# Extended Dynamic Mode Decomposition with Learned Koopman Eigenfunctions for Prediction and Control

Carl Folkestad*,[1], Daniel Pastor*,[1], Igor Mezic[3], Ryan Mohr[2], Maria Fonoberova[2], and Joel Burdick[1]

*Abstract*— **This paper presents a novel learning framework to construct Koopman eigenfunctions for unknown, nonlinear dynamics using data gathered from experiments. The learning framework can extract spectral information from the full nonlinear dynamics by learning the eigenvalues and eigenfunctions of the associated Koopman operator. We then exploit the learned Koopman eigenfunctions to learn a lifted linear state-space model. To the best of our knowledge, our method is the first to utilize Koopman eigenfunctions as lifting functions for EDMD-based methods. We demonstrate the performance of the framework in state prediction and closed loop trajectory tracking of a simulated cart pole system. Our method is able to significantly improve the controller performance while relying on linear control methods to do nonlinear control.**

## I. Introduction

A key step in developing a high performance robotic application is the modeling of the robot's mechanics. Standard modelling and identification techniques require extensive knowledge of the system and laborious system identification procedures [1]. Moreover, one must still design a controller for the nonlinear mechanics that are identified. While methods exists to show stability and safety of nonlinear systems [2], [3], linearized models around a desired trajectory with PID or LQR control are often used in practice.

*Learning* can capture the salient aspects of a robot's complex mechanics and environmental interactions. Gaussian process dynamical systems models [4] can identify nonlinear affine control models in a non-parametric way. Alternatively, spectrally normalized neural networks [5] can fit dynamics models with stability guarantees. Yet, effective nonlinear control design incorporating state and actuator constraints after identifying the model can be challenging. Deep neural networks for control Lyapunov function augmentation [6] can be used for control design with different types of constraints but learns a task-specific augmentation that cannot be used for other objectives. Similarly, model-free reinforcement learning (MFRL) [7] learns feedback policies that implicitly incorporate the robot's dynamics. However, sample efficiency is very low. Moreover, while safety during MFRL is now possible [8], [9], one cannot yet guarantee that learned policies will satisfy performance requirements or state and actuator limits.

Our work contributes to Koopman inspired modelling and identification techniques, which have received substantial recent attention [10], [11]. In particular, the Dynamic Mode Decomposition (DMD) and extended DMD (EDMD) methods have emerged as efficient numerical algorithms to identify finite dimensional approximations of the Koopman operator associated with the system dynamics [12], [13]. The methods are easy to implement, mainly relying on least squares regression, and computationally and mathematically flexible, enabling numerous extensions and applications [14].

For example, DMD-based methods have been successfully used in the field of fluid mechanics to capture low-dimensional structure in complex flows [15], in robotics for external perturbation force detection [16], and in neuroscience to identify dynamically relevant features in ECOG data [17]. More recently, Koopman-style modeling has been extended to *controlled* nonlinear systems [18], [19]. This is particularly interesting as EDMD can be used to approximate nonlinear control systems by a lifted state space model. As a result, well developed linear control design methods such as robust, adaptive, and model predictive control (MPC) [20] can be utilized to design nonlinear controllers.

Typically, EDMD-methods employ a dictionary of functions used to lift the state variables to a space where the dynamics are approximately linear. However, if not chosen carefully, the time evolution of the dictionary functions cannot be described by a linear combination of the other functions in the dictionary. This results in error accumulation when the lifted state space model is used for prediction, potentially causing significant prediction performance degradation. To mitigate this problem we develop a learning framework that can extract spectral information from the full nonlinear dynamics by learning the eigenvalues and eigenfunctions of the associated Koopman operator. Limited attention has been given to constructing eigenfunctions from data. Sparse identification techniques have been used to identify approximate eigenfunctions [21] but rely on defining an appropriate candidate function library. Other previous methods for identifying Koopman eigenfunctions (e.g., [22]) depend upon assumptions that are problematical for robotic systems: the ID data is gathered while the robot operates under open loop controls, which can lead to catastrophic system damage.

This paper presents a novel learning framework, *Koopman Eigenfunction Extended Dynamic Mode Decomposition* (KEEDMD), to construct Koopman eigenfunctions for unknown, nonlinear dynamics using a data gathered from experiments. We then exploit the learned Koopman eigenfunctions to learn a lifted linear state-space model. To the best of our knowledge, our method is the first to utilize Koopman eigenfunctions as lifting functions for EDMD-

*Both authors contributed equally, [1] Division of Engineering and Applied Sciences, California Institute of Technology, Pasadena, CA 91125, USA, [2] AIMDyn Inc., [3] University of California, Santa Barbara

Corresponding authors: carl.folkestad@caltech.edu, dpastorm@caltech.edu

based methods. Furthermore, we demonstrate that the identified model can readily be used with MPC [23] on simulated experiments.

### A. Notation

We denote the space of all continuous functions on some domain $\mathcal{X} \subset \mathbb{R}^d$ as $\mathcal{C}(\mathcal{X})$, the Jacobian of the function $f(\mathbf{x})$ evaluated at $\mathbf{x} = \mathbf{a}$ is denoted $\mathbf{D}f(a)$. $\mathbb{N}_0$ is the set of natural numbers including zero. $I$ is the identity matrix of appropriate dimensions. $\delta_{jk}$ is the kronecker delta, $\delta_{jk} = 1$ if and only if $j = k$.

## II. Preliminaries on Koopman Operator Theory

This section briefly reviews basic facts about the Koopman operator, and then summarizes key results that form the theoretical underpinnings for the Koopman eigenfunction learning methodology presented in Section III.

### A. The Koopman Operator

Consider the autonomous dynamical system:

$$\dot{\mathbf{x}} = f(\mathbf{x}) = A\mathbf{x} + v(\mathbf{x}) \tag{1}$$

with state $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ and $f(\cdot)$ Lipschitz continuous on $\mathcal{X}$. We assume that system (1) has a fixed point at the origin: $f(0) = 0$. For a system with a single attractor in $\mathcal{X}$ this can be achieved without loss of generality by a change of coordinates. The flow of this dynamical system is denoted by $S_t(\mathbf{x})$ and is defined as

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

for all $\mathbf{x} \in \mathcal{X}$ and all $t \geq 0$. The *Koopman operator semigroup* $(U_t)_{t \geq 0}$, hereafter denoted as the *Koopman operator*, is defined as

$$U_t \gamma = \gamma \circ S_t \tag{3}$$

for all $\gamma \in \mathcal{C}(\mathcal{X})$, where $\circ$ denotes function composition. Each element of the Koopman operator maps continuous functions to continuous functions, $U_t : \mathcal{C}(\mathcal{X}) \to \mathcal{C}(\mathcal{X})$. Crucially, each $U_t$ is a *linear* operator. An *eigenfunction* of the Koopman operator associated to an eigenvalue $e^{\lambda} \in \mathbb{C}$ is any function $\phi \in \mathcal{C}(\mathcal{X})$ that defines a coordinate evolving linearly along the flow of (1) satisfying

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

### B. Construction of Eigenfunctions for Nonlinear Dynamics

For any sufficiently smooth autonomous dynamical system that is asymptotically stable to a fixed point, Koopman eigenfunctions can be constructed by first finding the eigenfunctions of the system linearization around the fixed point and then composing them with a diffeomorphism [24]. To see this, consider asymptotically stable dynamics of the form (1). The linearization of the dynamics around the origin is

$$\dot{\mathbf{y}} = \mathbf{D}f(0)\mathbf{y} = \hat{A}\mathbf{y}, \ \mathbf{y} \in \mathcal{Y} \tag{5}$$

The following proposition describes how to construct eigenfunction-eigenvalue pairs for the linearized system (5).

**Proposition 1.** *Let $\hat{A}_1$ denote the linearization (5) of the nonlinear system (1) with $\mathcal{Y}$ scaled into the unit hypercube, $\mathcal{Y}_1 \subset \mathcal{Q}_1$, and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_d\}$ be a basis of the eigenvectors of $\hat{A}_1$ corresponding to nonzero eigenvalues $\{\lambda_1, \ldots, \lambda_d\}$. Let $\{\mathbf{w}_1, \ldots, \mathbf{w}_d\}$ be the adjoint basis to $\{\mathbf{v}_1, \ldots, \mathbf{v}_d\}$ such that $\langle \mathbf{v}_j, \mathbf{w}_k \rangle = \delta_{jk}$ and $\mathbf{w}_j$ is an eigenvector of $\hat{A}_1^*$ at eigenvalue $\bar{\lambda}_j$. Then, the linear functional*

$$\psi_j(\mathbf{y}) = \langle \mathbf{y}, \mathbf{w}_j \rangle \tag{6}$$

*is a nonzero eigenfunction of $U_{\hat{A}_1}$, the Koopman operator associated to $\hat{A}_1$. Furthermore, for any tuple $(m_1, \ldots, m_d) \in \mathbb{N}_0^d$*

$$\left( \sum_{j=1}^{d} \lambda_j^{m_j}, \prod_{j=1}^{d} \psi_j^{m_j} \right) \tag{7}$$

*is an eigenpair of the Koopman operator $U_{\hat{A}_1}$.*

*Proof:* A less formal description of the results in the proposition and associated proofs are described in [24], Example 4.6. By utilizing inner-product properties, $\psi_j$ is an eigenfunction of $U_{\hat{A}}$ as described in (4) since

$$\begin{aligned}(U_t \psi_j)(\mathbf{y}) &= U_t \langle \mathbf{y}, \mathbf{w}_j \rangle = \langle \mathbf{y}, U_t^* \mathbf{w}_j \rangle = \langle \mathbf{y}, e^{\bar{\lambda}_j} \mathbf{w}_j \rangle \\ &= e^{\lambda_j} \langle \mathbf{y}, \mathbf{w}_j \rangle = e^{\lambda_j} \psi_j(\mathbf{y})\end{aligned}$$

By scaling the state-space such that $\mathcal{Y}_1 \subset \mathcal{Q}_1$, the linear eigenfunctions (6) form a vector space on $\mathcal{Y}_1$ that is closed under point-wise products. The construction of arbitrarily many eigenpairs (7) therefore follows from the semi-group property of eigenfunctions (see [11], Prop. 5). □

In the following we denote the linear functionals (6) as *principal eigenfunctions*. The eigenfunctions for the Koopman operator associated with the linearized dynamics can be used to construct eigenfunctions associated with the Koopman operator of the nonlinear dynamics through the use of a *conjugacy map*, as described in the following proposition.

**Proposition 2.** *Assume that the nonlinear system (1) is topologically conjugate to the linearized system (5) via the diffeomorphism $h : \mathcal{X} \to \mathcal{Y}$. Let $B \in \mathcal{X}$ be a simply connected, bounded, positively invariant open set in $\mathcal{X}$ such that $h(B) \subset Q_r \subset \mathcal{Y}$, where $Q_r$ is a cube in $\mathcal{Y}$. Scaling $Q_r$ to the unit cube $Q_1$ via the smooth diffeomorphism $g : Q_r \to Q_1$ gives $(g \circ h)(B) \subset Q_1$. Then, if $\psi$ is an eigenfunction for $U_{\hat{A}_1}$ at $e^{\lambda}$, then $\psi \circ g \circ h$ is an eigenfunction for $U_f$ at eigenvalue $e^{\lambda}$, where $U_f$ is the Koopman operator associated with the nonlinear dynamics (1).*

*Proof:* See [11], Proposition 7. □

The following extension of the Hartman-Grobman theorem guarantees the existence of the diffeomorphism, $h$ described in Proposition 2, between the linearized and nonlinear systems in the entire basin of attraction of a fixed point, for sufficiently smooth dynamics.

**Theorem 3.** *Consider the system (1) with $v(\mathbf{x}) \in \mathcal{C}^2(\mathcal{X})$. Assume that matrix $A \in \mathbb{R}^{d \times d}$ is Hurwitz, i.e., all of its eigenvalues have negative real parts. So, the fixed point $\mathbf{x} =$*

$\mathbf{0}$ *is exponentially stable and let* $\Omega$ *be its basin of attraction. Then* $\exists h(\mathbf{x}) \in \mathcal{C}^1(\Omega) : \Omega \to \mathbb{R}^d$*, such that*

$$\mathbf{y} = c(\mathbf{x}) = \mathbf{x} + h(\mathbf{x}) \tag{8}$$

*is a* $\mathcal{C}^1$ *diffeomorphism with* $\mathbf{D}c(\mathbf{0}) = I$ *in* $\Omega$ *and satisfies* $\dot{\mathbf{y}} = A\mathbf{y}$.

*Proof:* See [25], Theorem 2.3. □

## III. Data-driven Koopman Eigenfunctions for Unknown Nonlinear Dynamics

Using the results of the previous section, we now develop a data-driven approach to learn the diffeomorphism $h(x)$ described in Proposition 2 and Equation 8, resulting in a methodology for constructing Koopman eigenfunctions from data.

### A. Modeling Assumptions

We consider the dynamical system

$$\dot{\mathbf{x}} = a(\mathbf{x}) + B\mathbf{u} \tag{9}$$

where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, $a(\mathbf{x}) : \mathcal{X} \to \mathcal{X}$, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$, $B \in \mathbb{R}^{d \times m}$, and where $a(\mathbf{x})$ and $B$ are unknown. The proposed method does not allow a state-dependent $B$ matrix but current work addresses this issue [26]. We assume that we have access to a nominal linear model

$$\dot{\mathbf{x}} = A_{nom}\mathbf{x} + B_{nom}\mathbf{u} \tag{10}$$

where $\mathbf{x} \in \Omega \subset \mathcal{X} \subset \mathbb{R}^d$, $A_{nom} \in \mathbb{R}^{(d \times d)}$, $B_{nom} \in \mathbb{R}^{(d \times m)}$, $\mathbf{u} \in \mathcal{U}$ and an associated nominal linear feedback controller $\mathbf{u}^{nom} = K_{nom}\mathbf{x}$ that stabilizes the system (9) to the origin in a region of attraction $\Omega$ around the origin. The nominal model (10) can for example be obtained from first principles modeling, parameter identification techniques and linearization of the constructed model around the fixed point if needed.

### B. Constructing Eigenfunctions from Data

Algorithm 1 constructs Koopman eigenfunctions from data, based on the foundations introduced in Section II-B. $M_t$ trajectories of fixed length $T$ are executed from initial conditions $\mathbf{x}_0^j \in \Omega$ $j = 1, \dots, M_t$, and are guided by the nominal control law $\mathbf{u}^{nom}$. The system's states and control actions are sampled at a fixed interval $\Delta t$, resulting in a data set

$$\mathcal{D} = \left( \left( \mathbf{x}_k^j, \mathbf{u}_k^j \right)_{k=0}^{M_s} \right)_{j=1}^{M_t} \tag{11}$$

where $M_s = T/\Delta t$. Variable length trajectories and sampling rates can be implemented with minor modifications.

Under the nominal control law, Koopman eigenfunctions for the nominal linearized model (10) can be constructed as in Proposition 1 using the eigenvectors and eigenvalues of the closed loop dynamics matrix $A_{cl} = A_{nom} + B_{nom}K_{nom}$. I.e. let $\mathcal{Q}_r$ be a hypercube of radius $r$ such that $\mathcal{X} \subset \mathcal{Q}_r$, a scaling function $g : \mathcal{Q}_r \to \mathcal{Q}_1$ can then be constructed (by scaling each coordinate) to get the scaled dynamics matrix

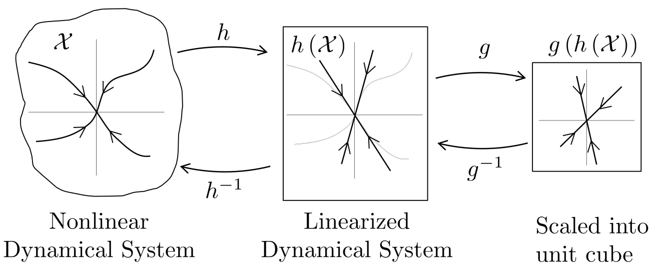



Fig. 1: Chain of topological conjugacies used to construct eigenfunctions, adapted from [24].

$A_{cl,1}$. Furthermore, let $\{\mathbf{v_j}\}_{j=1}^d$ be a basis of eigenvectors of $A_{cl,1}$ with corresponding eigenvalues $\{\lambda_j\}_{j=1}^d$ and let $\{\mathbf{w_j}\}_{j=1}^d$ be the adjoint basis to $\{\mathbf{v_j}\}_{j=1}^d$. Then $\psi_j(\mathbf{y}) = \langle \mathbf{y}, \mathbf{w}_j \rangle$ is an eigenfunction of $U_{A_{cl,1}}$ with eigenvalue $e_j^{\lambda}$ and we can construct an arbitrary number of eigenpairs using the product rule (7).

The eigenfunction construction for the linearized system only relies on the nominal model. To construct Koopman eigenfunctions for the true nonlinear dynamical system, we aim to learn the diffeomorphism (8) between the linearized model (10) and the true dynamics (9), see Figure 1. This diffeomorphism is guaranteed to exist in the entire basin of attraction $\Omega$ by Theorem 3. Let $\mathcal{H}_h$ be a class of continuous nonlinear function mapping $\mathbb{R}^d$ to $\mathbb{R}^d$. The diffeomorphism is found by solving the following optimization problem:

$$\begin{aligned} \min_{h \in \mathcal{H}_h} & \sum_{k=1}^{M_t} \sum_{j=1}^{M_s} (\dot{\mathbf{x}}_k^j + \dot{h}(\mathbf{x}_k^j) - A_{cl}(\mathbf{x}_k^j + h(\mathbf{x}_k^j)))^2 \\ \text{s.t.} & \quad \mathbf{D}h(\mathbf{0}) = \mathbf{0} \end{aligned} \tag{12}$$

which is a direct transformation of Theorem 3 into the setting with unknown nonlinear dynamics. The form of problem (12) is found by minimizing the squared loss $\dot{\mathbf{y}}_k - A_{cl}\mathbf{y}_k$ over all data pairs, substituting $\mathbf{y} = \mathbf{x} + h(\mathbf{x})$, and adding the constraint $\mathbf{D}a(\mathbf{0}) = I$ results in the formulated optimization problem (12).

We next formulate (12) as a general supervised learning problem. Consider the data set of input-output pairs $\mathcal{D}_h =$

**Algorithm 1** Data-driven Koopman Eigenpair Construction

**Require:** Data set $\mathcal{D} = \left((\mathbf{x}_k^j, \mathbf{u}_k^j)_{k=0}^{M_s}\right)_{j=1}^{M_t}$, nominal model matrices $A_{nom}$, $B_{nom}$, nominal control gains $K_{nom}$, number of lifting functions $N$, $N$ power combinations $(m_1^{(i)}, \dots, m_d^{(i)}) \in \mathbb{N}_0^d, i = 1, \dots, N$

1: Construct principal eigenpairs for the linearized dynamics: $(\lambda_j, \psi_j(\mathbf{y})) \leftarrow (\lambda_j, \langle \mathbf{y}, \mathbf{w}_j \rangle), \quad j = 1, \dots, n$
2: Construct $N$ eigenpairs from the principal eigenpairs: $(\tilde{\lambda}_i, \tilde{\psi}_i) \leftarrow \left( \sum_{j=1}^d \lambda_j^{m_j^{(i)}}, \prod_{j=1}^d \psi_j^{m_j^{(i)}} \right), \quad i = 1, \dots, N$
3: Fit diffeomorphism estimator: $h(\mathbf{y}) \leftarrow \text{ERM}(\mathcal{H}_h, \mathcal{L}_h, \mathcal{D})$
4: Construct scaling function: $g(\mathbf{y}) \leftarrow g : \mathcal{Q}_r \to \mathcal{Q}_1$
5: Construct $N$ eigenpairs for the nonlinear dynamics: $(\tilde{\lambda}_i, \phi_i) \leftarrow (\tilde{\lambda}_i, \tilde{\psi}_i(g(h(\mathbf{y})))), \quad i = 1, \dots, N$

**Output:** $\Lambda = \text{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_N), \quad \boldsymbol{\phi} = [\phi_1, \dots, \phi_N]^T$

$$\textbf{Position dynamics:} \quad \min_{B_{\mathbf{p}}\in\mathbb{R}^{(d/2)\times m}} ||\mathbf{y_p} - X_{\mathbf{p}} B_{\mathbf{p}}^T||_2^2, \quad X_{\mathbf{p}} = [U], \quad \mathbf{y_p} = [\dot{P} - IV] \tag{13a}$$

$$\textbf{Velocity dynamics:} \quad \min_{A_{\mathbf{v}}\in\mathbb{R}^{(d/2)\times(n+N)}, B_{\mathbf{v}}\in\mathbb{R}^{(d/2)\times m}} ||\mathbf{y_v} - X_{\mathbf{v}}[A_{\mathbf{v}}\ B_{\mathbf{v}}]^T||_2^2, \quad X_{\mathbf{v}} = [P\ V\ \Phi\ U], \quad \mathbf{y_v} = [\dot{V}] \tag{13b}$$

$$\textbf{Eigenfunction dynamics:} \quad \min_{B_{\boldsymbol{\phi}}\in\mathbb{R}^{N\times m}} ||\mathbf{y}_{\boldsymbol{\phi}} - X_{\boldsymbol{\phi}} B_{\boldsymbol{\phi}}^T||_2^2, \quad X_{\boldsymbol{\phi}} = [U - U_{nom}], \quad \mathbf{y}_{\boldsymbol{\phi}} = [\dot{\Phi} - \Lambda\Phi] \tag{13c}$$

---

$\{(\mathbf{x}_k, \dot{\mathbf{x}}_k), \dot{\mathbf{x}}_k - A_{cl}\mathbf{x}_k\}_{k=1}^{M_s \cdot M_t}$, constructed from the state measurements (perhaps by calculating numerical derivatives $\dot{\mathbf{x}}_k^j$ as needed), and aggregated a data matrix. The class $\mathcal{H}_h$ can be any function class suitable for supervised learning (e.g. deep neural networks) as long as the Jacobian of the function $h(\mathbf{x}) \in \mathcal{H}_h$ w.r.t. the input can be readily calculated. Assuming $h(\mathbf{x}) \in \mathcal{H}_h$ we define the loss function

$$\begin{aligned}
&\mathcal{L}_h(\mathbf{x}, \dot{\mathbf{x}}, A_{cl}\mathbf{x} - \dot{\mathbf{x}}) = \\
&\qquad ||\dot{h}(\mathbf{x}) - A_{cl}h(\mathbf{x}) - (A_{cl}\mathbf{x} - \dot{\mathbf{x}})||^2 + \alpha||\mathbf{D}h(\mathbf{0})||^2 \\
&\quad = ||\mathbf{D}h(\mathbf{x})\dot{\mathbf{x}} - A_{cl}h(\mathbf{x}) - (A_{cl}\mathbf{x} - \dot{\mathbf{x}})||^2 + \alpha||\mathbf{D}h(\mathbf{0})||^2
\end{aligned} \tag{14}$$

where parameter $\alpha$ penalizes the violation of constraint (12). The supervised learning goal is to select a function in $\mathcal{H}_h$ through empirical risk minimization (ERM):

$$\min_{h\in\mathcal{H}_h} \frac{1}{M_s \cdot M_t} \sum_{k=1}^{M_s\cdot M_t} \mathcal{L}_h(\mathbf{x}_k, \dot{\mathbf{x}}_k, A_{cl}\mathbf{x}_k - \dot{\mathbf{x}}_k)\ . \tag{15}$$

Finally, with function $h$ identified from ERM (15), Proposition 2 implies that the Koopman eigenfunctions for the unknown dynamics under the nominal control law can be constructed from the eigenfunctions of the linearized system by the function composition:

$$\phi_j(\mathbf{x}) = \tilde{\psi}_j(g(h(\mathbf{x}))) \tag{16}$$

where $g$ is the scaling function ensuring that the basin of attraction $\Omega$ is scaled to lie within the unit hypercube $Q_1$ and $\tilde{\psi}_j$ is an eigenfunction for the linearized system with associated eigenvalue $\tilde{\lambda}_j$ constructed with (7).

Importantly, because the diffeomorphism is learned from data, it may not perfectly capture the underlying diffeomorphism over all of $\Omega$, and thus the eigenfunctions for the unknown dynamics are approximate. The error arises from the fact that the ERM problem is underdetermined resulting in the possibility of multiple approximations with equal loss while failing to capture the underlying diffeomorphism. This is especially an issue when encountering states and state time derivatives not reflected in the training data and introduces a demand for exploratory control inputs to cover a larger region of the state space of interest. This can be achieved by introducing a random perturbation of the control action deployed on the system and is akin to persistence of excitation in adaptive control [27]. To understand these effects, state dependent model error bounds are needed but they are out of the scope of this paper.

## IV. Koopman Eigenfunction Extended Dynamic Mode Decomposition

To use the constructed Koopman eigenfunctions for prediction and control, we develop an EDMD-based method to build a linear model in a lifted space. Since this method exploits the structure of the Koopman eigenfunctions, it is dubbed *Koopman Eigenfunction Extended Dynamic Mode Decomposition* (KEEDMD). We construct $N$ eigenfunctions $\{\phi_j\}_{j=1}^N$ with associated eigenvalues $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_N)$ as outlined in Section III and define the lifted state as

$$\mathbf{z} = [\mathbf{x}, \boldsymbol{\phi}(\mathbf{x})]^T \tag{17}$$

where $\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \dots, \phi_N(\mathbf{x})]$. We seek to learn a model of the form

$$\dot{\mathbf{z}} = A\mathbf{z} + B\mathbf{u} \tag{18}$$

where matrices $A \in \mathbb{R}^{(N+n)\times(N+n)}, B \in \mathbb{R}^{(N+n)\times m}$ are unknown, and are to be inferred from the collected data.

We focus on systems governed by Lagrangian dynamics, whose state space coordinates consist of position, $\mathbf{p}$, and velocity $\mathbf{v}$: $\mathbf{x} = [\mathbf{p}, \mathbf{v}]^T$, with $\dot{\mathbf{p}} = \mathbf{v}$. The rows of $A$ corresponding to the position states are known. Furthermore, by construction the eigenvalues $\Lambda$ describe the evolution of the eigenfunctions under the nominal control law. Therefore, the rows of $A$ corresponding to eigenfunctions are also known. As a result, the lifted state space model has the following structure:

$$\begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\boldsymbol{\phi}}\left(\begin{bmatrix}\mathbf{p}\\ \mathbf{v}\end{bmatrix}\right) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I & 0 \\ A_{\mathbf{vp}} & A_{\mathbf{vv}} & A_{\mathbf{v}\boldsymbol{\phi}} \\ -B_{\boldsymbol{\phi}}K_{nom} & & \Lambda \end{bmatrix}}_{A} \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \boldsymbol{\phi}\left(\begin{bmatrix}\mathbf{p}\\ \mathbf{v}\end{bmatrix}\right) \end{bmatrix} + \underbrace{\begin{bmatrix} B_{\mathbf{p}} \\ B_{\mathbf{v}} \\ B_{\boldsymbol{\phi}} \end{bmatrix}}_{B} \mathbf{u} \tag{19}$$

where $0, I, \Lambda, K_{nom}$ are fixed matrices and $A_{\mathbf{vp}}, A_{\mathbf{vv}}, A_{\mathbf{v}\boldsymbol{\phi}}$, $B_{\mathbf{p}}, B_{\mathbf{v}}, B_{\boldsymbol{\phi}}$ are determined from data. The term $-B_{\boldsymbol{\phi}}K_{nom}$ accounts for the effect of the nominal controller on the evolution of the eigenfunctions. To infer the different parts of (19), we construct the data matrices and formulate the loss function for three separate ordinary least squares regression problems defined in Equation (13). The data matrices are aggregations of the data snapshots, e.g. $P = [\mathbf{p}_1^1, \dots, \mathbf{p}_{M_s}^1, \dots, \mathbf{p}_1^{M_t}, \dots, \mathbf{p}_{M_s}^{M_t}]^T$. Furthermore, $P, V, \Phi, U, U_{nom}$ are derived from measurements and $\dot{P}, \dot{V}, \dot{\Phi}$ are found by numerically differentiating $P, V, \Phi$, respectively. $U$ and $U_{nom}$ are related by $U = U_{nom} + U_{pert}$, where $U_{pert}$ is the random perturbation added to the control action to induce exploratory behavior as discussed in Section

III. The KEEDMD exploits the control perturbation to learn the effect of actuation on the Koopman eigenfunctions.

To reduce overfitting, different forms of regularization can be added to the objectives of the regression formulations. In particular, LASSO-regularization promoting sparsity in the learned matrices has been shown to perform well for dynamical systems [28] when used in normal EDMD. This has also been the case in our numerical simulation, where LASSO-regularization seem to improve the prediction performance and the stability of the results.

When the lifted state space model is identified, state estimates can be obtained as $\mathbf{x} = C\mathbf{z}$, where $C = [I \quad 0]$. $C$ is denoted the *projection matrix* of the lifted state space model. If the state itself is not included in the lifted state, the $C$-matrix can be found by formulating a least squares problem with loss function $||\mathbf{x} - C\mathbf{z}||_2^2$.

### A. Model Predictive Controller Details

Inspired by [20], we use the Koopman operator to transform the original non-linear optimization problem into an efficient quadratic program (QP) that is solved at each time step. The QP formulation requires us to discretize the previously learned linear continuous dynamics. We assume a known objective function that is solely a function of states and controls. For simplicity, we use a quadratic objective function with respect to the state error and control action, but other objective functions can be used by simply adding them to the lifting functions. We assume known control bounds $u_{\min}, u_{\max} \in \mathbb{R}^m$ and state bounds $x_{\min}, x_{\max} \in \mathbb{R}^n$. All these assumptions define the following optimization problem that we solve at each time step:

$$\begin{array}{lll} \min\limits_{\substack{u\in\mathbb{R}^{m\times N_p} \\ z\in\mathbb{R}^{N\times N_p}}} & \sum_{p=1}^{N_p} \left[ (Cz_p - \tau_p)^T Q \, (Cz_p - \tau_p) + u_p^T R u_p \right] & \\ \text{s.t.} & z_p = A_d z_{p-1} + B_d u_p & \\ & x_{\min} \leq C z_p \leq x_{\max} & p = 1, \dots, N_p \\ & u_{\min} \leq u_p \leq u_{\max} & \\ & z_0 = \phi(x_k) & \end{array} \tag{20}$$

here $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ are positive semidefinite cost matrices, $\tau \in \mathbb{R}^{n\times N_p}$ is the reference trajectory, $A_d \in \mathbb{R}^{N\times N}$ and $B_d \in \mathbb{R}^{N\times m}$ are the discrete time versions of (18), $C \in \mathbb{R}^{n\times N}$ is the projection matrix, and $\phi \in \mathbb{R}^N$ are the eigenfunctions.

To remove the dependency on the lifting dimension $N$ in Eq. (20), the state is eliminated via an explicit relation with the control input. This formulation is referred as the *dense* form MPC. This step greatly reduces the number of optimization variables, which is beneficial as we must solve the MPC problem in real-time. In this form, the MPC controller is agnostic not only of the lifting dimension but of the whole Koopman formalism, *i.e.* the eigenfunctions $\phi$ and linear matrices $A_d$, $B_d$ and $C$ do not directly appear in the formulation. In addition, we relax the state constraints while keeping hard control bounds in order to ensure there is always a solution to the quadratic program. The relaxation penalty can be tuned to have negligible violation of the constraints and to avoid numerical problems.

The solution of (20) is a sequence of control actions $u \in \mathbb{R}^{m\times N_p}$. Following MPC convention, only the first controller command is used, namely $u_k$ at time step $k$, whereupon the optimization problem is resolved.

## V. Experimental Results

To obtain an initial evaluation of the performance of the proposed framework, we study the canonical cart pole system with continuous dynamics[1]:

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{M+m}\left(ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2 + F\right) \\ \frac{1}{l}\left(g\sin\theta + \ddot{x}\cos\theta\right) \end{bmatrix} \tag{21}$$

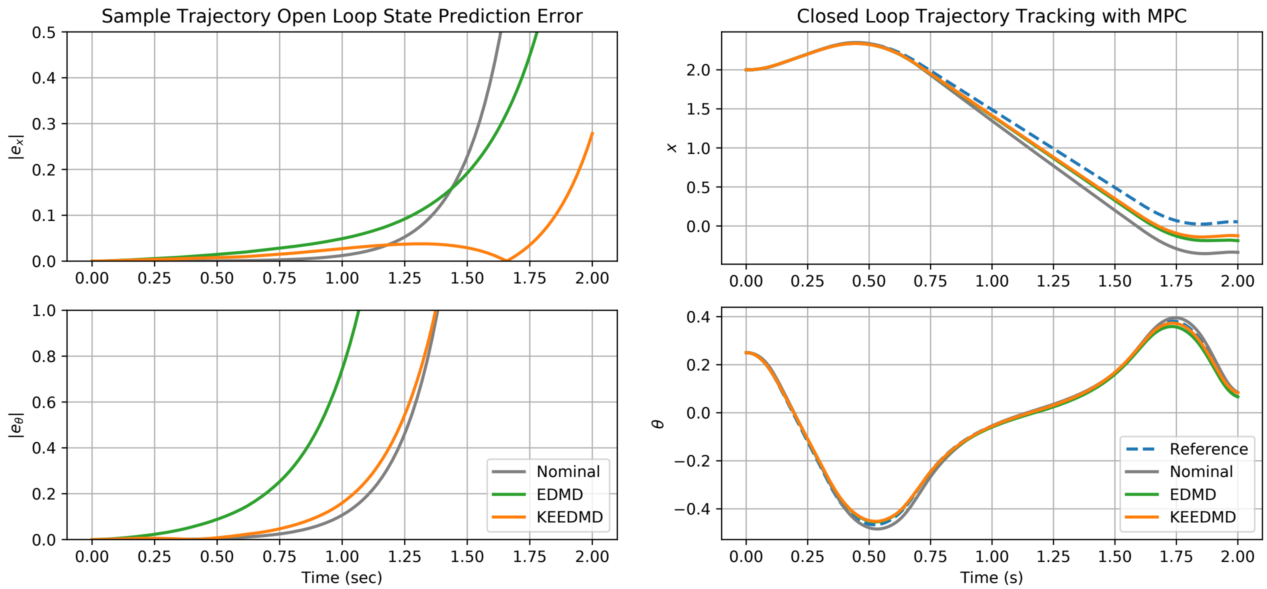
where $x, \theta$ are the cart's horizontal position and the angle between the pole and the vertical axis, respectively, $M, m$ are the cart's and pole tip's mass, respectively, $l$ is the pole length, $g$ the gravitational acceleration, and $F$ the horizontal force input on the cart. The linearization of the dynamics around the origin is used as the nominal model. Starting with knowledge of the nominal model only, our goal is to learn a lifted state space model of the dynamics to improve the system's ability to move to the origin from a initial condition two meters away. This problem is interesting because the cart pole system is underactuated and therefore incapable of reaching the origin (in all states) with PD control. We will collect data with a nominal controller, learn the lifted state space model and use this model to design an improved MPC-controller.

To build the dataset used for training, 40 trajectories are simulated by sampling an initial point in the interval $(x, \theta, \dot{x}, \dot{\theta}) \in [-2.5, 2.5] \times [-0.25, 0.25] \times [-0.05, 0.05] \times [-0.05, 0.05]$, generating a two second long trajectory from the initial point to the origin with a MPC-controller based on the nominal model, and simulating the system with a PD controller stabilizing the system to the trajectory. The PD controller is perturbed with white noise of variance 0.5 to aid the model fitting as described in Section III and state and control action snapshots are sampled from the simulated trajectories at 100 hz. With the collected data, eigenfunctions are constructed as described in Algorithm 1 and a lifted state space model is identified according to (13).

To benchmark our results, we compare our prediction and control results against (1) the nominal model, and (2) a EDMD-model with the state and Gaussian radial basis functions as lifting functions. In both the EDMD and KEEDMD models, a lifting dimension of 85 is used and elastic net regularization is added with regularization parameters determined by cross validation. The diffeomorphism, $h$, is parameterized by a 3-layer neural network with 30 units in each layer and implemented with *PyTorch* [29]. The EDMD and KEEDMD regressions are implemented with *Scikit-learn* [30].

First, we compare the open loop prediction performance by executing a two second long trajectory with the MPC

[1]The code for learning and control is publicly available on `https://github.com/Cafolkes/keedmd`




(a) Open loop prediction

(b) Closed loop trajectory tracking

Fig. 2: Comparison of the nominal model, EDMD, and KEEDMD (a) prediction performance and (b) closed loop performance.

controller based on the nominal model and then using that control sequence as an open loop control sequence in the nominal model, EDMD-model, and the KEEDMD-model. The absolute error between the predicted evolution and the true system evolution over the duration of a sample trajectory is depicted in Figure 2a. The nominal model is able to predict the evolution accurately for approximately the first second. Both the EDMD and KEEDMD models are able to predict the evolution accurately for a longer duration and KEEDMD is both accurately predicting the evolution longer than EDMD and outperforming (lower absolute error) EDMD before EDMD diverges. Although the performance improvement may not seem large, it leads to significant improvements when the model is used in closed loop control.

To evaluate the closed loop performance, we compare the behaviour of the three different models on the task of moving from initial point $(x_0, \theta_0, \dot{x}_0, \dot{\theta}_0) = (2, 0.25, 0, 0)$ in two seconds. The nominal model is used to generate a trajectory from the initial point to the origin. Then, a dense form MPC-controller using the learned lifted state space model is implemented in Python using the QP solver *OSQP* [31]. The trajectory tracking performance is significantly improved when the lifted state space models are used, see Figure 2b. To analyze the differences further, the improvement of total MPC cost with quadratic penalty of tracking error and control effort with fixed penalty matrices $Q, R$ is reported in Table I. Our method significantly outperforms the nominal model and EDMD-based MPC-controllers.

TABLE I: Improvement in MPC cost with learned models

| | **Improvement over nominal model** | **Improvement over EDMD-model** |
|---|---|---|
| EDMD | $-65.30\%$ | |
| KEEDMD | $-74.91\%$ | $-27.70\%$ |

## VI. Conclusions and Future Work

We presented a novel method to learn non-linear dynamics, using Koopman Eigenfunctions constructed from principal eigenfunctions and a non-linear diffeomorphism as lifting functions for Extended Dynamic Mode Decomposition (EDMD). We then used a model predictive controller framework to obtain an optimal controller, while respecting state and control input bounds. We showed in simulation that the method significantly outperforms the linearization around the origin as well as the classical EDMD method with the same number of lifting functions. These preliminary results show focusing on the spectral properties of the Koopman Operator allow for a more compact representation while achieving similar performance. Furthermore, we demonstrate that our methodology can be used to implement a nonlinear MPC controller in a highly computationally efficient manner by exploiting the linear structure and eliminating the dependence on the lifting dimension. In future work, this method will be applied on experimental platforms. One of the main current limitations is the need to collect data using a linear stabilizing controller and current work is investigating this issue [26]. In addition, different control strategies like robust and adaptive control can be investigated.

## Acknowledgement

This work has been supported in part by Raytheon Company and the DARPA Physics of Artificial Intelligence program, HR00111890033. The first author is grateful for the support of the Aker Scholarship Foundation.

# REFERENCES


[1] S. Lupashin, M. Hehn, M. W. Mueller, A. P. Schoellig, M. Sherback, and R. DAndrea, "A platform for aerial robotics research and demonstration: The flying machine arena," *Mechatronics*, vol. 24, no. 1, pp. 41–54, 2014.

[2] H. K. Khalil, "Nonlinear systems," *Upper Saddle River*, 2002.

[3] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[4] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian Process Dynamical Systems," in *Proc. Neural Information Processing Sytems*, 2006.

[5] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural Lander: Stable Drone Landing Control using Learned Dynamics," 2018.

[6] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic Learning with Control Lyapunov Functions for Uncertain Robotic Systems," 2019.

[7] Y. Duan, X. Chen, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," *arXiv*, 2016.

[8] J. García and F. Fernández, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, 2015.

[9] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks," in *Proc. AAAI*, 2019.

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

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

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

[13] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A DataDriven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[14] S. Brunton and N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. No. April, 2019.

[15] K. Taira, S. L. Brunton, S. T. M. Dawson, C. W. Rowley, T. Colonius, B. J. Mckeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal Analysis of Fluid Flows: An Overview," 2017.

[16] E. Berger, M. Sastuba, D. Vogt, B. Jung, and H. Ben Amor, "Estimation of perturbations in robotic behavior using dynamic mode decomposition," *Advanced Robotics*, vol. 29, pp. 331–343, 3 2015.

[17] 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*, vol. 258, pp. 1–15, 2016.

[18] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, 2018.

[19] J. L. Proctor, S. L. Brunton, and J. Nathan Kutz, "Generalizing Koopman Theory to Allow for Inputs and Control *," vol. 17, no. 1, pp. 909–930, 2018.

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

[21] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," no. April, 2017.

[22] M. Korda and I. Mezić, "Learning Koopman eigenfunctions for prediction and control: the transient case," pp. 1–32, 2018.

[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[24] R. Mohr and I. Mezić, "Construction of eigenfunctions for scalar-type operators via Laplace averages with connections to the Koopman operator," pp. 1–25, 2014.

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

[26] C. Folkestad, D. Pastor, and J. Burdick, "Episodic Koopman Learning of Nonlinear Robot Dynamics with Application to Fast Quadrotor Landing," in *Proc. International Conference of Robotics and Autonomy*, Sep. (submitted) 2020.

[27] L. Ljung, *System identificationTheory for the user*. 1987.

[28] S. L. Brunton, J. L. Proctor, and J. Nathan Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems,"

[29] A. Pazske, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, D. A. Lin, Zeming, L. Antiga, and A. Lerer, "Automatic Differentiation in PyTorch," tech. rep., 2017.

[30] F. Pedregosa FABIANPEDREGOSA, V. Michel, O. Grisel OLIVIER-GRISEL, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot andÉdouardand, a. Duchesnay, and F. Duchesnay EDOUARD-DUCHESNAY, "Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An Operator Splitting Solver for Quadratic Programs," *2018 UKACC 12th International Conference on Control, CONTROL 2018*, no. 1, p. 339, 2018.