in \Gamma$ and hence Eq. (10) is equivalent to the defining Eq. (4) defining the Koopman eigenfunctions. To prove (12) observe that g Lipschitz implies that $\phi_{\lambda,g}$ is Lipschitz and the result follows from (5) by the chain rule and the Rademacher theorem, which asserts almost-everywhere differentiability of Lipschitz functions. □

Several remarks are in order.

Richness We emphasize that this construction works for an *arbitrary* $\lambda \in \mathbb{C}$ and an *arbitrary* function g continuous[3] on Γ. Therefore, there are uncountably many eigenfunctions that can be generated in this way and in this work we exploit this to construct a sufficiently rich collection of eigenfunctions such that the projection error (8) is minimized. The richness of the class of eigenfunctions is analyzed theoretically in Section 3.2 and used practically in Section 4 for data-driven learning of eigenfunctions.

Time direction The same construction can be carried out backwards in time or forward and backward in time, as long as Γ is non-recurrent for the time interval considered. In this work we focus on forward-in-time construction which naturally lends itself to data-driven applications where typically only forward-in-time data is available.

History This construction is very closely related to the concept of *open eigenfunctions* introduced in [19], which were subsequently used in [4] to find conjugacies between dynamical systems. This work is, to the best of our knowledge, the first one to use such construction for prediction and control using linear predictors.

3.1 Non-recurrent set vs Non-recurrent surface

It is useful to think of the non-recurrent set Γ as an $n-1$ dimensional surface so that X_T is full dimensional. Such surface can be for example any level set of a Koopman eigenfunction with non-zero real part (e.g., isostable) or a level set of a Lyapunov function. However, these level sets can be hard to obtain in practice; fortunately, their knowledge is *not required*. The reason for this is that the set Γ can be a finite *discrete* set in which case X_T is simply the collection of all trajectories with initial conditions in Γ; since trajectories are one-dimensional,

[3] In this work we restrict our attention to functions g continuous on Γ but in principle discontinuous functions of a suitable regularity class could be used as well.

any randomly generated finite (or countable) discrete set will be non-recurrent on $[0, T]$ with probability one. This is a key feature of our construction that will be utilized in Section 4 for a data-driven learning of the eigenfunctions. A natural question arises: can one find a non-recurrent surface passing through a given finite discrete non-recurrent set Γ? The answer is positive, provided that the points in Γ do not lie on the same trajectory and the flow can be rectified:

Lemma 1 *Let $\Gamma = \{x^1, \dots, x^M\}$ be a finite set of points in X and let X' be a full dimensional compact set containing Γ on which the flow of (1) can be rectified, i.e., there exists a diffeomorphism $\zeta : Y' \to X'$ through which (1) is conjugate[4] to*

$$\dot{y} = (0, \dots, 0, 1) \tag{13}$$

with $Y' \subset \mathbb{R}^n$ convex. Assume that no two points in the set Γ lie on the same trajectory of (1). Then there exists an $n-1$ dimensional surface $\hat{\Gamma} \supset \Gamma$, closed in the standard topology of $\mathbb{R}^n$, such that $x \in \hat{\Gamma}$ implies $S_t(x) \notin \hat{\Gamma}$ for any $t > 0$ satisfying $S_{t'}(x) \in X'$ for all $t' \in [0, t]$.

Proof: Let $y^j = \zeta^{-1}(x^j)$, $j = 1, \dots, M$ and let $\Gamma_Y = \{y^1, \dots, y^M\} = \zeta^{-1}(\Gamma)$. The goal is to construct an $n-1$ dimensional surface $\hat{\Gamma}_Y$, closed in $\mathbb{R}^n$, passing through the points $\{y^1, \dots, y^M\}$, i.e., $\hat{\Gamma}_Y \supset \Gamma_Y$ and satisfying

$$y \in \hat{\Gamma}_Y \Longrightarrow \hat{S}_t(y) \notin \hat{\Gamma}_Y \tag{14}$$

for any $t > 0$ satisfying $\hat{S}_{t'}(y) \in Y'$ for all $t' \in [0, t]$, where $\hat{S}_{t'}(y)$ denotes the flow of (13). Once $\hat{\Gamma}_Y$ is constructed, the required surface $\hat{\Gamma}$ is obtained as $\hat{\Gamma} = \zeta(\hat{\Gamma}_Y)$.

Given the nature of the rectified dynamics (13), the condition (14) will be satisfied if $\hat{\Gamma}_Y$ is a graph of a Lipschitz continuous function $\gamma : \mathbb{R}^{n-1} \to \mathbb{R}$ such that

$$\hat{\Gamma}_Y = \{(y_1, \dots, y_n) \in Y' \mid y_n = \gamma(y_1, \dots, y_{n-1})\}.$$

We shall construct such function γ. Denote $\bar{y}^j = (y_1^j, \dots, y_{n-1}^j)$ the first $n-1$ components of each point $y^j \in \mathbb{R}^n$. The nature of the rectified dynamics (13), convexity of Y' and the fact that x^j's (and hence y^j's) do not lie on the same trajectory implies that $\bar{y}^j$'s are distinct. Therefore, the pairs $(\bar{y}^j, y_n^j) \in \mathbb{R}^{n-1} \times \mathbb{R}$, $j = 1, \dots, M$, can be interpolated with a Lipschitz continuous function $\gamma : \mathbb{R}^{n-1} \to \mathbb{R}$. One such example of γ is

$$\gamma(y_1, \dots, y_{n-1}) = \max_{j \in \{1, \dots, M\}} \Big\{ \big[1 - \|(y_1, \dots, y_{n-1}) - \bar{y}^j\|\big] y_n^j \Big\},$$

where we assume that $y_n^j \geq 0$ (which can be achieved without loss of generality by translating the y_n-th coordinate since Y' is compact) and $\|\cdot\|$ is any norm on $\mathbb{R}^{n-1}$. Another example is a multivariate polynomial interpolant of degree d which always exists for any d satisfying $\binom{n-1+d}{d} \geq M$. Since both ζ and γ are Lipschitz continuous, the surface $\hat{\Gamma}$ is $n-1$ dimensional and is closed in the standard topology of $\mathbb{R}^n$. □

Remark 1 (Rectifyability) *It is a well-known fact that for the dynamics (1) to be rectifyable on a domain X', the vector field f should be non-singular on this domain (i.e., $f(x) \neq 0$ for all $x \in X'$); see, e.g., [3, Chapter 2, Corollary 12].*

[4]Two dynamical systems $\dot{x} = f_1(x)$ and $\dot{y} = f_2(y)$ are conjugate through a diffeomorphism ζ if the associated flows $S_{1,t}$ and $S_{2,t}$ satisfy $S_{1,t}(x) = \zeta(S_{2,t}(\zeta^{-1}(x)))$. For the vector fields, this means that $f_2(y) = (\frac{\partial \zeta}{\partial y})^{-1} f_1(\zeta(y))$.

3.2 Span of the eigenfunctions

A crucial question arises: can one approximate an arbitrary continuous function by a linear combination of the eigenfunctions constructed using the approach described in Section 3 by selecting more and more boundary functions g and eigenvalues λ? Crucially for our application, if this is the case, we can make the projection error (8) and thereby also the prediction error (9) arbitrarily small by enlarging the set of eigenfunctions $\boldsymbol{\phi}$. If this is the case, does one have to enlarge the set of eigenvalues or does it suffice to only increase the number of boundary functions g? In this section we give a precise answer to these questions.

Before we do so, we set up some notation. Given any set $\Lambda \subset \mathbb{C}$, we define

$$\text{lattice}(\Lambda) = \Big\{ \sum_{k=1}^{p} \alpha_k \lambda_k \mid \lambda_k \in \Lambda,\ \alpha_k \in \mathbb{N}_0,\ p \in \mathbb{N} \Big\}. \tag{15}$$

A basic result in the Koopman operator theory asserts that if Λ is a set of eigenvalues of the Koopman operator, then so is lattice(Λ). Now, given $\Lambda \subset \mathbb{C}$ and $G \subset \mathcal{C}(\Gamma)$, we define

$$\Phi_{\Lambda,G} = \{\phi_{\lambda,g} \mid \lambda \in \Lambda,\ g \in G\}, \tag{16}$$

where $\Phi_{\lambda,g}$ is given by (11). In words, $\phi_{\Lambda,G}$ is the set of all eigenfunctions arising from all combinations of boundary functions in G and eigenvalues in Λ using the procedure described in Section 3.

Now we are ready to state the main result of this section:

Theorem 2 *Let Γ be a non-recurrent set, closed in the standard topology of $\mathbb{R}^n$ and let $\Lambda_0 \subset \mathbb{C}$ be an arbitrary[5] set of complex numbers such that at least one has a non-zero real part and $\Lambda_0 = \bar{\Lambda}_0$. Set $\Lambda = \text{lattice}(\Lambda_0)$ and let $G = \{g_i\}_{i=1}^{\infty}$ denote an arbitrary set of functions whose span is dense in $\mathcal{C}(\Gamma)$ in the supremum norm. Then the span of $\Phi_{\Lambda,G}$ is dense in $\mathcal{C}(X_T)$, i.e., for every $h \in \mathcal{C}(X_T)$ and any $\epsilon > 0$ there exist eigenfunctions $\phi_1, \ldots, \phi_N \in \Phi_{\Lambda,G}$ and complex numbers $c_1, \ldots, c_N$ such that*

$$\sup_{x \in X_T} \left| h(x) - \sum_{i=1}^{N} c_i \phi_i(x) \right| < \epsilon.$$

Proof: Step 1. First, we observe that it is sufficient to prove the density of

$$\Phi_\Lambda = \text{span}\{\phi_{\lambda,g} \mid \lambda \in \Lambda,\ g \in \mathcal{C}(\Gamma)\}$$

in $\mathcal{C}(X_T)$. To see this, assume Φ_Λ is dense in $\mathcal{C}(X_T)$ and consider any function $h \in \mathcal{C}(X_T)$ and $\epsilon > 0$. Then there exist eigenfunctions ϕ_{λ_i,g_i}, $i = 1, \ldots, k$, defined by (11) with $g_i \in \mathcal{C}(\Gamma)$ and $\lambda_i \in \Lambda$ such that

$$\sup_{x \in X_T} \Big| \sum_{i=1}^{k} \phi_{\lambda_i,g_i}(x) - h(x) \Big| < \epsilon.$$

[5]In the extreme case, the set Λ_0 can consists of a single non-zero real number or a single conjugate pair with non-zero real part.

Here we used the linearity of (11) with respect to g to subsume the coefficients of the linear combination to the functions g_i.

Since $\mathrm{span}\{G\}$ is dense in $\mathcal{C}(\Gamma)$, there exist functions $\tilde{g}_i \in \mathrm{span}\{G\}$ such that

$$\sup_{x\in\Gamma} |g_i - \tilde{g}_i| < \frac{\epsilon}{k} \min\{1, |e^{\lambda_i T}|\}.$$

In addition, because Eq. (11) defining $\phi_{\lambda,g}$ is linear in g for any fixed λ, it follows that $\phi_{\lambda,\tilde{g}_i} \in \mathrm{span}\{\Phi_{\Lambda,G}\}$ and hence also $\sum_i \phi_{\lambda,\tilde{g}_i} \in \mathrm{span}\{\Phi_{\Lambda,G}\}$. Therefore it suffices to bound the error between h and $\phi_{\lambda,\tilde{g}}$. We have

$$\begin{aligned}
\sup_{x\in X_T} |\sum_{i=1}^{k} \phi_{\lambda_i,\tilde{g}_i}(x) - h(x)| &\le \sup_{x\in X_T} \Big|\sum_i \phi_{\lambda_i,g_i}(x) - h(x)\Big| + \sup_{x\in X_T} \left|\sum_{i=1}^{k} (\phi_{\lambda_i,g_i}(x) - \phi_{\lambda_i,\tilde{g}_i}(x))\right| \\
&\le \epsilon + \sup_{x\in X_T} \left|\sum_{i=1}^{k} e^{-\lambda_i \tau(x)} [g(S_{\tau(x)}(x)) - \tilde{g}_i(S_{\tau(x)}(x))]\right| \\
&\le \epsilon + \sum_{i=1}^{k} \left[\sup_{x\in X_T} |e^{-\lambda_i\tau(x)}| \cdot \sup_{x\in\Gamma} |g_i(x) - \tilde{g}_i(x)|\right] \\
&\le \epsilon + \frac{\epsilon}{k} \sum_{i=1}^{k} \max\{1, |e^{-\lambda_i T}|\} \min\{1, |e^{\lambda_i T}|\} \le 2\epsilon,
\end{aligned}$$

where we used the facts that $\tau(x) \in [0, T]$ and $S_{\tau(x)}(x) \in \Gamma$.

Step 2. We will show that Φ_Λ is a subalgebra of $\mathcal{C}(X_T)$ closed under complex conjugation that separates points and contains a non-zero constant function; then the Stone–Weierstrass theorem will imply the desired results. By construction, Φ_Λ is a linear subspace of $\mathcal{C}(X_T)$ and hence it suffices to show that Φ_Λ is closed under multiplication and complex conjugation, separates points and contains a non-zero constant function.

To see that that Φ_Λ is closed under multiplication, consider $\phi_1 \in \Phi_\Lambda$ and $\phi_2 \in \Phi_\Lambda$ of the form $\phi_1 = \sum_i \phi_{\lambda_i,g_i}$ and $\phi_1 = \sum_i \phi_{\lambda_i',g_i'}$ with $\lambda_i \in \Lambda$ and $\lambda_i' \in \Lambda$, $g_i \in \mathcal{C}(\Gamma)$, $g_i' \in \mathcal{C}(\Gamma)$. Then we have

$$\phi_1\phi_2 = \sum_{i,j} \phi_{\lambda_i,g_i}\phi_{\lambda_j',g_j'} = \sum_{i,j} e^{-\lambda_i\tau} e^{-\lambda_j'\tau} (g_i \circ S_\tau)(g_j' \circ S_\tau) = \sum_{i,j} \phi_{\lambda_i+\lambda_j', g_i g_j'} \in \Phi_\Lambda$$

since $\lambda_i + \lambda_j' \in \Lambda$ because of (15) and $g_i' g_j' \in \mathcal{C}(\Gamma)$.

To see that Φ_Λ separated points of X_T, i.e., for each $x_1 \in X_T$ and $x_2 \in X_T$, $x_1 \neq x_2$, there exists $\phi \in \Phi_\Lambda$ such that $\phi(x_1) \neq \phi(x_2)$. We consider two cases. First, suppose that x_1 and x_2 lie on the same trajectory of (1). Then these two points are separated by $\phi_{\lambda,1}$ for any λ with nonzero real part; by the assumptions of the theorem there exists $\lambda \in \Lambda$ with a non-zero real part and hence the associated $\phi_{\lambda,1}$ belongs to Φ_Λ. Second, suppose x_1 and x_2 do not lie on the same trajectory. Then these two points are separated by $\phi_{0,g}$ with $g(S_{\tau(x_1)}(x_1)) \neq g(S_{\tau(x_2)}(x_2))$; such g always exists in $\mathcal{C}(\Gamma)$ because $\mathcal{C}(\Gamma)$ separates points of Γ.

To see that Φ_Γ contains a constant non-zero function, consider $\phi_{\lambda,g}$ with $\lambda = 0$ and $g = 1$, which is equal to 1 on X_T and belongs to Φ_Λ.

Finally, to see that Φ_Λ is closed under complex conjugation, consider $\phi = \sum_i \phi_{\lambda_i, g_i}$, $\lambda_i \in \Lambda$ and $g_i \in \mathcal{C}(\Gamma)$. Then

$$\bar{\phi} = \sum_i \overline{\phi_{\lambda_i, g_i}} = \sum_i e^{-\bar{\lambda}_i \tau} \overline{g_i} \circ S_\tau = \sum_i \phi_{\bar{\lambda}_i, \overline{g_i}} \in \Phi_\Lambda$$

since $\bar{\Lambda} = \Lambda$ by assumption and $\mathcal{C}(\Gamma) = \overline{\mathcal{C}(\Gamma)}$. □

Selection of λ's and g's An interesting question arises regarding an optimal selection of the eigenvalues $\lambda \in \Lambda$ and boundary functions $g \in G$ assuring that the projection error (8) converges to zero as fast as possible. As it turns, the boundary functions g can be chosen optimally using convex optimization, for each component of $\boldsymbol{h}$ separately. The optimal choice of the eigenvalues λ appears to be more difficult and seems to be inherently non-conconvex. Choices of both g's and λ's using optimization are discussed in detail in Sections 4.1 and 4.2. Importantly, the algorithms for selection of g's and λ's do not rely on any problem insight and do not require the choice of basis functions.

Example (role of λ's) To get some intuition to the interplay between $\boldsymbol{h}$ and λ, consider the linear system $\dot{x} = ax$, $x \in [0, 1]$ and the non-recurrent set $\Gamma = \{1\}$. In this case, any function on Γ is constant, so the set of boundary functions G can be chosen to consist of the constant function equal to one. Then it follows from (11) that given any complex number $\lambda \in \mathbb{C}$, the associated eigenfunction is $\phi_\lambda(x) := \phi_{\lambda,1}(x) = x^{\frac{\lambda}{a}}$. Given an observable $\boldsymbol{h}$, the optimal choice of the set of eigenvalues $\Lambda \subset \mathbb{C}$ is such that the projection error (8) is minimized, which in this case translates to making

$$\min_{(c_\lambda \in \mathbb{C})_{\lambda \in \Lambda}} \left\| \boldsymbol{h} - \sum_{\lambda \in \Lambda} c_\lambda x^{\frac{\lambda}{a}} \right\| \tag{17}$$

as small as possible with the choice of Λ. Clearly, the optimal choice (in terms of the number of eigenfunctions required to achieve a given projection error) of Λ depends on $\boldsymbol{h}$. For example, for $\boldsymbol{h} = x$, the optimal choice is $\Lambda = \{a\}$, leading to a zero projection error with only one eigenfunction. For other observables, however, the choice of $\Lambda = \{a\}$ need not be optimal. For example, for $\boldsymbol{h} = x^b$, $b \in \mathbb{R}$, the optimal choice leading to zero projection error with only one eigenfunction is $\Lambda = \{a \cdot b\}$. The statement of Theorem 2 then translates to the statement that the projection error (17) is zero for any $\Lambda = \{k \cdot \lambda_0 \mid k \in \mathbb{N}_0\}$ with $\lambda_0 < 0$ and any continuous observable $\boldsymbol{h}$. The price to pay for this level of generality is the asymptotic nature of the result, requiring the cardinality of Λ (and hence the number of eigenfunctions) going to infinity.

3.3 Generalized eigenfunctions

This section describes how *generalized eigenfunctions* can be constructed with a simple modification of the proposed method. Importantly for this work, generalized eigenfunctions also give rise to Koopman invariant subspaces and therefore can be readily used for linear

prediction. Given a complex number λ and $g_1, \ldots, g_{n_\lambda}$, consider the Jordan block

$$J_\lambda = \begin{bmatrix} \lambda & 1 & \\ & \ddots & 1 \\ & & \lambda \end{bmatrix}$$

and define

$$\begin{bmatrix} \psi_{\lambda,g_1}(S_t(x_0)) \\ \vdots \\ \psi_{\lambda,g_{n_\lambda}}(S_t(x_0)) \end{bmatrix} = e^{J_\lambda t} \begin{bmatrix} g_1(x_0) \\ \vdots \\ g_{n_\lambda}(x_0) \end{bmatrix} \tag{18}$$

for all $x_0 \in \Gamma$ or equivalently

$$\begin{bmatrix} \psi_{\lambda,g_1}(x) \\ \vdots \\ \psi_{\lambda,g_{n_\lambda}}(x) \end{bmatrix} = e^{-J_\lambda \tau(x)} \begin{bmatrix} g_1(S_{\tau(x)}(x)) \\ \vdots \\ g_{n_\lambda}(S_{\tau(x)}(x)) \end{bmatrix} \tag{19}$$

for all $x \in X_T$. Define also

$$\boldsymbol{\psi} = \left[\psi_{\lambda,g_1}, \ldots, \psi_{\lambda,g_{n_\lambda}}\right]^\top.$$

With this notation, we have the following theorem:

Theorem 3 *Let Γ be a non-recurrent set, $g_i \in \mathcal{C}(\Gamma)$, $i = 1, \ldots, n_\lambda$, and $\lambda \in \mathbb{C}$. Then the subspace*

$$\operatorname{span}\{\psi_{\lambda,g_1}, \ldots, \psi_{\lambda,g_{n_\lambda}}\} \subset \mathcal{C}(X_T)$$

is invariant under the action of the Koopman semigroup $\mathcal{K}_t$. Moreover

$$\boldsymbol{\psi}(S_t(x)) = e^{J_\lambda t}\boldsymbol{\psi}(x) \tag{20}$$

and

$$\frac{d}{dt}\boldsymbol{\psi}(S_t(x)) = J_\lambda \boldsymbol{\psi}(S_t(x)) \tag{21}$$

for any $x \in X_T$ and any $t \in [0, T]$ such that $S_{t'}(x) \in X_T$ for all $t' \in [0, t]$.

Proof: Let $h \in \operatorname{span}\{\psi_{\lambda,g_1}, \ldots, \psi_{\lambda,g_{n_\lambda}}\}$. Then $h = c^\top \boldsymbol{\psi}$ for some $c \in \mathbb{C}^{n_\lambda}$. Given $x \in X_T$, we have

$$\boldsymbol{\psi}(S_t(x)) = e^{-J_\lambda(\tau(x)-t)} \begin{bmatrix} g_1(S_{\tau(x)-t}(S_t(x))) \\ \vdots \\ g_{n_\lambda}(S_{\tau(x)-t}(S_t(x))) \end{bmatrix} = e^{J_\lambda t} e^{-J_\lambda(\tau(x))} \begin{bmatrix} g_1(S_{\tau(x)}(x)) \\ \vdots \\ g_{n_\lambda}(S_{\tau(x)}(x)) \end{bmatrix} = e^{J_\lambda t}\boldsymbol{\psi}(x),$$

which is (20). Therefore $\mathcal{K}_t h = c^\top \boldsymbol{\psi} \circ S_t = c^\top e^{tJ_\lambda}\boldsymbol{\psi} \in \operatorname{span}\{\psi_{\lambda,g_1}, \ldots, \psi_{\lambda,g_{n_\lambda}}\}$ as desired. Eq. (21) follows immediately from (20). □

Beyond Jordan blocks The proof of Theorem 3 reveals that there was nothing special of using a Jordan block in (18). Indeed, the entire construction works with an *arbitrary* matrix A in place of J_λ. However, nothing is gained by using an arbitrary matrix A since the span of the generalized eigenfunctions constructed using A is identical to that of the corresponding Jordan normal form of A, which is just the direct sum of the spans associated to the individual Jordan blocks.

4 Learning eigenfunctions from data

Now we use the construction of Section 3 to learn eigenfunction from data. In particular, we leverage the freedom in *choosing* (optimally, if possible) the eigenvalues λ as well as the boundary functions g to learn a *rich* set of eigenfunctions such that the projection error (8) (and thereby also the prediction error (9)) is minimized. The choice of g's and λ's is carried out using numerical optimization (convex in the case of g's), without relying on problem insight and without requiring a choice of basis. We first describe how the eigenfunctions can be constructed from data, assuming the eigenvalues and boundary functions have been chosen and after that we describe how this choice can be carried out using numerical optimization.

Throughout this section we assume that we have available data in the form of M_{t} distinct equidistantly sampled trajectories with $M_{\mathrm{s}}+1$ samples each, where $M_{\mathrm{s}} = T/T_{\mathrm{s}}$ with T_{s} being the sampling interval (what follows straightforwardly generalizes to non-equidistantly sampled trajectories of unequal length). That is, the data is of the form

$$\mathcal{D} = \left((x_k^j)_{k=0}^{M_{\mathrm{s}}} \right)_{j=1}^{M_{\mathrm{t}}}, \tag{22}$$

where the superscript indexes the trajectories and the subscript the discrete time within the trajectory, i.e., $x_k^j = S_{kT_{\mathrm{s}}}(x_0^j)$, where x_0^j is the initial condition of the j^{th} trajectory. The non-recurrent set Γ is simply defined as

$$\Gamma = \{x_0^1, \ldots, x_0^{M_{\mathrm{t}}}\}.$$

We also assume that we have chosen a vector of complex numbers (i.e., the eigenvalues)

$$\Lambda = (\lambda_1, \ldots, \lambda_N)$$

as well as a vector of continuous functions

$$G = (g_1, \ldots, g_N)$$

defining the values of the eigenfunctions on the non-recurrent set Γ. The functions in G will be referred to as *boundary functions*.

Now we can construct N eigenfunctions using the developments of Section 3. To do so, define the matrix

$$\mathbf{G}(i,j) = g_i(x_0^j) \tag{23}$$

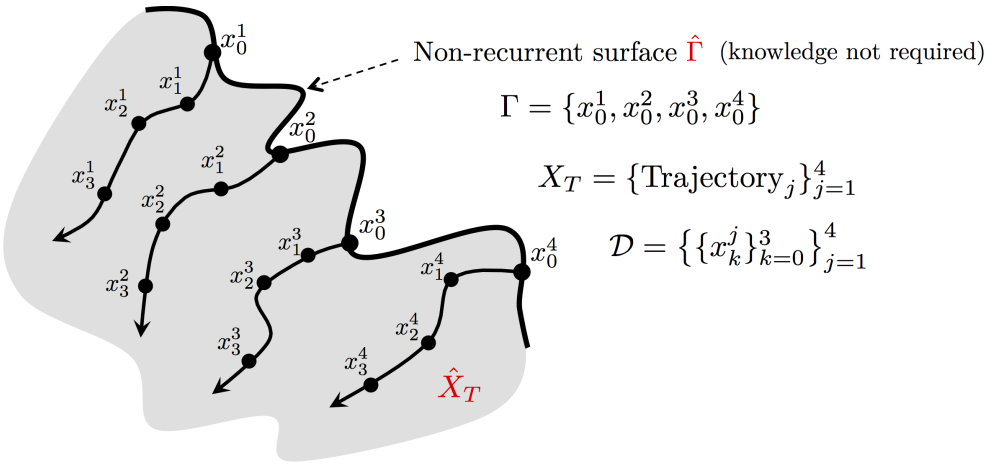

Figure 2: Illustration of the non-recurrent set Γ, the non-recurrent surface $\hat{\Gamma}$. Note that the non-recurrent surface $\hat{\Gamma}$ does not need to be known explicitly for learning the eigenfunctions. Only sampled trajectories $\mathcal{D}$ with initial conditions belonging to distinct trajectories are required (see Lemma 1). Note also that even though the existence of the non-recurrent surface is assured by Lemma 1, this surface can be highly irregular (e.g., oscillatory), depending on the interplay between the dynamics and the locations of the initial conditions.

collecting the values of the boundary functions on the initial points of the trajectories in the data set. Define also

$$\phi_{\lambda_i,g_i}(x_0^j) := g(x_0^j), \quad j = 1, \ldots, M_{\mathrm{t}}.$$

Then Eq. (10) uniquely defines the values of ϕ_{λ_i,g_i} on the entire data set. Specifically, we have

$$\phi_{\lambda_i,g_i}(x_k^j) = e^{\lambda_i k T_{\mathrm{s}}} \mathbf{G}(i,j) \tag{24}$$

for all $k \in \{0, \ldots M_{\mathrm{s}}\}$ and all $j \in \{1, \ldots M_{\mathrm{t}}\}$. According to Lemma 1 (provided its assumptions hold), there exists an entire non-recurrent surface $\hat{\Gamma}$ passing through the initial conditions of the trajectories in the data set $\mathcal{D}$. Even though this surface is unknown to us, its existence implies that the eigenfunctions computed through (24) on $\mathcal{D}$ are in fact samples of continuous eigenfunctions defined on

$$\hat{X}_T = \bigcup_{t \in [0,T]} S_t(\hat{\Gamma})\,; \tag{25}$$

see Figure 2 for an illustration. As a result, the eigenfunctions ϕ_{λ_i,g_i} can be learned on the entire set $\hat{X}_T$ (or possibly even larger region) via interpolation or approximation. Specifically, given a set of basis functions

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{N_\beta} \end{bmatrix}$$

with $\beta_i \in \mathcal{C}(X)$, we can solve the interpolation problems

$$\begin{aligned} \underset{c \in \mathbb{C}^{N_\beta}}{\text{minimize}} \quad & \delta_1 \|c\|_1 + \|c\|_2^2 \\ \text{subject to} \quad & c^\top \boldsymbol{\beta}(x_k^j) = \phi_{\lambda_i,g_i}(x_k^j), \\ & k \in \{0, \ldots, M_{\mathrm{s}}\}, \;\; j \in \{1, \ldots, M_{\mathrm{t}}\} \end{aligned} \tag{26}$$

for each $i \in \{1, \dots, N\}$. Alternatively, we can solve the approximation problems

$$\underset{c \in \mathbb{C}^{N_\beta}}{\text{minimize}} \quad \sum_{k=0}^{\mathrm{M_s}} \sum_{j=1}^{M_\mathrm{t}} \left| c^\top \boldsymbol{\beta}(x_k^j) - \phi_{\lambda_i, g_i}(x_k^j) \right|^2 + \delta_1 \|c\|_1 + \delta_2 \|c\|_2^2 \tag{27}$$

for each $i \in \{1, \dots, N\}$. In both problems the classical ℓ_1 and ℓ_2 regularizations are optional, for promoting sparsity of the resulting approximation and preventing overfitting; the numbers $\delta_1 \geq 0$, $\delta_2 \geq 0$ are the corresponding regularization parameters. The resulting approximation to the eigenfunction ϕ_{λ_i, g_i}, denoted by $\hat{\phi}_{\lambda_i, g_i}$, is given by

$$\hat{\phi}_{\lambda_i, g_i}(x) = c_i^\top \boldsymbol{\beta}(x), \tag{28}$$

where $c_i^\top$ is the solution to (26) or (27) for a given $i \in \{1, \dots, N\}$. Note that the approximation $\hat{\phi}_{\lambda_i, g_i}(x)$ is defined on the entire state space X; if the interpolation method (26) is used then the approximation is exact on the data set $\mathcal{D}$ and one expects it to be accurate on X_T and possibly also on $\hat{X}_T$, provided that the non-recurrent surface $\hat{\Gamma}$ (if it exists) and the functions in G give rise to eigenfunctions well approximable (or learnable) by functions from the set of basis functions $\boldsymbol{\beta}$. The eigenfunction learning procedure is summarized in Algorithm 1.

Algorithm 1 Eigenfunction learning

Require: Data $\mathcal{D} = \left((x_k^j)_{k=0}^{M_\mathrm{s}} \right)_{j=1}^{M_\mathrm{t}}$, matrix $\mathbf{G} \in \mathbb{C}^{N \times M_\mathrm{t}}$, complex numbers $\Lambda = (\lambda_1, \dots, \lambda_{N_\Lambda})$, basis functions $\boldsymbol{\beta} = [\beta_1, \dots, \beta_{N_\beta}]^\top$, sampling time T_s.

1: **for** $i \in \{1, \dots, N\}$ **do**
2: **for** $j \in \{1, \dots, M_\mathrm{t}\}$, $k \in \{0, \dots, M_\mathrm{s}\}$ **do**
3: $\phi_{\lambda, g}(x_k^j) := e^{\lambda k T_\mathrm{s}} \mathbf{G}(i, j)$
4: Solve (26) or (27) to get c_i
5: Set $\hat{\phi}_i := c_i^\top \boldsymbol{\beta}$

Output: $\{\hat{\phi}_i\}_{i \in \{1, \dots, N\}}$

Interpolation methods We note that (26) and (27) are only two possibilities for extending the values of the eigenfunctions from the data points to all of X; more sophisticated interpolation/approximation methods may yield superior results.

Choosing initial conditions As long as the initial conditions in Γ lie on distinct trajectories that are non-periodic over the simulated time interval, the set Γ is non-recurrent as required by our approach. This is achieved with probability one if, for example, the initial conditions are sampled uniformly at random over X (assuming the cardinality of X is infinite) and the dynamics is non-periodic or the simulation time is chosen such that the trajectories are non-periodic over the simulated time interval. In practice, one will typically choose the initial conditions such that the trajectories sufficiently cover a subset of the state space of interest (e.g., the safe region of operation of a vehicle). In addition, it is advantageous (but not necessary) to sample the initial conditions from a sufficiently regular surface (e.g., a ball or ellipsoid) approximating a non-recurrent surface in order to ensure that the resulting eigenfunctions are well behaved (e.g., in terms of the Lipschitz constant) and hence easily interpolable / approximable.

4.1 Optimal selection of boundary functions g

First we describe the general idea behind the optimal selection of g's and then derive from it a convex-optimization-based algorithm. Given $\lambda \in \mathbb{C}$, let $\mathcal{L}_\lambda : \mathcal{C}(\Gamma) \to \mathcal{C}(X_T)$ denote the operator that maps a boundary function $g \in \mathcal{C}(\Gamma)$ to the associated eigenfunction $\phi_{\lambda,g}$, i.e.,

$$\mathcal{L}_\lambda g = e^{-\lambda\tau}(g \circ S_\tau).$$

Notice, crucially, that this operator is linear in g. Given a vector of continuous boundary functions

$$G = (g_1, \dots, g_N),$$

$g_i \in \mathcal{C}(\Gamma)$, and a vector of eigenvalues

$$\Lambda = (\lambda_1, \dots, \lambda_N),$$

$\lambda_i \in \mathbb{C}$, the projection error (9) boils down to

$$\|\boldsymbol{h} - \mathrm{proj}_{\mathcal{V}_{G,\Lambda}} \boldsymbol{h}\|, \tag{29}$$

where

$$\mathcal{V}_{G,\Lambda} = \mathrm{span}\{\mathcal{L}_{\lambda_1} g_1, \dots, \mathcal{L}_{\lambda_N} g_N\}.$$

Here $\mathrm{proj}_{\mathcal{V}_{G,\Lambda}}$ denotes the projection operator[6] onto the finite-dimensional subspace $\mathcal{V}_{G,\Lambda}$.

The goal is then to minimize (29) with respect to G. In general, this is a non-convex problem. However, we will show that if each component of $\boldsymbol{h}$ is considered separately, this problem admits a convex reformulation. Assume therefore that the total budget of N boundary functions is partitioned as

$$G = (G_1, \dots, G_{n_{\boldsymbol{h}}})$$

with $\sum_{i=1}^{n_{\boldsymbol{h}}} N_i = N$, where $N_i = \#G_i$. The eigenvalues are partitioned correspondingly, i.e.,

$$\Lambda = (\Lambda_1, \dots, \Lambda_{n_{\boldsymbol{h}}}), \quad \#\Lambda_i = N_i.$$

Then, given $i \in \{1, \dots, n_{\boldsymbol{h}}\}$ and $\Lambda_i \in \mathbb{C}^{N_i}$, we want to solve

$$\underset{G_i \in \mathcal{C}(\Gamma)^{N_i}}{\text{minimize}} \|\boldsymbol{h}_i - \mathrm{proj}_{\mathcal{V}_{G_i,\Lambda_i}} \boldsymbol{h}_i\|.$$

This problem is equivalent to

$$\underset{g_{i,j} \in \mathcal{C}(\Gamma),\, c_{i,j} \in \mathbb{C}}{\text{minimize}} \ \Big\|\boldsymbol{h}_i - \sum_{j=1}^{N_i} c_{i,j} \mathcal{L}_{\lambda_{i,j}} g_{i,j}\Big\|.$$

Since $\mathcal{L}_\lambda$ is linear in g, we have $c_{i,j}\mathcal{L}_{\lambda_{i,j}} g_{i,j} = \mathcal{L}_{\lambda_{i,j}}(c_{i,j} g_{i,j})$ with $c_{i,j} g_{i,j} \in \mathcal{C}(\Gamma)$ and hence, using the substitution $c_{i,j} g_{i,j} \leftarrow g_{i,j}$, this problem is equivalent to

$$\underset{g_{i,j} \in \mathcal{C}(\Gamma)}{\text{minimize}} \ \Big\|\boldsymbol{h}_i - \sum_{j=1}^{N_i} \mathcal{L}_{\lambda_{i,j}} g_{i,j}\Big\|, \tag{30}$$

which is a *convex* function of $g_{i,j}$ as desired.

[6]Depending on the norm used in (29), the projection on $\mathcal{V}_{G,\Lambda}$ may not be unique, in which case we assume that a tiebreaker function has been applied, making the projection operator well defined.

Regularization Beyond minimization of the projection error (29), one may also wish to optimize the regularity of the resulting eigenfunctions $\phi_{\lambda,g} = \mathcal{L}_\lambda g$ as these functions will have to be represented in a computer when working with data. Such regularity can be optimized by including an additional term penalizing, for instance a norm of the gradient of $\mathcal{L}_\lambda g$, resulting in a Sobolev-type regularization. Adding this term to (30) results in

$$\underset{g_{i,j}\in\mathcal{C}(\Gamma)}{\text{minimize}} \ \Big\|\boldsymbol{h}_i - \sum_{j=1}^{N_i} \mathcal{L}_{\lambda_{i,j}} g_{i,j}\Big\| + \alpha \sum_{j=1}^{N_i} \|\mathcal{D}\mathcal{L}_{\lambda_{i,j}} g_{i,j}\|', \tag{31}$$

where $\mathcal{D}$ is a differential operator (e.g., the gradient), $\alpha \geq 0$ is a regularization parameter and the norms $\|\cdot\|$ and $\|\cdot\|'$ do not need to be the same.

4.1.1 Data-driven algorithm for optimal selection of g's

Now we derive a data-driven algorithm from the abstract developments above. The idea is to optimize directly over the values of the boundary functions g, which, when restricted to the data set (47), are just finitely many complex numbers collected in the matrix $\mathbf{G}$ (23). Formally, assume that the norm in (29) is given by the L_2 norm with respect to the empirical measure on the data set. In that case, problem (30) becomes

$$\underset{\mathbf{g}_{i,j}\in\mathbb{C}^{M_\mathrm{t}}}{\text{minimize}} \ \Big\|\mathbf{h}_i - \sum_{j=1}^{N_i} \mathbf{L}_{\lambda_{i,j}} \mathbf{g}_{i,j}\Big\|_2, \tag{32}$$

where $\mathbf{h}_i$ is the vector collecting the values of $\boldsymbol{h}_i$ stacked on top each other trajectory-by-trajectory, the optimization variables $\mathbf{g}_{i,j}$ are vectors containing the values of the boundary functions $g_{i,j}$ on the starting points of the trajectories in the data set and the matrices $\mathbf{L}_{\lambda_{i,j}}$ are given by

$$\mathbf{L}_{\lambda_{i,j}} = \mathrm{bdiag}(\underbrace{\Lambda_{i,j}, \ldots, \Lambda_{i,j}}_{\mathrm{M_t\ times}}),$$

where

$$\Lambda_{i,j} = [1, e^{\lambda_{i,j}T_\mathrm{s}}, e^{2\lambda_{i,j}T_\mathrm{s}}, \ldots, e^{M_\mathrm{s}\lambda_{i,j}T_\mathrm{s}}]^\top.$$

This problem is equivalent to

$$\underset{\mathbf{g}_i\in\mathbb{C}^{N_i M_\mathrm{t}}}{\text{minimize}} \ \Big\|\mathbf{h}_i - \mathbf{L}_{\Lambda_i}\mathbf{g}_i\Big\|_2^2, \tag{33}$$

where

$$\mathbf{L}_{\Lambda_i} = [\mathbf{L}_{\lambda_{i,1}}, \mathbf{L}_{\lambda_{i,2}}, \ldots, \mathbf{L}_{\lambda_{i,N_i}}], \quad \mathbf{g}_i = [\mathbf{g}_{i,1}^\top, \mathbf{g}_{i,2}^\top, \ldots, \mathbf{g}_{i,N_i}^\top]^\top. \tag{34}$$

Problem (33) is a least-squares problem with optimal solution

$$\mathbf{g}_i^\star = \mathbf{L}_{\Lambda_i}^\dagger \mathbf{h}_i. \tag{35}$$

Define the N-by-M_t matrix $\mathbf{G}$ by

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_{1,1}^\star & \cdots & \mathbf{g}_{1,N_1}^\star & \cdots & \mathbf{g}_{n_{\boldsymbol{h}},1}^\star & \cdots & \mathbf{g}_{n_{\boldsymbol{h}},N_{\boldsymbol{h}}}^\star \end{bmatrix}^\top \tag{36}$$

collecting the values of all boundary functions on the initial points of the trajectories, i.e., $\mathbf{G}(i,j)$ is the value of boundary function i on the data point x_0^j (here the vector $\mathbf{g}_i^\star$ is assumed to be partitioned in the same way as $\mathbf{g}_i$ in (34)). The matrix $\mathbf{G}$ is then used in Algorithm 1.

Regularization With regularization, assuming the $\|\cdot\|'$ norm in (31) is the L_2 norm with respect to the empirical measure on the data set, we get

$$\underset{\mathbf{g}_i \in \mathbb{C}^{N_i M_\mathrm{t}}}{\text{minimize}} \ \left\|\mathbf{h}_i - \mathbf{L}_{\Lambda_i}\mathbf{g}_i\right\|_2^2 + \|\mathbf{D}\mathbf{L}_{\Lambda_i}\mathbf{g}_i\|_2^2, \tag{37}$$

where $\mathbf{D}$ is a discrete representation of the differential operator $\mathcal{D}$ used in (31). The optimal solution to (37) is given by

$$\mathbf{g}_i^\star = \begin{bmatrix} \mathbf{L}_{\Lambda_i} \\ \mathbf{D}\mathbf{L}_{\Lambda_i} \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{h}_i \\ 0 \end{bmatrix}.$$

The matrix $\mathbf{G}$ is then defined by (36) and used in Algorithm 1.

4.2 Selection of eigenvalues λ

Now we describe how to select the eigenvalues using numerical optimization. For simplicity, we will work in the setting without regularization, the generalization being straightforward. Plugging in (35) into (33) and using the fact that $\mathbf{L}_{\Lambda_i}\mathbf{L}_{\Lambda_i}^\dagger$ is the orthogonal projection operator onto the column space of $\mathbf{L}_{\Lambda_i}$ (and hence is self-adjoint and idempotent), the minimum in (33) is equal to

$$\|\mathbf{h}_i\|_2^2 - \|\mathbf{L}_{\Lambda_i}\mathbf{L}_{\Lambda_i}^\dagger \mathbf{h}_i\|_2^2. \tag{38}$$

The optimal choice of $\Lambda_i = (\lambda_{i,1}, \ldots, \lambda_{i,N_i}) \in \mathbb{C}^{N_i}$ minimizes (38). Unfortunately, (38) is a non-convex function of Λ_i with no obvious convexification available. Fortunately, the value of N_i is typically modest and therefore this optimization problem can be (at least approximately) solved by using local optimization initialized from a collection of randomly chosen initial conditions. Crucial to this is the availability of an analytic expression for the gradient of (38) with respect to Λ_i; for simplicity of analysis we assume that the matrix $\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{L}_{\Lambda_i}$ is invertible, which is the case generically provided that $M_\mathrm{t}M_\mathrm{s} > M_\mathrm{t}N_i$ (in which case the matrix $\mathbf{L}_{\Lambda_i}$ is tall). Deriving the gradient in the fully general setting is straightforward but lengthy, using the derivative of matrix pseudoinverse [27]. Assuming $\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{L}_{\Lambda_i}$ is invertible, (38) becomes

$$p_i(\Lambda_i) := \|\mathbf{h}_i\|_2^2 - \mathbf{h}_i^\mathsf{H}\mathbf{L}_{\Lambda_i}(\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{L}_{\Lambda_i})^{-1}\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{h}_i. \tag{39}$$

Using the fact that for a matrix $A(\theta)$, depending on $\theta \in \mathbb{R}$, we have

$$\frac{d}{d\theta}[A(\theta)^{-1}] = -A(\theta)^{-1}\frac{dA(\theta)}{d\theta}A(\theta)^{-1},$$

we obtain

$$\frac{\partial p_i(\Lambda_{i,j})}{\partial \theta_{i,j}} = -2\mathcal{R}\left\{\mathbf{h}_i^\mathsf{H}\frac{\partial \mathbf{L}_{\Lambda_i}}{\partial \theta_{i,j}}\mathbf{q}_i\right\} + \mathbf{q}_i^\mathsf{H}\left[\mathbf{L}_{\Lambda_i}^\mathsf{H}\frac{\partial \mathbf{L}_{\Lambda_i}}{\partial \theta_{i,j}} + \left(\frac{\partial \mathbf{L}_{\Lambda_i}}{\partial \theta_{i,j}}\right)^\mathsf{H}\mathbf{L}_{\Lambda_i}\right]\mathbf{q}_i, \tag{40}$$

where $\theta_{i,j} \in \mathbb{R}$ stands either for the real or imaginary part of $\lambda_{i,j}$ (the expressions are the same for both) and

$$\mathbf{q}_i = (\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{L}_{\Lambda_i})^{-1}\mathbf{L}_{\Lambda_i}^\mathsf{H}\mathbf{h}_i.$$

Continuing, we have

$$\frac{\partial \mathbf{L}_{\Lambda_i}}{\partial \theta_{i,j}} = \left[0, \ldots, 0, \frac{\mathbf{L}_{i,j}}{\partial \theta_{i,j}}, 0, \ldots, 0\right]$$

with the same block structure as (34) and the non-zero element on the j^{th} block position and with

$$\frac{\mathbf{L}_{i,j}}{\partial\theta_{i,j}} = \text{bdiag}\left(\frac{\partial\Lambda_{i,j}}{\partial\theta_{i,j}}, \ldots, \frac{\partial\Lambda_{i,j}}{\partial\theta_{i,j}}\right),$$

where

$$\frac{\partial\Lambda_{i,j}}{\partial\lambda^{\mathbb{R}}_{i,j}} = T_{\text{s}} \begin{bmatrix} 0 \\ 1 \\ 2 \\ \vdots \\ M_{\text{s}} \end{bmatrix} \circ \Lambda_{i,j}, \quad \frac{\partial\Lambda_{i,j}}{\partial\lambda^{\mathbb{I}}_{i,j}} = \sqrt{-1}\frac{\partial\Lambda_{i,j}}{\partial\lambda^{\mathbb{R}}_{i,j}},$$

with $\circ$ denoting the componentwise (Hadamard) product and $\sqrt{-1}$ the imaginary unit and with $\lambda^{\mathbb{R}}_{i,j}$ and $\lambda^{\mathbb{I}}_{i,j}$ denoting the real respectively imaginary parts of $\lambda_{i,j}$

This allows us to evaluate $\frac{\partial p(\Lambda_i)}{\partial\lambda^{\mathbb{R}}_{i,j}}$ and $\frac{\partial p(\Lambda_i)}{\partial\lambda^{\mathbb{I}}_{i,j}}$ for all j and hence allows us to evaluate the gradient of $p(\Lambda_i)$ with respect to the real and imaginary parts of the eigenvalues in Λ_i.

Special structure We note that without regularization, the least-squares problem (33) can be decomposed "trajectory-by-trajectory" to M_{t} independent least-squares problems. Moreover, the matrix in each of these least-squares problems is a Vandermonde matrix for which specialized least-squares solution methods exist (e.g., [6]). This special special structure also implies that the gradient of $p(\Lambda_i)$ is a sum of M_{t} independently computable terms and hence amenable to parallel computation.

4.3 Generalized eigenfunctions from data

Algorithm 1 can be readily extended to the case of generalized eigenfunctions as described in Section 3.3. Step 3 of this algorithm is replaced by

$$\begin{bmatrix} \psi_{\lambda,g_1}(x^j_k) \\ \vdots \\ \psi_{\lambda,g_{n_\lambda}}(x^j_k) \end{bmatrix} = e^{J_\lambda k T_{\text{s}}} \begin{bmatrix} g_1(x^j_0) \\ \vdots \\ g_{n_\lambda}(x^j_0) \end{bmatrix},$$

where, as in Section 3.3, J_λ is a Jordan block of size n_λ associated to an eigenvalue λ and $g_1, \ldots, g_{n_\lambda}$ are continuous boundary functions. Step 4 of Algorithm 1 (interpolation / approximation) is then performed on each ψ_{λ,g_i} separately. Note that with Jordan block of size one, the entire procedure reduces to the case of eigenfunctions.

We note that there are no restrictions on the parings of Jordan blocks (of arbitrary size) and continuous boundary functions, thereby providing additional freedom for constructing very rich invariant subspaces of the Koopman operator.

4.4 Obtaining matrices A and C

Here we describe how to obtain the matrices A and C in (6). Let

$$\hat{\boldsymbol{\phi}} = \begin{bmatrix} \hat{\phi}_1 \\ \vdots \\ \hat{\phi}_N \end{bmatrix}$$

denote the vector of N eigenfunction approximations obtained from Algorithm 1. The matrix A is then given simply by

$$A = \operatorname{diag}(\lambda_1, \ldots, \lambda_N),$$

where λ_i are eigenvalues associated to $\hat{\phi}_i$. Provided that the boundary functions were chosen optimally as described in Section 4.1, the matrix C is obtained as

$$C = \operatorname{bdiag}(\mathbf{1}_{N_1}, \ldots, \mathbf{1}_{N_{n_{\boldsymbol{h}}}}), \tag{41}$$

where $\mathbf{1}_{N_i}$ is a row vector of ones of length N_i and N_i constitute a partition of N as described in Section 4.1.

Generally, irrespective of how the boundary functions were chosen, the matrix C can be obtained by (approximately) solving (8) with $\boldsymbol{\phi}$ replaced by $\hat{\boldsymbol{\phi}}$. This problem is typically not solvable analytically in high dimensions since it requires a multivariate integration or uniform bounding of a continuous function (depending on the norm used in (8)). Therefore, we use a sample-based approximation. If the L_2 norm is used in (8), we solve the optimization problem

$$\underset{C \in \mathbb{C}^{n_{\boldsymbol{h}} \times N}}{\text{minimize}} \quad \sum_{i=1}^{\bar{M}} \left\| \boldsymbol{h}(\bar{x}_i) - C\hat{\boldsymbol{\phi}}(\bar{x}_i) \right\|_2^2. \tag{42}$$

For the sup-norm, we solve

$$\underset{C \in \mathbb{C}^{n_{\boldsymbol{h}} \times N}}{\text{minimize}} \quad \max_{i \in \{1,\ldots,\bar{M}\}} \left\| \boldsymbol{h}(\bar{x}_i) - C\hat{\boldsymbol{\phi}}(\bar{x}_i) \right\|_\infty. \tag{43}$$

The samples $\{\bar{x}_i\}_{i=1}^{\bar{M}}$ can either coincide with the samples $\{x_k^j\}_{j,k}$ used for learning of the eigenfunctions or they can be generated anew (e.g., to emphasize certain regions of state-space where accurate projection (and hence prediction) is required). See Section 4.6 for a discussion of computational aspects of solving these two problems.

4.5 Exploiting algebraic structure

It follows immediately from the definition of the Koopman operator eigenfunction (4) that products and powers of eigenfunctions are also eigenfunctions. In particular, given $\phi_1, \ldots, \phi_{N_0}$ eigenfunctions with the associated eigenvalues $\lambda_1, \ldots, \lambda_{N_0}$, the function

$$\phi = \phi_1^{p_1} \cdot \ldots \cdot \phi_N^{p_{N_0}} \tag{44}$$

is a Koopman eigenfunction with the associated eigenvalue

$$\lambda = p_1\lambda_1 + \ldots + p_{N_0}\lambda_{N_0}.$$

This holds for any nonnegative *real or integer* powers $p_1, \ldots, p_{N_0}$.

This algebraic structure can be exploited to generate additional eigenfunction approximations starting from those obtained using Algorithm 1, at a very little additional computational cost. In particular, one can construct only a handful of eigenfunction approximations using Algorithm 1, e.g., with Λ being a single real eigenvalue or a single complex conjugate pair and the set G consisting of linear coordinate functions x_i, $i = 1, \ldots, n$. This initial set of eigenfunctions can then be used to generate a very large number of additional eigenfunction approximations using (44) in order to ensure that the projection error (8) is small. When queried at a previously unseen state (e.g., during feedback control), only the eigenfunction approximations $\hat{\phi}_1, \ldots, \hat{\phi}_{N_0}$ have to be computed using interpolation or approximation (which can be costly if the number of basis functions $\boldsymbol{\beta}$ is large in step 5 of Algorithm 1) whereas the remaining eigenfunction approximations are obtained by simply taking powers and products according to (44).

4.6 Computational aspects

The main computational burden of the proposed method is the solution to the interpolation or approximation problems (26) and (27). Both these problems are *convex* optimziation problems that can be reliably solved using generic packages such as MOSEK or Gurobi. For very large problem instances, specialized packages for ℓ_1 / ℓ_2 regularized least-squares problems may need to be deployed (see, e.g. [36, 22]). We note that for each pair (λ, g) the coefficients $\boldsymbol{\beta}(x_k^j)$ remain the same, which can be exploited to drastically speed up the solution.

For problems without ℓ_1 regularization, we have an explicit solution

$$c_i = \mathbf{W}^\dagger \mathbf{w}$$

for (26) and

$$c_i = (\mathbf{W} + \delta_2 \mathbf{I})^\dagger \mathbf{w},$$

for (27) where

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{\beta}(x_0^1) & \ldots & \boldsymbol{\beta}(x_{M_\mathrm{s}}^1) & \boldsymbol{\beta}(x_0^2) & \ldots & \boldsymbol{\beta}(x_{M_\mathrm{s}}^2) & \ldots & \boldsymbol{\beta}(x_0^{M_\mathrm{t}}) & \ldots & \boldsymbol{\beta}(x_{M_\mathrm{s}}^{M_\mathrm{t}}) \end{bmatrix}^\top$$

and

$$\mathbf{w} = \begin{bmatrix} \phi_i(x_0^1) & \ldots & \phi_i(x_{M_\mathrm{s}}^1) & \phi_i(x_0^2) & \ldots & \phi_i(x_{M_\mathrm{s}}^2) & \ldots & \phi_i(x_0^{M_\mathrm{t}}) & \ldots & \phi_i(x_{M_\mathrm{s}}^{M_\mathrm{t}}) \end{bmatrix}^\top,$$

where $\phi_i = \phi_{\lambda_i, g_i}$ as defined in (24).

The projection problems (42) and (43) are both convex optimization problems that can be easily solved using generic convex optimization packages (e.g., MOSEK or Gurobi). The use of such tools is necessary for the sup-norm projection problem (43). However, for the least-squares projection problem (42), linear algebra is enough with the analytical solution being

$$C = [\boldsymbol{h}(\bar{x}_1), \ldots, \boldsymbol{h}(\bar{x}_{\bar{M}})][\hat{\boldsymbol{\phi}}(\bar{x}_1), \ldots, \hat{\boldsymbol{\phi}}(\bar{x}_{\bar{M}})]^\dagger.$$

5 Linear predictors for controlled systems

In this section we describe how to build linear predictors for controlled systems. Assume a nonlinear controlled system of the form

$$\dot{x} = f(x) + Hu \tag{45}$$

with the state $x \in X \subset \mathbb{R}^n$ and control input $u \in U \subset \mathbb{R}^m$ and $H \in \mathbb{R}^{n\times m}$. This form is general since any control system of the form $\dot{\tilde{x}} = \tilde{f}(\tilde{x}, v)$ can be transformed to (45) using the state inflation[7] $x = [\tilde{x}^\top, v^\top]^\top$, $\dot{v} = u$, which leads to

$$f := \begin{bmatrix} \tilde{f}(\tilde{x}, v) \\ 0 \end{bmatrix}, \quad H := \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix}.$$

As in [12], the goal is to construct a predictor in the form of a controlled *linear* dynamical system

$$\dot{z} = Az + Bu \tag{46a}$$
$$z_0 = \hat{\boldsymbol{\phi}}(x_0), \tag{46b}$$
$$\hat{y} = Cz. \tag{46c}$$

Whereas [12] uses a one-step procedure (essentially a generalization of the extended dynamic mode decomposition (EDMD) to controlled systems), here we follow a two-step procedure, where we first construct eigenfunctions for the uncontrolled system

$$\dot{x} = f(x).$$

We assume that we have two data sets available. The first one is an uncontrolled dataset $\mathcal{D}$ with the same structure as in (47) in Section 4. The second data set, $\mathcal{D}_\mathrm{c}$, is with control in the form of $M_\mathrm{t,c}$ equidistantly sampled trajectories with $M_\mathrm{s,c} + 1$ samples each, i.e.,

$$\mathcal{D}_\mathrm{c} = \left((x_k^j)_{k=0}^{M_\mathrm{s,c}}, (u_k^j)_{k=0}^{M_\mathrm{s,c}-1} \right)_{j=1}^{M_\mathrm{t,c}}, \tag{47}$$

where $x_{k+1}^j = S_{T_\mathrm{s}}(x_k^j, u_k^j)$, where $S_t(x, u)$ denotes the solution to (45) at time t starting from x and with the control input held constant and equal to u in $[0, t]$. We note that both the number of trajectories and the trajectory length may differ for the controlled and uncontrolled data sets.

Step 1 – $\hat{\boldsymbol{\phi}}$, A, C In the first step of the procedure we construct approximate eigenfunctions $\hat{\boldsymbol{\phi}}$ of (1) (with $f(x) = f_\mathrm{c}(x, 0)$) using the procedure described in Section 4 , obtaining also the matrices A and C.

[7] Imposing constraints on the control input of the state-inflated system corresponds to imposing constraints on the derivative of the original control input, which is important in practical applications.

Step 2 – matrix B In order to obtain the matrix B we perform a regression on the controlled data set (47). The quantity to be minimized is a *multi-step* prediction error. Crucially, this multi-step error can be minimized in a *convex* fashion; this is due to the fact that the matrices A and C are already known and fixed at this step and the predicted output $\hat{y}$ of (46) depends affinely on B. This is in stark contrast to EDMD-type methods, where only one-step ahead prediction error can be minimized in a convex fashion. In order to keep expressions simple we assume that the time interval over which we want to minimize the prediction error coincides with the length of the trajectories in our data set (everything generalizes straightforwardly to shorter prediction times). The problem to be solved therefore is

$$\underset{B_{\mathrm{d}}\in\mathbb{R}^{N\times m}}{\text{minimize}} \quad \sum_{j=1}^{M_{\mathrm{t,c}}}\sum_{k=1}^{M_{\mathrm{s,c}}} \|\boldsymbol{h}(x_k^j) - \hat{y}_k(x_0^j)\|_2^2, \tag{48}$$

where

$$\hat{y}_k(x_0^j) = CA_{\mathrm{d}}^k z_0^j + \sum_{i=0}^{k-1} CA_{\mathrm{d}}^{k-i-1} B_{\mathrm{d}} u_i^j$$

is the output $\hat{y}$ of (45) at time kT_{s} starting from the (known) initial condition

$$z_0^j = \hat{\boldsymbol{\phi}}(x_0^j).$$

The discretized matrices A_{d} (known) and B_{d} (to be determined) are related to A and B by

$$A_{\mathrm{d}} = e^{AT_{\mathrm{s}}}, \quad B_{\mathrm{d}} = \left(\int_0^{T_{\mathrm{s}}} e^{-As}\, ds\right) B. \tag{49}$$

We note that in the above expression the matrix multiplying B is invertible for any $T_{\mathrm{s}} > 0$ and therefore B can be uniquely recovered from the knowledge of B_{d}. Using vectorization, the output $\hat{y}_k(x_0^j)$ can be re-written as

$$\hat{y}_k(x_0^j) = CA_{\mathrm{d}}^k z_0^j + \sum_{i=0}^{k-1} \left[(u_i^j)^\top \otimes (CA_{\mathrm{d}}^{k-i-1})\right] \mathrm{vec}(B_{\mathrm{d}}), \tag{50}$$

where $\mathrm{vec}(\cdot)$ denotes the (column-major) vectorization of a matrix and $\otimes$ the Kronecker product. Since A_{d}, C, z_0^j and $\boldsymbol{h}(x_k^j)$ are all known, plugging in (50) to the least-squares problem (48) leads to the minimization problem

$$\underset{b\,\in\,\mathbb{R}^{mN}}{\text{minimize}} \quad \|\Theta b - \theta\|_2^2, \tag{51}$$

where

$$\Theta = \begin{bmatrix}\Theta_1^\top & \Theta_2^\top & \dots & \Theta_{M_{\mathrm{t}}}^\top\end{bmatrix}^\top, \qquad \theta = \begin{bmatrix}\theta_1^\top & \theta_2^\top & \dots & \theta_{M_{\mathrm{t}}}^\top\end{bmatrix}^\top$$

with

$$\Theta_j = \begin{bmatrix} (u_0^j)^\top \otimes C \\ (u_1^j)^\top \otimes C + u_0^j \otimes (CA_{\mathrm{d}}) \\ \vdots \\ \sum_{i=0}^{M_{\mathrm{s}}-1} \left[(u_i^j)^\top \otimes (CA_{\mathrm{d}}^{M_{\mathrm{s}}-i-1})\right] \end{bmatrix}, \qquad \theta_j = \begin{bmatrix} \boldsymbol{h}(x_1^j) - CA_{\mathrm{d}} z_0^j \\ \boldsymbol{h}(x_2^j) - CA_{\mathrm{d}}^2 z_0^j \\ \vdots \\ \boldsymbol{h}(x_{M_{\mathrm{s}}}^j) - CA_{\mathrm{d}}^{M_{\mathrm{s}}} z_0^j \end{bmatrix}.$$

The matrix B_{d} is then given by

$$B_{\mathrm{d}} = \mathrm{vec}^{-1}(\Theta^{\dagger}\theta), \tag{52}$$

where $\Theta^{\dagger}\theta$ is an optimal solution to (51). Since $A = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$, the matrix B is obtained as

$$B = \left(\int_0^{T_{\mathrm{s}}} e^{-As}\, ds\right)^{-1} B_{\mathrm{d}} = \mathrm{diag}\Big(\frac{\lambda_1}{1 - e^{-\lambda_1 T_{\mathrm{s}}}}, \ldots, \frac{\lambda_N}{1 - e^{-\lambda_N T_{\mathrm{s}}}}\Big) B_{\mathrm{d}}. \tag{53}$$

5.1 Koopman model predictive control

In this section we briefly describe how the linear predictor (46) can be used within a *linear* model predictive control (MPC) scheme to control *nonlinear* dynamical systems. This method was originally developed in [12] and this section closely follows this work; the reader is referred therein for additional details as well as to [15, 1] for applications in power grid and fluid flow control.

An MPC controller solves at each step of a closed-loop operation an optimization problem where a given cost function is minimized over a finite prediction horizon with respect to the predicted control inputs and predicted outputs of the dynamical system. For nonlinear systems, this is almost always a nonconvex optimization problem due to the equality constraint in the form of the nonlinear dynamics. In the Koopman MPC framework, on the other hand, we solve the *convex quadratic* optimization problem (QP)

$$\begin{array}{lll}
\underset{u_i, z_i, \hat{y}_i}{\text{minimize}} & z_{N_{\mathrm{p}}}^{\top} Q_{N_{\mathrm{p}}} z_{N_{\mathrm{p}}} + q_{N_{\mathrm{p}}}^{\top} z_{N_{\mathrm{p}}} + \sum_{i=0}^{N_{\mathrm{p}}-1} z_i^{\top} Q_i z_i + u_i^{\top} R_i u_i + q_i^{\top} z_i + r_i^{\top} u_i & \\
\text{subject to} & z_{i+1} = A_{\mathrm{d}} z_i + B_{\mathrm{d}} u_i, & i = 0, \ldots, N_{\mathrm{p}} - 1 \\
& E_i \hat{y}_i + F_i u_i \leq b_i, & i = 0, \ldots, N_{\mathrm{p}} - 1 \\
& E_{N_{\mathrm{p}}} \hat{y}_{N_{\mathrm{p}}} \leq b_{N_{\mathrm{p}}} & \\
& \hat{y}_i = C z_i & \\
\text{parameter} & z_0 = \hat{\boldsymbol{\phi}}(x_{\mathrm{current}}), &
\end{array} \tag{54}$$

where the cost matrices $Q_i \in \mathbb{R}^{n_{\boldsymbol{h}} \times n_{\boldsymbol{h}}}$ and $R_i \in \mathbb{R}^{m \times m}$ are positive semidefinite and N_{p} is the prediction horizon. The optimization problem is parametrized by $x_{\mathrm{current}} \in \mathbb{R}^n$ which is the current state measured during the closed-loop operation. The control input applied to the system is the first element of the control sequence optimal in (54). Notice that in (54) we use directly the discretized predictor matrices A_{d} and B_{d}, where $A_{\mathrm{d}} = \mathrm{diag}(e^{\lambda_1 T_{\mathrm{s}}}, \ldots, e^{\lambda_N T_{\mathrm{s}}})$ and B_{d} is given by (52) with T_{s} being the sampling interval. See Algorithm 2 for a summary of the Koopman MPC in this sampled data setting.

Handling nonlinearities Crucially, all nonlinearities in x are subsumed in the output mapping $\boldsymbol{h}$ and therefore predicted in a linear fashion through (46) (or its discretized equivalent). For example, assume we wish to minimize the predicted cost

$$J_{\mathrm{nonlin}} = J_{\mathrm{quad}} + l_{N_{\mathrm{p}}}(x_{N_{\mathrm{p}}}) + \sum_{i=0}^{N_{\mathrm{p}}-1} l(x_i), \tag{55}$$

subject to the stage and terminal constraints

$$c(x_i) + Du \leq 0, \qquad i = 0, \ldots, N_\mathrm{p} - 1, \tag{56a}$$
$$c_{N_\mathrm{p}}(x_i) \leq 0, \qquad i = 0, \ldots, N_\mathrm{p} - 1, \tag{56b}$$

where

$$J_\mathrm{quad} = x_{N_\mathrm{p}}^\top Q_{N_\mathrm{p}} x_{N_\mathrm{p}} + q_{N_\mathrm{p}}^\top x_{\mathrm{N}_p} + \sum_{i=0}^{N_\mathrm{p}-1} x_i^\top Q x_i + u_i^\top R u_i + q^\top x_i + r^\top u_i$$

is convex quadratic and $c : \mathbb{R}^n \to \mathbb{R}^{n_c}$, $c_{N_\mathrm{p}} : \mathbb{R}^n \to \mathbb{R}^{n_{c_\mathrm{p}}}$, $l : \mathbb{R}^n \to \mathbb{R}$ and $l_{N_\mathrm{p}} : \mathbb{R}^n \to \mathbb{R}$ are nonlinear functions. The mapping $\boldsymbol{h}$ is then set to

$$\boldsymbol{h}(x) = \begin{bmatrix} x \\ l(x) \\ l_{N_\mathrm{p}}(x) \\ c(x) \\ c_{N_\mathrm{p}}(x) \end{bmatrix}.$$

Replacing $\boldsymbol{h}$ by $\hat{y}$ in (55), the objective function J_nonlin translates to a *convex* quadratic in $(u_0, \ldots, u_{N_\mathrm{p}-1})$ and $(\hat{y}_0, \ldots, \hat{y}_{N_\mathrm{p}})$; similarly the stage and terminal constraints (56) translate to *affine* (and hence convex) inequality constraints on $(u_0, \ldots, u_{N_\mathrm{p}-1})$ and $(\hat{y}_0, \ldots, \hat{y}_{N_\mathrm{p}})$.

Note that polytopic constraints on control inputs can be encoded by selecting certain components of the vector function $c(x_i)$ equal to constant functions. For example, box constraints on u of the form $u \in [u_\mathrm{min}, u_\mathrm{max}]$ with $u_\mathrm{min} \in \mathbb{R}^m$ and $u_\mathrm{max} \in \mathbb{R}^m$ are encoded by selecting

$$c(x) = \begin{bmatrix} -u_\mathrm{max} \\ u_\mathrm{min} \\ \tilde{c}(x) \end{bmatrix}, \quad D = \begin{bmatrix} \mathbf{I} \\ -\mathbf{I} \\ \tilde{D} \end{bmatrix},$$

where $\tilde{c}(\cdot)$ and $\tilde{D}$ model additional state-input constraints.

No free lunch At this stage it should be emphasized that since $\hat{y}$ is only an approximation of the true output $\boldsymbol{h}$, the convex QP (54) is only an approximation of the nonlinear MPC problem with (55) as objective, and (45) and (56) as constraints. This is unavoidable at this level of generality since such nonlinear MPC problems are typically NP-hard whereas the convex QP (54) is polynomial time solvable. Nevertheless, as long as the prediction $\hat{y}$ is accurate, we also expect the solution of the linear MPC problem (54) to be close to the optimal solution of the nonlinear MPC problem, thereby resulting in near-optimal closed-loop performance.

5.1.1 Dense form Koopman MPC

Importantly for real-world deployment, in problem (54), the possibly high-dimensional variables z_i and $\hat{y}_i$ can be solved for in terms of the variable

$$\mathbf{u} := [u_0, \ldots, u_{N_\mathrm{p}-1}]^\top,$$

obtaining

$$\begin{array}{ll} \underset{\mathbf{u}\in\mathbb{R}^{mN_p}}{\text{minimize}} & \mathbf{u}^\top H_1 \mathbf{u}^\top + h^\top \mathbf{u} + z_0^\top H_2 \mathbf{u} \\ \text{subject to} & L\mathbf{u} + M z_0 \leq d \\ \text{parameter} & z_0 = \hat{\boldsymbol{\phi}}(x_{\text{current}}) \end{array} \tag{57}$$

for some matrices H_1, H_2, L, M and vectors h, d (explicit expressions in terms of the data of (54) are in the Appendix). Notice that once the product $z_0^\top H_2$ is evaluated, the cost of solving the optimization problem (57) is *independent* of the number of eigenfunctions N used. This is essential for practical applications since N can be large in order to ensure a small prediction error (9). The optimization problem (57) is a convex QP that can be solved by any of the generic packages for convex optimization (e.g., MOSEK or Gurobi) but also using highly tailored tools exploiting the specifics of the MPC formulation. In this work, we relied on the qpOASES package [7] that uses a homotopy-based active set method which is particularly suitable for dense-form MPC problems and effectively utilizes warm starting to reduce the closed-loop computation time.

The closed-loop operation of the Koopman MPC is summarized in Algorithm 2. Here we assume sampled-data operation, where the control input is computed every T_s seconds and held constant between the sampling times. We note, however, that the mapping

$$x_{\text{current}} \mapsto \mathbf{u}_0^\star,$$

where $\mathbf{u}_0^\star$ is the first component of the optimal solution $\mathbf{u}^\star = [\mathbf{u}_0^\star, \ldots, \mathbf{u}_{N_\text{p}-1}^\star]^\top$ to the problem (57), defines a feedback controller that can be evaluated at an arbitrary state $x \in \mathbb{R}^n$ at an arbitrary time.

Algorithm 2 Koopman MPC – closed-loop operation

1: **for** $k = 0, 1, \ldots$ **do**
2: Set $x_{\text{current}} = x(kT_\text{s})$ (current state of (45))
3: Compute $z_0 = \hat{\boldsymbol{\phi}}(x_{\text{current}})$
4: Solve (57) to get an optimal solution $\mathbf{u}^\star = [\mathbf{u}_0^\star, \ldots, \mathbf{u}_{N_\text{p}-1}^\star]^\top$
5: Apply $\mathbf{u}_0^\star$ to the system (45) for $t \in \big[kT_\text{s}\,,\,(k+1)T_\text{s}\big)$

6 Numerical examples

In the numerical examples we investigate the performance of the predictors on the Van der Pol oscillator and the damped Duffing oscillator. The two dynamical systems exhibit a very different behavior: The former is has a stable limit cycle whereas the latter two stable equilibria and an unstable equilibrium. However, interestingly but in line with the theory, we observe a very good performance of the predictors constructed for both systems, away from the limit cycle and singularities. On the Duffing system, we also investigate feedback control using the Koopman MPC, managing both transition between the two stable equilibria as well stabilization of the unstable one, in a purely data-driven and convex-optimization-based fashion.

6.1 Van der Pol oscillator

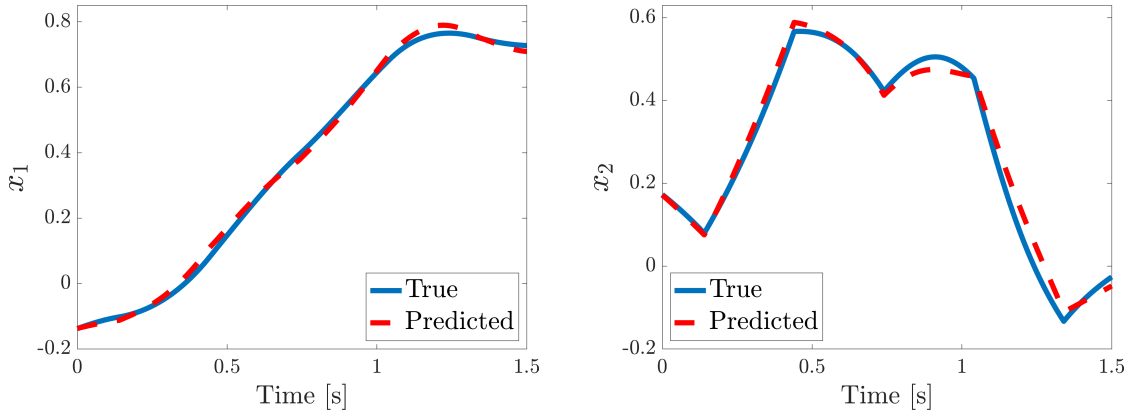

Figure 3: Van der Pol oscillator – Prediction with 20 eigenfunctions with optimized selection of eigenvalues and boundnary functions, a randomly chosen initial condition and square wave forcing.

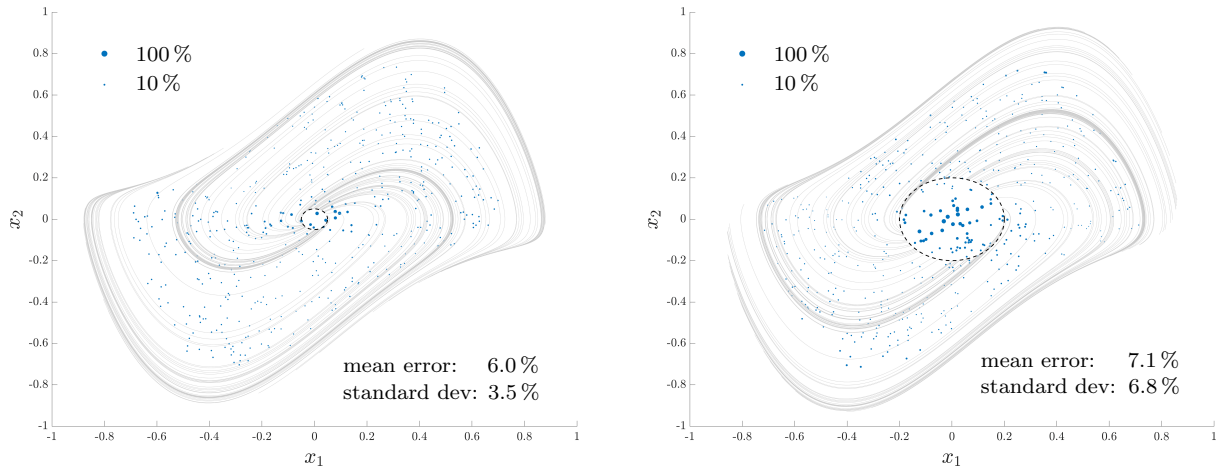

Figure 4: Van der Pol oscillator – Spatial distribution of the prediction error (controlled) with 20 eigenfunctions with optimized selection of eigenvalues and boundary functions. The trajectories used for construction of the eigenfunctions are depicted in grey. Their initial conditions were sampled from a circle of radius 0.05 (left pane) and 0.2 (right pane), both depicted in dashed black; neither circle is a non-recurrent surface for the dynamics (which is *not* required by the method). The error for each of the 500 initial conditions from the independently generated test set is encoded by the size of the blue marker.

In the first example, we consider the classical Van der Pol oscillator with forcing

$$\begin{aligned}\dot{x}_1 &= 2x_2\\ \dot{x}_2 &= -0.8x_1 + 2x_2 - 10x_1^2x_2 + u.\end{aligned}$$

We investigate the performance of the proposed predictors, both in controlled and uncontrolled (i.e., $u = 0$) settings. The function $\boldsymbol{h}$ to predict is the state itself, i.e. $\boldsymbol{h}(x) = x$, and hence the output $\hat{y}$ of (6) and (46) predicts the state of the system. We investigate the performance of the proposed predictor as a function of the number of eigenfunctions N used.

First, we construct the eigenfunction approximations as described in Section 4. The budget of N eigenfunctions is split equally among the components of $\boldsymbol{h}$, i.e., $N_1 = N_2 = N/2$. We generate a set of $M_\mathrm{t} = 100$ five second long trajectories sampled with a sampling period $T_\mathrm{s} = 0.01\,\mathrm{s}$ (i.e., $M_\mathrm{s} = 500$). The initial conditions of the trajectories are sampled uniformly over a circle of radius 0.05. As a first and natural choice of eigenvalues Λ, we utilize $\Lambda_\mathrm{lat} = \mathrm{lattice}_{d_\mathrm{lat}}(\frac{1}{T_\mathrm{s}} \log \Lambda_\mathrm{DMD})$, where Λ_DMD are the two eigenvalues obtained by applying the dynamic mode decomposition algorithm to the data set and

$$\mathrm{lattice}_d(\Lambda) = \Big\{ \sum_{k=1}^{p} \alpha_k \lambda_k \mid \lambda_k \in \Lambda,\ \alpha_k \in \mathbb{N},\ p \in \mathbb{N},\ \sum_{k=1}^{p} \alpha_k \leq d \Big\}. \tag{58}$$

The number of eigenvalues obtained in this way is $N_\mathrm{lat} = \binom{2+d_\mathrm{lat}}{d_\mathrm{lat}}$. For each value of N, we choose d_lat such that $N_\mathrm{lat} \geq N_1 = N_2 = N/2$ and use the first $N/2$ eigenvalues of Λ_lat in the algorithm of Section 4.2 for optimal choice of the boundary functions for each component of $\boldsymbol{h}$; we do not use regularization, i.e., the matrix $\mathbf{G}$ is obtained using (35) and (36). Second, we investigate the benefit of optimizing the eigenvalues as described in Section 4.2; the objective function (39) is minimized using local Newton-type algorithm implemented in Matlab's fmincon, with analytic gradients computed using (40) and initial condition given by Λ_lat.

The N eigenfunctions are computed on the data set using (24) and linear interpolation is used to define them on the entire state space. The C matrix is computed using (42) with $\bar{x}_i$ being the data used to construct the eigenfunctions plus a random noise uniformly distributed over $[-0.05, 0.05]^2$. This fully defines the linear predictor (6) in the uncontrolled setting. To get the B matrix in the controlled setting we generate a second data set with forcing. The initial conditions are the same as in the uncontrolled setting; the forcing is piecewise constant signal taking a random uniformly distributed value in $[-1, 1]$ in each sampling interval; the length of each trajectory is two seconds. The matrix B and its discrete counterpart B_d are then obtained using (52) and (53). In the controlled setting, we investigate the prediction performance for two control signals distinct from the signal used during identification. The first one is a square wave with unit amplitude and period 300 ms and the second one a sinusoid wave with unit amplitude and period 60 ms. Figure 3 shows the true and predicted trajectories for the randomly chosen initial condition $x_0 = [-0.1382, 0.1728]^\top$. Table 1 reports the prediction error over one second time interval as a function of N. The error is reported as the root mean square error

$$\text{Prediction error} = 100 \cdot \frac{\sqrt{\sum_k \|x_\mathrm{pred}(kT_s) - x_\mathrm{true}(kT_s)\|_2^2}}{\sqrt{\sum_k \|x_\mathrm{true}(kT_s)\|_2^2}}. \tag{59}$$

averaged over 500 randomly chosen initial conditions in the interior of the limit cycle. We observe that optimization of the eigenvalues brings about a significant improvement in performance, especially with a small number of eigenfunctions. We also observe that without control, the average prediciton error is close to 1 % with only 20 eigenfunctions.

Next, we investigate the spatial distribution of the prediction error as a function of the initial condition. We report results for two sets of data – the original set as described above and a

Table 1: Van der Pol oscilator – Prediction error averaged over 500 randomly chosen initial conditions as a function of the total number of eigenfunctions N, with and without optimization of the eigenvalues λ.

# of eigenfunctions N	4	8	12	16	20
λ not optimized					
Prediction error [uncontrolled]	100.4 %	95.6 %	51.34 %	13.31 %	5.44 %
Prediction error [square wave control]	102.2 %	115.7 %	51.2 %	14.5 %	8.3 %
Prediction error [sinus wave control]	101.3 %	97.0 %	53.3 %	13.8 %	6.2 %
λ optimized					
Prediction error [uncontrolled]	24.0 %	9.7 %	5.1 %	2.4 %	1.4 %
Prediction error [square wave control]	27.6 %	11.0 %	7.8 %	6.7 %	6.0 %
Prediction error [sinus wave control]	25.5 %	10.3 %	5.8 %	3.7 %	2.9 %

second set with initial conditions starting on a larger circle of radius 0.2 centered around the origin and trajectory length of three seconds. Figure 4 reports the results (for brevity we depict only the results with square wave control, the other case being qualitatively similar). We observe that choosing a smaller smaller circle to sample the initial conditions from results in a larger portion of the state-space being covered by the generated trajectories and hence smaller mean prediction error as well as its standard deviation.

6.2 Damped Duffing oscillator

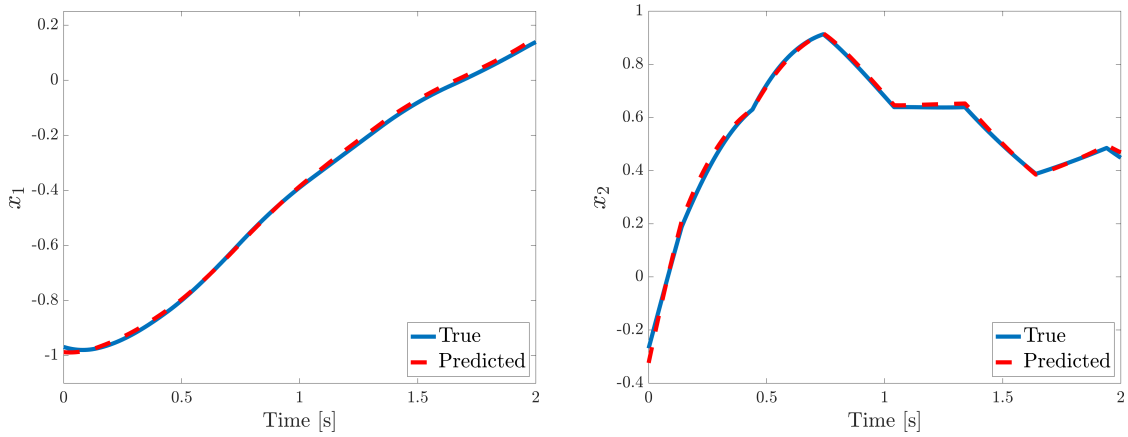

Figure 5: Duffing oscillator – Prediction with twenty eigenfunctions with optimized selection of eigenvalues and boundary functions, a randomly chosen initial condition and square wave forcing.

As our second example we consider the damped duffing oscillator with forcing

$$\begin{aligned}\dot{x}_1 &= x_2\\ \dot{x}_2 &= -0.5x_2 - x_1(4x_1^2 - 1) + 0.5u.\end{aligned}$$

The uncontrolled system (with $u = 0$) has two stable equilibria at $(-0.5, 0)$ and $(0.5, 0)$ as well as an unstable equilibrium at the origin.

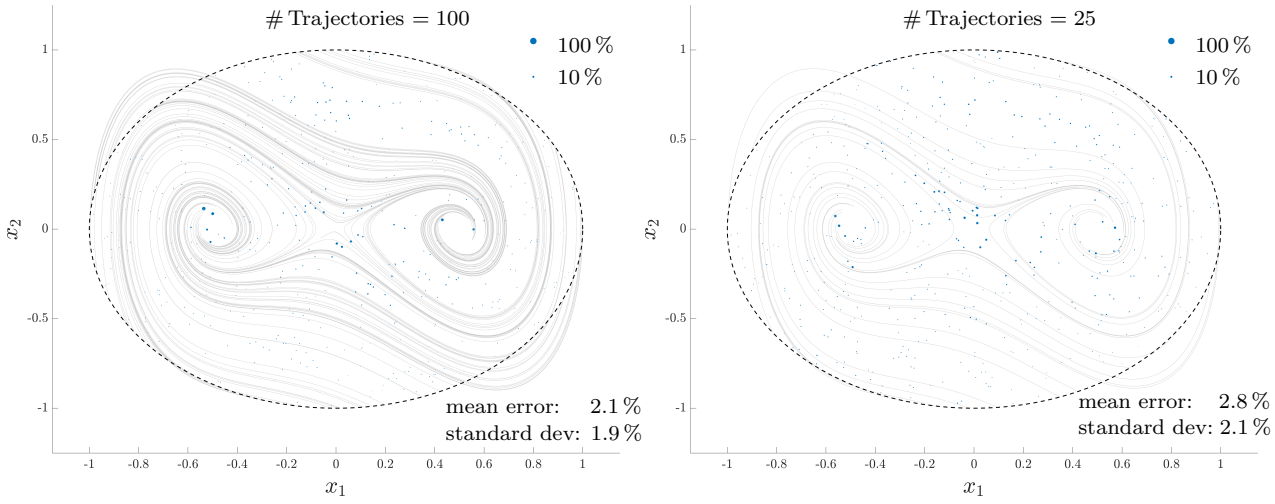

Figure 6: Damped Duffing oscillator – Spatial distribution of the prediction error (square wave control signal) with twenty eigenfunctions with optimized selection of eigenvalues and boundary functions. The trajectories used for construction of the eigenfunctions are depicted in grey; their initial conditions were sampled uniformly from the unit circle (dashed black) – note that the unit circle is *not* a non-recurrent surface for the dynamics and this property is *not* required by the method. The prediction error for each of the 500 initial conditions from the independently generated test set is encoded by the size of the blue marker.

Prediction First we investigate the performance of the proposed predictors on this system. The setup is very similar to the previous example. The function $\boldsymbol{h}$ to predict is the state itself, i.e., $\boldsymbol{h}(x) = x$, and we investigate the predictive performance as function of the number of eigenfunctions N used. First, we construct the eigenfunction approximations as described in Section 4. The budget of N eigenfunctions is split equally among the components of $\boldsymbol{h}$, i.e., $N_1 = N_2 = N/2$. We generate $M_\mathrm{t} = 100$ eight second long trajectories sampled with a sampling period $T_\mathrm{s} = 0.01$ (i.e., $M_\mathrm{s} = 800$). The initial conditions are chosen randomly from a uniform distribution on the unit circle. We note that the unit circle is *not* a non-recurrent surface for this system.

As before, for non-optimized eigenvalues we utilize $\Lambda_\mathrm{lat} = \mathrm{lattice}_{d_\lambda}(\frac{1}{T_\mathrm{s}} \log \Lambda_\mathrm{DMD})$, where Λ_DMD are the eigenvalues obtained from applying the DMD algorithm to the generated data set and $\mathrm{lattice}_{d_\lambda}(\cdot)$ is defined in (58). For each value of N, we choose d_lat such that $N_\mathrm{lat}(d_\mathrm{lat}) \geq N_1 = N_2 = N/2$ and use the first $N/2$ eigenvalues of Λ_lat in the algorithm of Section 4.2 for optimal choice of the boundary functions for each component of $\boldsymbol{h}$; we do not use regularization, i.e., the matrix $\mathbf{G}$ is obtained using (35) and (36). Second, we investigate the benefit of optimizing the eigenvalues as described in Section 4.2; the objective function (39) is minimized using local Newton-type algorithm implemented in Matlab's fmincon, with analytic gradients computed using (40) and initial condition given by Λ_lat.

The N eigenfunctions are computed on the data set using (24) and linear interpolation is used to define them on the entire state space. The matrix C is obtained using (41). To get the matrix B in the controlled setting we generate data with forcing. The initial conditions are the same as in the uncontrolled setting; the forcing is piecewise constant signal taking a random uniformly distributed values in $[-1, 1]$ in each sampling interval; the length of each

Table 2: Damped Duffing oscillator – Prediction error averaged over 500 randomly chosen initial conditions as a function of the total number of eigenfunctions N, with and without optimization of the eigenvalues λ.

# of eigenfunctions	12	16	20	24	28
λ **not optimized**					
Prediction error [uncontrolled]	26.0 %	22.5 %	8.3 %	2.7 %	1.1 %
Prediction error [square wave control]	26.0 %	22.1 %	7.6 %	2.7 %	1.5 %
Prediction error [sinus wave control]	25.2 %	21.2 %	7.3 %	2.4 %	1.1 %
λ **optimized**					
Prediction error [uncontrolled]	11.2 %	4.7 %	2.1 %	1.1 %	0.8 %
Prediction error [square wave control]	10.6 %	5.0 %	2.3 %	1.5 %	1.3 %
Prediction error [sinus wave control]	10.4 %	4.4 %	2.0 %	1.2 %	0.9 %

trajectory is two seconds. The matrix B and its discrete counterpart B_{d} are then obtained using (52) and (53). We investigate performance for two control signals distinct from the one used during identification. The first one is a square wave with unit amplitude and period 300 ms and the second one a sinusoid wave with unit amplitude and period 60 ms. Figure 5 shows a prediction for a randomly chosen initial condition in the controlled setting with the the square wave forcing.

Table 2 reports the prediction error (59) averaged over 500 initial conditions chosen randomly inside the unit circle for different values of N. We observe that optimization of eigenvalues brings about a significant improvement, especially with a small number of eigenfunctions. We also notice that with 28 eigenfucntions, the prediction error is around one percent in both the controlled and uncontrolled scenarios. Figure 6 then shows the spatial distribution of the prediction error over a one second prediction time interval, where we compare the predictors constructed from 100 trajectories and 25 trajectories. We observe that the prediction error deteriorates only very little, showing a remarkable robustness to the amount of data used (we tested this further and even with 10 trajectories, the mean error increased to only 5.4 %). We also observed the locations of the optimized eigenvalues to be very robust with respect to the number of trajectories used.

Feedback control Next, we apply the Koopman MPC developed in Section 5.1 to control the system with twenty eigenfunctions with optimimized selection of the eigenvalues and boundary functions. The goal is to track a piecewise constant reference signal, where we move from one stable equilibrium to the other $(0.5, 0) \mapsto (-0.5, 0)$, continue to the unstable saddle point at the origin and finish at $(0.25, 0)$ which is not an equilibrium point for the uncontrolled system but is stabilizable for the controlled one. The matrices Q and R in (54) were chosen $Q = \mathrm{diag}(1, 0.1)$ and $R = 10^{-4}$ and we imposed the constraint $u \in [-1, 1]$ on the control input. The prediction horizon was set to one second, i.e., $N_{\mathrm{p}} = 1.0/T_{\mathrm{s}} = 100$. The results are depicted in Figure 7; the tracking goal was achieved as desired. During the closed-loop operation, the MPC problem (57) was solved using the qpOASES solver [7]. The average computation time per time step was 0.59 ms, of which approximately 0.43 ms was spent evaluating the eigenfunction mapping $\hat{\boldsymbol{\phi}}$ (Step 3 of Algorithm 2) whereas the solution to the quadratic program (57) took on average only 0.16 ms. The evaluation of $\hat{\boldsymbol{\phi}}$ could be

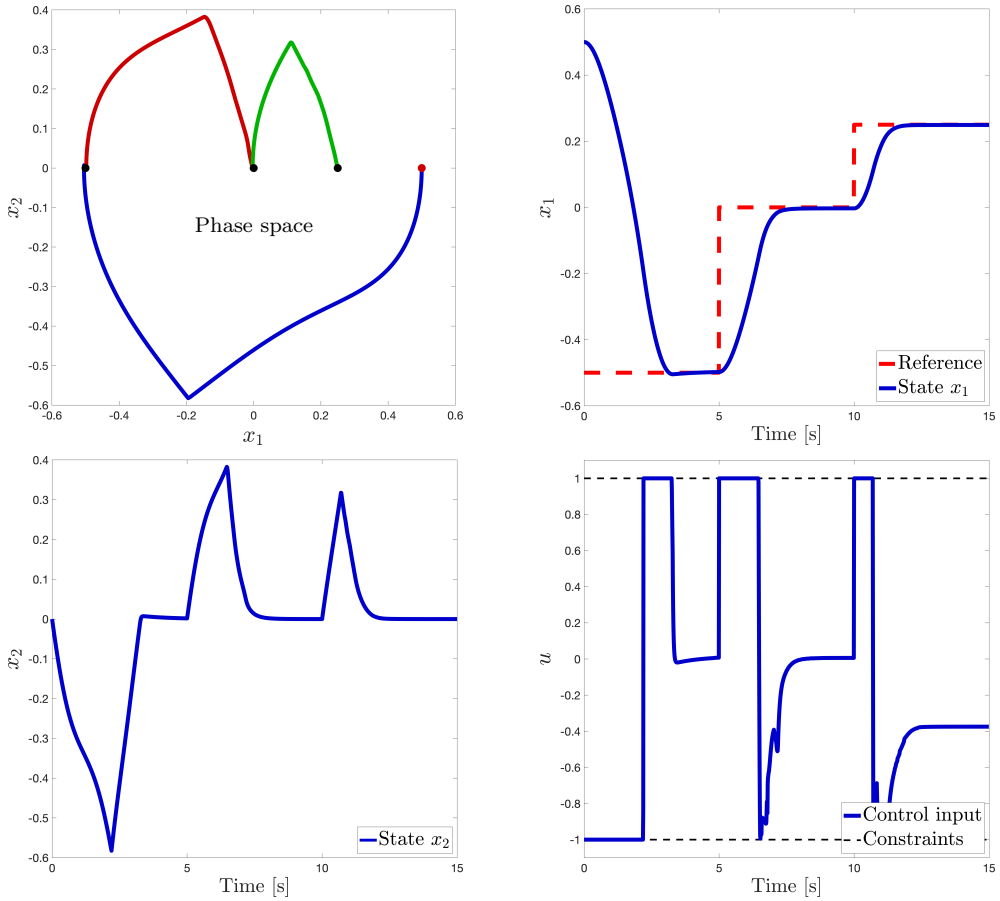

Figure 7: Duffing oscillator – feedback control using Koopman MPC.

significantly sped up by a more sophisticated interpolation implementation. We emphasize that the entire design was purely data driven and based only on linear model predictive control.

7 Conclusion

This work presented a systematic framework for data-driven learning of Koopman eigenfunctions in non-recurrent regions of the state-space. The method is geared toward prediction and control using *linear predictors*, allowing for feedback control and state estimation for nonlinear dynamical systems using established tools for linear systems that rely solely on convex optimization or simple linear algebra. The proposed method exploits the richness of the spectrum of the Koopman operator away from attractors to construct a large number of eigenfunctions in order to minimize the projection error of the state (or any other observable of interest) on the span of the eigenfunctions. The proposed method is purely data-driven and very simple, relying only on linear algebra and/or convex optimization, with computation complexity comparable to DMD-type methods and with very little user-input required; in particular only an interpolation method has to be selected, whereas the boundary functions and eigenvalues are determined from data using numerical optimization.

Future work will explore possible extensions of the method to the case where recurrences are

present.

Appendix

The matrices in (57) are given in terms of the data of (54) by

$$H_1 = \mathbf{R} + \mathbf{B}^\top \mathbf{Q} \mathbf{B}, \quad h = \mathbf{q}^\top \mathbf{B} + \mathbf{r}^\top, \quad H_2 = 2\mathbf{A}^\top \mathbf{Q} \mathbf{B},$$

$$L = \mathbf{F} + \mathbf{E}\mathbf{B}, \quad M = \mathbf{E}\mathbf{A}, \quad d = [b_0^\top, \dots, b_{N_\mathrm{p}}^\top]^\top,$$

where

$$\mathbf{A} = \begin{bmatrix} I \\ CA_\mathrm{d} \\ CA_\mathrm{d}^2 \\ \vdots \\ CA_\mathrm{d}^{N_\mathrm{p}} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB_\mathrm{d} & 0 & \dots & 0 \\ CA_\mathrm{d}B_\mathrm{d} & CB_\mathrm{d} & \dots & 0 \\ \vdots & \ddots & \ddots & \\ CA_\mathrm{d}^{N_p-1}B_\mathrm{d} & \dots & CA_\mathrm{d}B_\mathrm{d} & CB_\mathrm{d} \end{bmatrix}, \mathbf{F} = \begin{bmatrix} F_0 & 0 & \dots & 0 \\ 0 & F_1 & \dots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & F_{N_\mathrm{p}-1} \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

$$\mathbf{Q} = \mathrm{diag}(Q_0, \dots, Q_{N_\mathrm{p}}), \quad \mathbf{R} = \mathrm{diag}(R_0, \dots, R_{N_\mathrm{p}-1}),$$
$$\mathbf{E} = \mathrm{diag}(E_0, \dots, E_{N_\mathrm{p}}), \, \mathbf{q} = [q_0, \dots, q_{N_\mathrm{p}}], \quad \mathbf{r} = [r_0, \dots, r_{N_\mathrm{p}-1}].$$

8 Acknowledgments

The authors would like to thank Corbinian Schlosser for pointing out to us a new proof of Theorem 2, which both simplifies our original proof as well as eliminates one of the assumptions. The authors would also like to thank Shai Revzen for insightful discussions. This research was supported by the ARO-MURI grant W911NF17-1-0306.

References

[1] H. Arbabi, M. Korda, and I. Mezić. A data-driven Koopman model predictive control framework for nonlinear flows. *arXiv preprint arXiv:1804.05291*, 2018.

[2] H. Arbabi and I. Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017.

[3] V. I. Arnold. *Ordinary differential equations*. Springer-Verlag, third edition, 1984.

[4] E. M. Bollt, Q. Li, F. Dietrich, and I. Kevrekidis. On matching, and even rectifying, dynamical systems through koopman operator eigenfunctions. *SIAM Journal on Applied Dynamical Systems*, 17(2):1925–1960, 2018.

[5] B. W. Brunton, L. A. Johnson, J. G. Ojemann, and J. N. Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of neuroscience methods*, 258:1–15, 2016.

[6] Z. Drmač, I. Mezić, and R. Mohr. On least squares problems with certain vandermonde–khatri–rao structure with applications to dmd. *arXiv preprint arXiv:1811.12562*, 2018.

[7] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[8] M. Georgescu and I. Mezić. Building energy modeling: A systematic approach to zoning and model reduction using Koopman mode analysis. *Energy and buildings*, 86:794–802, 2015.

[9] N. Govindarajan, R. Mohr, S. Chandrasekaran, and I. Mezić. On the approximation of koopman spectra for measure preserving transformations. *arXiv preprint arXiv:1803.03920*, 2018.

[10] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *arXiv preprint arXiv:1707.01146*, 2017.

[11] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315, 1931.

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

[13] M. Korda and I. Mezić. On convergence of extended dynamic mode decomposition to the Koopman operator. *Journal of Nonlinear Science*, 28(2):687–710, 2018.

[14] M. Korda, M. Putinar, and I. Mezić. Data-driven spectral analysis of the Koopman operator. *Applied and Computational Harmonic Analysis*, 2018.

[15] M. Korda, Y. Susuki, and I. Mezić. Power grid transient stabilization using Koopman model predictive control. *IFAC-PapersOnLine*, 51(28):297–302, 2018.

[16] K. Küster, R. Derndinger, and R. Nagel. *The Koopman Linearization of Dynamical Systems*. PhD thesis, Diplomarbeit, März 2015, Arbeitsbereich Funktionalanalysis, Mathematisches Institut, Eberhard-Karls-Universität Tübingen, 2015.

[17] A. Mauroy and J. Goncalves. Linear identification of nonlinear systems: A lifting technique based on the koopman operator. *arXiv preprint arXiv:1605.04457*, 2016.

[18] I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.

[19] I. Mezić. Koopman operator spectrum and data analysis. *arXiv preprint arXiv:1702.07597*, 2017.

[20] I. Mezić and A. Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1):101–133, 2004.

[21] R. Mohr and I. Mezić. Construction of eigenfunctions for scalar-type operators via laplace averages with connections to the koopman operator. *arXiv preprint arXiv:1403.6559*, 2014.

[22] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[23] J. L. Proctor, S. L. Brunton, and J. N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[24] F. Raak, Y. Susuki, I. Mezić, and T. Hikihara. On Koopman and dynamic mode decompositions for application to dynamic data with low spatial dimension. In *IEEE 55th Conference on Decision and Control (CDC)*, pages 6485–6491, 2016.

[25] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641(1):115–127, 2009.

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

[27] G. W. Stewart. On the perturbation of pseudo-inverses, projections and linear least squares problems. *SIAM review*, 19(4):634–662, 1977.

[28] A. Surana. Koopman operator based observer synthesis for control-affine nonlinear systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6492–6499. IEEE, 2016.

[29] A. Surana and A. Banaszuk. Linear observer synthesis for nonlinear systems using Koopman operator framework. In *IFAC Symposium on Nonlinear Control Systems (NOLCOS)*, 2016.

[30] A. Surana, M. O. Williams, M. Morari, and A. Banaszuk. Koopman operator framework for constrained state estimation. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 94–101. IEEE, 2017.

[31] N. Takeishi, Y. Kawahara, and T. Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.

[32] M. O. Williams, M. S. Hemati, S. T. M. Dawson, I. G. Kevrekidis, and C. W. Rowley. Extending data-driven Koopman analysis to actuated systems. In *IFAC Symposium on Nonlinear Control Systems (NOLCOS)*, 2016.

[33] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[34] H. Wu and F. Noé. Variational approach for learning Markov processes from time series data. *arXiv preprint arXiv:1707.04659*, 2017.

[35] H. Wu, F. Nüske, F. Paul, S. Klus, P. Koltai, and F. Noé. Variational Koopman models: Slow collective variables and molecular kinetics from short off-equilibrium simulations. *The Journal of Chemical Physics*, 146(15):154104, 2017.

[36] A. Y. Yang, Z. Zhou, A. G. Balasubramanian, S. S. Sastry, and Y. Ma. Fast ℓ_1-minimization algorithms for robust face recognition. *IEEE Transactions on Image Processing*, 22(8):3234–3246, 2013.