

Derivative-Based Koopman Operators for Real-Time Control of Robotic Systems

Giorgos Mamakoukas[✉], Maria L. Castaño[✉], Xiaobo Tan[✉], and Todd D. Murphey[✉]

Abstract—This paper presents a generalizable methodology for data-driven identification of nonlinear dynamics that bounds the model error in terms of the prediction horizon and the magnitude of the derivatives of the system states. Using higher-order derivatives of general nonlinear dynamics that need not be known, we construct a Koopman operator-based linear representation and utilize Taylor series accuracy analysis to derive an error bound. The resulting error formula is used to choose the order of derivatives in the basis functions and obtain a data-driven Koopman model using a closed-form expression that can be computed in real time. Using the inverted pendulum system, we illustrate the robustness of the error bounds given noisy measurements of unknown dynamics, where the derivatives are estimated numerically. When combined with control, the Koopman representation of the nonlinear system has marginally better performance than competing nonlinear modeling methods, such as SINDy and NARX. In addition, as a linear model, the Koopman approach lends itself readily to efficient control design tools, such as LQR, whereas the other modeling approaches require nonlinear control methods. The efficacy of the approach is further demonstrated with simulation and experimental results on the control of a tail-actuated robotic fish. Experimental results show that the proposed data-driven control approach outperforms a tuned PID (Proportional Integral Derivative) controller and that updating the data-driven model online significantly improves performance in the presence of unmodeled fluid disturbance. This paper is complemented with a video: https://youtu.be/9_wx0tdDta0.

Index Terms—Koopman operator, model learning, robotic fish, data-driven control

I. INTRODUCTION

DYNAMICS of robotic systems are often unknown, highly nonlinear, and high-dimensional, making real-time control challenging [1]. Underwater applications represent many of these unmet challenges. In particular, underwater robots are often underactuated (typically by design to reduce weight and cost) and highly nonlinear, and the fluid environments

they operate in are difficult to model and time-varying. These challenges call for feedback policies that can learn or adapt online to unmodeled changes [2] and balance model accuracy and computational efficiency.

One can draw from many control schemes, including linear quadratic regulator (LQR) [3], linear model predictive control (LMPC) [4], nonlinear model predictive control (NMPC) [5], feedback linearization [6], differential dynamic programming (DDP) [7], sequential action control (SAC) [8] and variants of the above [9]–[11]. In fact, several of these methods have already been explored in underwater tasks using robotic fish of different morphologies. Researchers have performed maneuvering, speed and orientation control, collision-avoidance, point-to-point navigation as well as velocity and position tracking using a myriad of control schemes, such as PID [12], [13], LQR [14], SAC [11], [15], fuzzy control [16], [17], geometric control [18], [19], sliding mode control [20], NMPC [21], feedback linearization [22], backstepping control [23] or even a combination of the above [24], [25]. However, the aforementioned methods are either system-specific, apply to dynamics with certain structures, or are computationally prohibitive for real-time identification and control of resource-constrained robots. They also typically require full knowledge of the dynamics.

The Koopman operator has recently drawn attention in the robotics community, as it can help address both the difficulty with nonlinearity and the need to incorporate data in the model [26]–[28]. Specifically, the Koopman operator propagates a nonlinear system in a linear manner without loss of accuracy by evolving functions of the states [29]. The linear representation allows one to control the nonlinear system using tools from linear optimal control [30], [31], which is often easier and faster to implement than nonlinear methods, thus enabling online feedback for high-dimensional nonlinear systems. Beyond the computational speed and the reduction in feedback complexity, the linear representation-based control could lead to better performance compared to a controller that is based on the original nonlinear system [32]. The Koopman operator can also be readily combined with machine learning tools to help learn unknown dynamics from data [33]–[43].

A downside of the Koopman operator, however, is that, unless a finite-dimensional invariant subspace exists [32], it is infinite-dimensional. For this reason, recent studies try to obtain a finite-dimensional approximation to the Koopman operator that still captures the dynamics with high fidelity [27], [31]. Koopman-based optimal control applications have suc-

Manuscript received October 5, 2020; revised February 15, 2021. This article was recommended for publication by the Editor Paolo Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation (IIS-1717951, IIS-1715714, DGE1424871). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

G. Mamakoukas and Todd D. Murphey are with the Department of Mechanical Engineering, Northwestern University, Evanston, Illinois 60208, USA (e-mail: giorgosmamakoukas@u.northwestern.edu).

Maria L. Castaño and Xiaobo Tan are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, Michigan 48824, USA.

This article has supplementary material and code at <https://github.com/giorgosmamakoukas/dataDrivenControlOfRoboticFish>.

successfully implemented such finite-dimensional approximations of the operator for various systems with unknown dynamics [40], [41]. In the trade-off between the dimensionality and the modeling accuracy of the linear representation, these studies face the challenge of finding the minimum number and choice of basis functions for the desired accuracy [39].

There has not been a systematic way to address general nonlinear systems; rather, most efforts rely on trial-and-error [40]–[42], [44]–[46] and machine learning tools [39], [47], or are system-specific [26]. Furthermore, there is no method available to bound the modeling error of the finite-dimensional Koopman operators for general nonlinear systems. The only relevant study in [48] analyzes the error bounds of Dynamic Mode Decomposition, closely related to the Koopman operator, for a limited class of systems (parabolic partial differential equations) and with restrictive assumptions on the stability of the identified dynamics.

In this work, we introduce a way of choosing the basis functions and analyze the model accuracy with error bounds. Specifically, we construct the basis functions for the Koopman operator using higher-order derivatives of the nonlinear dynamics, which need not be known; only the derivatives of the tracked states must be available. The error bounds, which depend on the prediction time horizon and the magnitude of the derivatives, can be used to determine the basis functions for the desired level of model accuracy. To our best knowledge, this is the first work that selects basis functions using a systematic methodology and provides an error bound on the accuracy of a Koopman representation for general nonlinear dynamics. To adapt to uncertain or changing dynamics, we obtain the operator using a data-driven, least-squares technique that has a closed-form solution. The linear representation is conducive to various controller designs; in this work, we demonstrate the utility of our online data-driven modeling approach in real-time control using LQR, the gains of which can be obtained with negligible computational cost.

We validate our approach with simulation and experimental results on the control of a tail-actuated robotic fish and compare it to a PID scheme. Although the tuned PID controller is successful at tracking the desired trajectories, it is outperformed in all tasks by the proposed Koopman-LQR controller. Furthermore, updating the dynamical model in real time significantly improves the performance of Koopman-LQR in the presence of unknown fluid disturbance.

This paper extends the work in [49] in significant ways. The additional contributions include analysis of error bounds on the linear approximation of nonlinear systems, comparison to state-of-the-art modeling methods, realization of online model updates and controller synthesis (thus enabling real-time adaptation), quantitative experimental evaluation, including trials in the presence of unknown fluid disturbance, and extensive comparison of the proposed approach with a PID controller.

The organization of the paper is as follows. Section II reviews the Koopman operator and methods to obtain a data-driven finite-dimensional approximation. Section III describes the proposed synthesis of Koopman basis functions and derives

the associated error bounds. Section IV evaluates the proposed data-driven modeling scheme on the control of a tail-actuated robotic fish, with and without the presence of flow disturbance. Section V summarizes the findings of this paper and discusses ideas for further expanding this work.

II. BACKGROUND

A. Koopman Operator

The Koopman operator $\mathcal{K}$ is an infinite-dimensional linear operator that evolves functions of the state $s \in \mathbb{R}^N$ (i.e., $\Psi(s)$, commonly referred to as observables) of a dynamical system. Given general nonlinear dynamics of the form

$$s_{k+1} = F(s_k), \quad (1)$$

where F is the flow map, the Koopman operator advances the observables with the flow of the dynamics:

$$\mathcal{K}\Psi = \Psi \circ F. \quad (2)$$

Thus, it advances measurements of the states linearly. That is,

$$\frac{d}{dt}\Psi(s) = \mathcal{K}\Psi(s) \quad \text{and} \quad \Psi(s_{k+1}) = \mathcal{K}_d\Psi(s_k), \quad (3)$$

where $\mathcal{K}$ and $\mathcal{K}_d$ are the continuous-time and discrete-time operators, respectively, related by $\mathcal{K} = \log(\mathcal{K}_d)/\Delta t$ [50]. In other words, it allows one to evolve the nonlinear dynamics in a linear setting without loss of accuracy. Contrary to linearizing the dynamics around a fixed point, which leads to inaccurate models away from the linearization point, the Koopman operator evolves a nonlinear system with full fidelity throughout the state space. For a more comprehensive review of the Koopman operator, we refer the reader to [34].

Expressing nonlinear systems in a linear manner is a desirable property for many reasons, such as investigating the global stability of a system [37], or extending the local linearization around a point to the whole basin of attraction [51]. In addition to studying the behavior of complex systems, the Koopman framework enables the use of linear optimal control for original nonlinear dynamics. Unfortunately, the infinite-dimensional nature of the Koopman operator makes practical use prohibitive.

B. Koopman Invariant Subspaces

There exist nonlinear systems that admit a finite-dimensional linear Koopman representation. Work in [32] analytically derives such Koopman invariant subspaces for nonlinear systems with a specific polynomial structure, whereas the authors in [52]–[56] identify such spaces from data. In these studies, the authors demonstrate that the LQR control based on the linear representation can outperform LQR control calculated based on the original, nonlinear dynamics. Unfortunately, Koopman invariant subspaces have only been found for a few systems, mentioned above. In fact, there can be no finite-dimensional invariant subspace that includes the states for systems with multiple fixed points [32].

In the absence of a finite-dimensional Koopman invariant subspace, a linear propagation of states will induce errors. Regardless, the benefits of a linear model motivate obtaining

an approximation to the Koopman operator that will evolve the nonlinear system with acceptable accuracy. Recent studies use data-driven regression schemes to approximate the infinite-dimensional operator $\mathcal{K}$ with a finite-dimensional representation $\tilde{\mathcal{K}}$ [27], [31], [41]. In this paper, we adopt the least-squares method shown in [27], which we detail next. Note that the regression method assumes basis functions that are already known, yet our main contribution is in systematically defining those observables in the first place, which we present in Section III.

C. Data-driven Finite-dimensional Approximation to Koopman Operators

To obtain an approximation to the Koopman operator, $\tilde{\mathcal{K}} \in \mathbb{R}^{w \times w}$, one can choose a set of observable functions $\Psi(s) = [\psi_1(s), \psi_2(s), \dots, \psi_w(s)] : \mathbb{R}^N \mapsto \mathbb{R}^w$ (which can include the states s themselves) and use data to solve a least-squares minimization problem. To allow for the effect of actuation, (3) is modified such that the observables include control terms u as well [31], [40]. For the discrete-time case, this minimization takes the form

$$\tilde{\mathcal{K}}_d^* = \underset{\tilde{\mathcal{K}}_d}{\operatorname{argmin}} \sum_{k=0}^{P-1} \frac{1}{2} \|\Psi(s_{k+1}, u_{k+1}) - \tilde{\mathcal{K}}_d \Psi(s_k, u_k)\|^2, \quad (4)$$

where P is the number of measurements. Each measurement is a set of an initial state s_k , final state s_{k+1} , and the actuation applied at the same instants, u_k and u_{k+1} , respectively. The above expression has a closed-form solution, given by

$$\tilde{\mathcal{K}}_d^* = \mathcal{A} \mathcal{G}^\dagger, \quad (5)$$

where

$$\begin{aligned} \mathcal{A} &= \frac{1}{P} \sum_{k=0}^{P-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T \\ \mathcal{G} &= \frac{1}{P} \sum_{k=0}^{P-1} \Psi(s_k, u_k) \Psi(s_k, u_k)^T \end{aligned} \quad (6)$$

and $\dagger$ is the Moore-Penrose pseudoinverse. Note that the time spacing Δt between measurements s_k and s_{k+1} must be consistent for all P training measurements.

The data-driven approximation of the Koopman operator is not inherently different from other system identification techniques. In fact, the Koopman operator can be approximated using any of the standard regression methods, such as ridge or lasso regression [57], [58]. More importantly, contrary to standard system identification tools that may try to estimate unknown parameters or, more generally, the nonlinear dynamics of a system [38], [59], the Koopman operator framework places the system identification task in the context of seeking linear transformations of the states, which is useful for control [60]–[62] and other purposes, as discussed in Section I.

D. LQR on Koopman Operator

Consider a linear system with states $s \in \mathbb{R}^N$, control $u \in \mathbb{R}^M$, and a performance objective

$$J = \sum_{k=0}^{\infty} (s_k - s_{des,k})^T Q (s_k - s_{des,k}) + u_k^T R u_k, \quad (7)$$

where $Q \succeq 0 \in \mathbb{R}^{N \times N}$ and $R \succ 0 \in \mathbb{R}^{M \times M}$ are weights on the deviation from the desired states s_{des} and the applied control, respectively. Further, consider the Koopman representation

$$\Psi(s_{k+1}, u_{k+1}) = \tilde{\mathcal{K}}_d \Psi(s_k, u_k). \quad (8)$$

For simplicity, we choose $\Psi(s, u) = [\Psi_s^T(s), \Psi_u^T(u)]^T$, where $\Psi_s(s) \in \mathbb{R}^{w_s}$ are the functions that depend on the states s and $\Psi_u(u) \in \mathbb{R}^{w_u}$ are the functions that depend on the input u , where $w = w_s + w_u$. Using this notation, we rewrite (8) as

$$\begin{bmatrix} \Psi_s(s_{k+1}) \\ \Psi_u(u_{k+1}) \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \Psi_s(s_k) \\ \Psi_u(u_k) \end{bmatrix}, \quad (9)$$

where $A \in \mathbb{R}^{w_s \times w_s}$ and $B \in \mathbb{R}^{w_s \times w_u}$ are submatrices of $\tilde{\mathcal{K}}_d$ that describe the dynamics of the state-dependent functions and change only when $\tilde{\mathcal{K}}_d$ is updated. We use $\Psi_u(u_k) = u_k$ to ensure that control appears linearly in the model such that

$$\Psi_s(s_{k+1}) = A \Psi_s(s_k) + B u_k. \quad (10)$$

Given the Koopman dynamics (9), we choose the performance objective

$$J_{\tilde{\mathcal{K}}} = \sum_{k=0}^{\infty} (\Psi_s(s_k) - \Psi_s(s_{des,k}))^T Q_{\tilde{\mathcal{K}}} (\Psi_s(s_k) - \Psi_s(s_{des,k})) + u_k^T R u_k, \quad (11)$$

where $Q_{\tilde{\mathcal{K}}} \succeq 0 \in \mathbb{R}^{w_s \times w_s}$ penalizes the deviation from the desired observable functions $\Psi_s(s_{des})$. We let the first N observables be the original states s and set

$$Q_{\tilde{\mathcal{K}}} = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}, \quad (12)$$

so that a meaningful comparison can be made with regards to the original nonlinear system and the associated objective function shown in (7). The Koopman representation is conducive to linear quadratic regulator (LQR) feedback of the form

$$u_k = -K_{LQR} (\Psi_s(s_k) - \Psi_s(s_{des,k})), \quad (13)$$

where $K_{LQR} \in \mathbb{R}^{M \times w_s}$, the LQR gains, can be readily calculated from A, B in (9) [63], [64]. Note that, for given LQR gains, the control is updated using only the functions $\Psi_s(s)$, leading to minimal computation. For more details on the control policy used for the Koopman representation, the reader can refer to [49]. Last, we want to emphasize that our approach for synthesizing data-driven Koopman representations can be used with different feedback schemes, such as MPC control.

III. SYNTHESIS OF BASIS FUNCTIONS FOR ERROR-BOUNDED KOOPMAN REPRESENTATION

This section motivates using higher-order derivatives of nonlinear dynamics to populate the observables of an approximate Koopman operator. The benefits of a derivative-based representation are twofold. First, subject to a finite number of basis functions, it allows one to best capture, locally in time, nonlinear dynamics. For systems that admit a finite-dimensional Koopman invariant subspace, it is straightforward to show that the terms in the observable functions $\Psi(s)$ capture all higher-order derivatives of the original states. This is the

reason why the linear representation matches the nonlinear dynamics with no error. This is also true for the invariant subspaces found in [32], where the Koopman observable functions are populated using the Carleman linearization approach [65]–[67]; for the polynomial systems considered there, the observable functions correspond to the higher-order derivatives of the nonlinear dynamics. When the derivative functions do not span an invariant subspace, populating the observables $\Psi(s_k)$ with higher-order derivatives instead of arbitrary basis functions generates, locally in time, an increasingly (with the order of derivatives) accurate linear representation of the nonlinear dynamics.

Second, the derivative-based representation enables the derivation of error bounds on future predictions. Notably, when the model is entirely data-driven, these error bounds might not be enforced, but offer sound bound estimates that still hold, as we illustrate in Section III-D, but which are then dependent on the quality of data used in the data-driven process. To approximate the Koopman operator, so far studies have largely focused on data-driven methods of the form in (4) that consider only the local error across one time step, that is

$$\Psi(s_{k+1}, u_{k+1}) - \tilde{K}_d \Psi(s_k, u_k), \quad (14)$$

as used in (4). Another measure of the accuracy of the Koopman representation is the global error, over an arbitrarily long time window across all time steps $m > 0$ (see Fig. 1), that is

$$\Psi(s_{k+m}, u_{k+m}) - \tilde{K}_d^m \Psi(s_k, u_k). \quad (15)$$

The derivative-based linear embedding methodology presented in this work enables the computation of global error bounds of the Koopman representation. By exploiting the accuracy properties of Taylor expansions, we can synthesize Koopman basis functions that bound the model error for any particular order of linear representation. The error bounds in turn allow one to select the lowest-order representation that meets a desired accuracy. This analysis is presented next.

The proposed linear embedding method does not require knowledge of the dynamics; instead, it requires only that the time derivatives of the system states of interest be available. The values of the derivatives can be either evaluated, using knowledge of the dynamics equations, or numerically estimated from state measurements (using finite differencing or other methods [68]). The method can be used

- for known nonlinear dynamics: each state derivative is analytically derived from the dynamics equation and constitutes a basis function for the Koopman operator (one basis function per derivative)
- for dynamics whose structure is known but coefficients might be unknown or changing, as we illustrate in Section III-C: derivatives are analytically derived from each term that appears in the dynamics equation; each term that is computed constitutes a separate basis function for the Koopman operator (at least one basis function per derivative)
- for completely unknown dynamics, as we illustrate in III-D: each state derivative is numerically calculated and

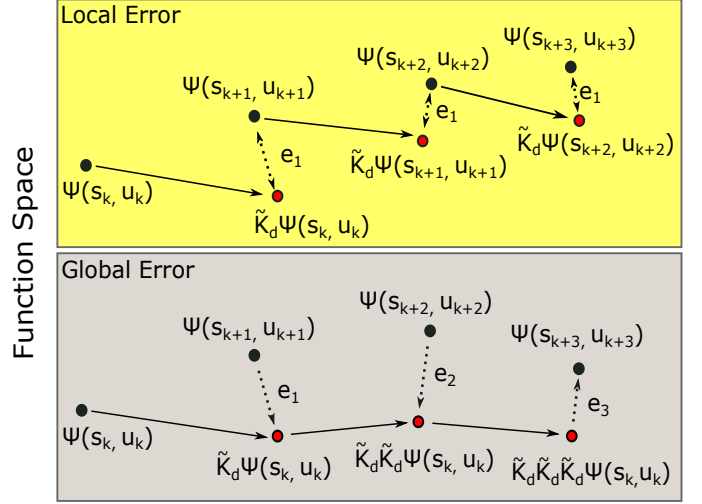


Fig. 1: Local and global errors induced by approximate Koopman operators. The local error is the error induced by the operator across one step, assuming no error in the initial conditions. The global error is the total deviation away from the true states across multiple steps.

constitutes a basis function for the Koopman operator (one basis function per derivative).

A. Error Bounds of Derivative-Based Koopman Operators

The evolution of a nonlinear function $f(t)$ that is continuously differentiable up to n th order can be approximated with a Taylor series as

$$\begin{aligned} \tilde{f}(t_{k+1}) = & f(t_k) + f'(t_k) \cdot (t_{k+1} - t_k) \\ & + \frac{f''(t_k)}{2!} \cdot (t_{k+1} - t_k)^2 + \dots + \frac{f^{(n)}(t_k)}{n!} (t_{k+1} - t_k)^n, \end{aligned} \quad (16)$$

where tilde ($\tilde{\cdot}$) denotes the predicted value of a function, and not its true value.¹ To keep the algebraic expressions compact, let $t_{k+1} - t_k = \Delta t$ and $f_k^{(i)} \triangleq f^{(i)}(t_k)$, $\forall i \in \mathbb{Z} \cap [1, n]$, which simplifies the above expression to

$$\tilde{f}_{k+1} = f_k + f'_k \cdot \Delta t + f''_k \cdot \frac{\Delta t^2}{2!} + \dots + f_k^{(n)} \frac{\Delta t^n}{n!}. \quad (17)$$

Propagating a function using its derivatives allows one to use the accuracy of the Taylor series to characterize the error in the evolution of a function across one time step Δt . The local error induced by a Taylor series approximation using up to n derivatives across one time step is $R_n(k) = f_{k+1} - \tilde{f}_{k+1}$. This error is calculated using Lagrange's remainder formula:

$$R_n(k) = \frac{f_c^{(n+1)}}{(n+1)!} \Delta t^{n+1}, \quad (18)$$

where $f_c^{(n+1)} \triangleq f^{(n+1)}(c)$ is the time derivative of order $n+1$ evaluated at some time $c \in [t_k, t_{k+1}]$. If there exists a positive

¹We assume that the true values of the function f and its derivatives are known at time t_k . Uncertainty about the initial values can be readily included in the formula of the global error, later shown in this paper.

real number L such that $|f_c^{(n+1)}| \leq L$ for all $c \in [t_k, t_{k+1}]$, then the upper error bound in (18) becomes

$$|R_n(k)| \leq \frac{L}{(n+1)!} \Delta t^{n+1}. \quad (19)$$

For the purpose of applying linear control synthesis tools to linear representations of nonlinear dynamics, we bring the Taylor approximation in (17) to a linear matrix form:

$$\underbrace{\begin{pmatrix} \tilde{f}_{k+1} \\ \tilde{f}'_{k+1} \\ \tilde{f}''_{k+1} \\ \vdots \\ \tilde{f}^{(n)}_{k+1} \end{pmatrix}}_{\Psi(s_{k+1})} \approx \underbrace{\begin{pmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} & \cdots & \frac{\Delta t^n}{n!} \\ 0 & 1 & \Delta t & \cdots & \frac{\Delta t^{n-1}}{(n-1)!} \\ 0 & 0 & 1 & \cdots & \frac{\Delta t^{n-2}}{(n-2)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{\tilde{K}_d} \underbrace{\begin{pmatrix} f_k \\ f'_k \\ f''_k \\ \vdots \\ f^{(n)}_k \end{pmatrix}}_{\Psi(s_k)}. \quad (20)$$

For a fixed Δt , expression (20) resembles (3), where the derivatives of the function f_k are the observables $\Psi(s_k)$. When representing the Taylor series expansion, the derivative functions $f^{(i)}$ are known at time step t_k and approximated at time t_{k+1} by $\tilde{f}^{(i)}$. When training a Koopman operator, pairs of measurements of the states s_k and s_{k+1} are used to evaluate the basis functions at the corresponding time steps: $\Psi(s_k)$ and $\Psi(s_{k+1})$. Note that, in (20), all derivatives of f_k are assumed to be different functions. The analytical expression (20) is equivalent to a Taylor series expansion (17) across one time step Δt for all the basis functions. Therefore, the same error analysis (19) applies to each observable in (20).

When propagating a function across multiple time-steps using (20), the observable functions are themselves numerically propagated instead of being evaluated with measurable states at each time step, as is the case for a typical integration scheme. As a result, error accumulates not only in approximation of the original function f , but in the other observables as well. To track the error in the original f , therefore, it is necessary to be able to model the error in all of the basis functions. Using the accuracy of the Taylor series structure, we are able to model the error on every basis function and ultimately bound the model error in f .

Theorem 1. Consider a general nonlinear function $f(t)$ that is continuously differentiable up to order n . Propagating $f(t)$ and its first n derivatives using the Taylor-based linear representation (20) induces an error in $f(t)$ that is given by

$$e_k = \sum_{i=1}^{k-1} \sum_{j=1}^n e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!}, \quad (21)$$

where $n \in \mathbb{Z}^{\geq 0}$ is the number of derivative basis functions used, $k \in \mathbb{Z}^{\geq 1}$ is the number of time steps into the future, and, from Lagrange's remainder formula (18), $f_{i,i+1}^{(n+1)}$ is the

$n+1$ th time derivative of function f evaluated at some time $t \in [t_i, t_{i+1}]$. The error bound is given by

$$|e_k| \leq \frac{T^{n+1}}{(n+1)!} |f_{\max}^{(n+1)}|, \quad (22)$$

where $T \triangleq k\Delta t$ is the prediction time horizon and $|f_{\max}^{(n+1)}|$ is the maximum magnitude of the $n+1$ th derivative.

Proof. For the derivation of the error expression (21), see Appendix A. For the derivation of the error bound formula (22), see Appendix B. $\square$

To the best of our knowledge, this is the first work that provides prediction error bounds on the accuracy of a Koopman representation for general nonlinear dynamics. Prediction error bounds based on Taylor series had previously only been derived for a single step, and not for an arbitrary number of time steps into the future, as we derive in this paper.

The error bound (22) is associated with the Koopman representation (20) for the dynamics of a single function f . The same methodology can be used to propagate multiple states of a system with coupled dynamics. Specifically, a system with states $s(t)$ and general nonlinear dynamics $\dot{s}(t) = g(s(t)) \in \mathbb{R}^N$ that are continuously differentiable up to order n

$$\frac{d}{dt} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} g_1(s) \\ g_2(s) \\ \vdots \\ g_N(s) \end{bmatrix} \quad (23)$$

can be propagated in discrete time as

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ \vdots \\ s_{N,k+1} \end{bmatrix} = \begin{bmatrix} s_{1,k} + g_{1,k} \cdot \Delta t + \cdots + g_{1,k}^{(n_1)} \frac{\Delta t^{n_1+1}}{(n_1+1)!} \\ s_{2,k} + g_{2,k} \cdot \Delta t + \cdots + g_{2,k}^{(n_2)} \frac{\Delta t^{n_2+1}}{(n_2+1)!} \\ \vdots \\ s_{N,k} + g_{N,k} \cdot \Delta t + \cdots + g_{N,k}^{(n_N)} \frac{\Delta t^{n_N+1}}{(n_N+1)!} \end{bmatrix}, \quad (24)$$

where n_j for $j \in \mathbb{Z} \cap [1, N]$ indicates the highest-order of derivatives of g_j used to propagate the j th state of the original dynamics (which does not have to be the same for all states). The above expression can be rewritten in a linear form similar

to (20)

$$\underbrace{\begin{bmatrix} s_{1,k+1} \\ g_{1,k+1} \\ \vdots \\ g_{1,k+1}^{(n_1)} \\ \hline s_{2,k+1} \\ g_{2,k+1} \\ \vdots \\ g_{2,k+1}^{(n_2)} \\ \hline \vdots \\ \hline s_{N,k+1} \\ g_{N,k+1} \\ \vdots \\ g_{N,k+1}^{(n_N)} \end{bmatrix}}_{\Psi(s_{k+1})} = \underbrace{\begin{bmatrix} T(n_1) & 0 & \cdots & 0 \\ \hline 0 & T(n_2) & \cdots & 0 \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & T(n_N) \end{bmatrix}}_{\tilde{\mathcal{K}}_d} \underbrace{\begin{bmatrix} s_{1,k} \\ g_{1,k} \\ \vdots \\ g_{1,k}^{(n_1)} \\ \hline s_{2,k} \\ g_{2,k} \\ \vdots \\ g_{2,k}^{(n_2)} \\ \hline \vdots \\ \hline s_{N,k} \\ g_{N,k} \\ \vdots \\ g_{N,k}^{(n_N)} \end{bmatrix}}_{\Psi(s_k)}, \quad (25)$$

where

$$T(n_j) = \begin{bmatrix} 1 & \Delta t & \cdots & \frac{\Delta t^{n_j+1}}{(n_j+1)!} \\ 0 & 1 & \cdots & \frac{\Delta t^{n_j}}{n_j!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{for } j \in \mathbb{Z} \cap [1, N]. \quad (26)$$

Note how (25) is grouped into submatrices that propagate independently each state and its higher-order derivatives. The basis functions are the states and their derivatives. Propagating a nonlinear system with states s and nonlinear dynamics $g(s(t))$ using (25) is equivalent to propagating each state separately using (20) and thus induces, for each state, an error given by an expression similar to (21).

The formulation in (25) uses basis functions that depend, for simplicity, only on the state s . When working with a system that has control inputs, one can treat controls u in a similar fashion and calculate its higher-order derivatives, by introducing u as dummy states that are the derivatives of the control input, a common practice [52].

Corollary 1. Consider general nonlinear dynamics $\dot{s}(t) = g(s(t))$ that are continuously differentiable up to order n . Propagating $s(t)$ using (25) induces a bounded error on state s_i , $i \in \mathbb{Z} \cap [1, N]$, given by (21), where $g(s(t))^{(i-1)} = f(t)^{(i)}$.

Proof. Consider the propagation of each state s_i and its derivative functions $g_i^{(\cdot)}$. Let each s_i be a function f_i whose n_i^{th} derivative is $f_i^{(n_i)}$. Then, from Theorem 1, propagating each state $s_i \triangleq f_i$ with (20) induces an error given by (21). $\square$

The error bound (22) allows one to calculate the maximum possible error in each system state when propagating it with the fixed linear matrix (25). As a result, the bound can be used to determine the desired number of derivatives that are needed

for each state that would generate minimal error given a fixed prediction horizon T and subject to the nonlinear dynamics. Alternatively, the error bound can also be used to compute the maximum length of the prediction time horizon for which the state error is bound to remain under a threshold given a set number of derivative basis functions.

Because, in general, there is no closure of the higher-order derivatives and the series has to be truncated, the analytical expression (20) would only lead to an approximate Koopman operator, as is commented in [32]. Note, for example, that the highest derivatives in (25) are not updated at all. For this reason, we use data-driven techniques to obtain a $\tilde{\mathcal{K}}_d$ that more accurately advances all of the basis functions than the analytical expression. On the other hand, the error bound (22) applies to a linear propagation of nonlinear dynamics using (25) and therefore is no longer guaranteed when a data-driven operator is used instead. Nevertheless, it can still serve to measure how amenable nonlinear dynamics are to a linear representation by revealing the relationship between the magnitude and order of the derivatives. Furthermore, empirically, the data-driven model does resemble the Taylor-series structure (25) such that the error bounds remain relevant.² Using simulation results, we next verify the similarity of the data-driven operator to the analytical expression in (20), as well as the validity of the error bounds.

B. Error Bound Estimation Using Data-Driven Operator

The error bound formula (22), derived for a linear representation of (25), remains relevant to a data-driven operator, when the latter has similar structure, i.e., small Frobenius distance, to the Taylor-series form (20). On the other hand, since an operator computed from data may not take exactly the form of (25), the bounds shown in (22) are not strictly enforced, but offer what we refer to as sound bound estimates in the remainder of the paper. To calculate the bound estimates, one needs to compute $|f_{\max}^{(n+1)}|$, the magnitude of the lowest-order derivative of the system states that is not used in the basis functions. When dynamics are known, this value can be calculated numerically. Alternatively, as we show next, one can exploit the Taylor-series structure of the data-driven operator to estimate the error bounds beyond the training set that has been used to generate the Koopman operator even when there is no knowledge of the dynamics.

Specifically, given a linear representation that approximates the Taylor-series structure (20), the local error across one time step induced by the data-driven model can be described by the Taylor series accuracy. Thus, using (22), the error across one time step ($k = 1$) of a function f can be written as

$$|e_1| \leq |f_{\max}^{(n+1)}| \frac{\Delta t^{(n+1)}}{(n+1)!} = |f_{\max}^{(n+1)}| \frac{\Delta t^{(n+1)}}{(n+1)!}, \quad (27)$$

²Although the data-driven solution is not guaranteed to bound all local errors within the Taylor series accuracy, given a training dataset that is a representative part of the state space, solutions that largely deviate from the Taylor-series structure in (20) would generate large local errors in parts of the state space and thus be avoided by the least-squares solution (5).

where e_1 is available from the data-driven training process (4). Let $|e_1|_{\max}$ be the maximum local error, i.e., $|e_1|_{\max} \geq |e_1|$. Then, when the training data set is large enough, one can get

$$|e_1|_{\max} \approx |f_{\max}^{(n+1)}| \frac{\Delta t^{(n+1)}}{(n+1)!}, \quad (28)$$

which is rearranged to

$$|f_{\max}^{(n+1)}| \approx |e_1|_{\max} \frac{(n+1)!}{\Delta t^{(n+1)}}. \quad (29)$$

In short, we use the maximum error across one time step from the training process to estimate the term $|f_{\max}^{(n+1)}|$, which in turn, using (22), allows us to estimate $|e_k|$, the error bound after k time steps. Alternatively, when no analytical model of the dynamics is available, the value $|f_{\max}^{(n+1)}|$ can also be estimated numerically using measurements of f .

C. Synthesis of Derivative-Based Koopman Observables with Structural Knowledge of Dynamics

The derivative-based approach proposed in this work populates Koopman observables with the system states s and their derivative functions. Each derivative is a separate function that can be computed from the analytical expression when dynamics are fully known, or numerically estimated from measurements when no model exists. In this subsection, we show how we construct the basis functions to exploit structural knowledge of dynamics that have unknown coefficients.

For simplicity, we assume that the dynamics of each system state depend on a single term, i.e., a nonlinear function multiplied by a coefficient; the case of having multiple such terms can be handled similarly. In particular, consider a nonlinear system with states $s \in \mathbb{R}^N$ and dynamics

$$\frac{d}{dt} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} c_1 g_1(s) \\ c_2 g_2(s) \\ \vdots \\ c_N g_N(s) \end{bmatrix}, \quad (30)$$

where c_i , $i \in \mathbb{Z} \cap [1, N]$, are unknown coefficients and $g_i(s)$ are nonlinear functions of the states s . The second-order time derivatives of the states s are

$$\frac{d^2}{dt^2} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix} = \begin{bmatrix} c_1 g_1'(s) \\ c_2 g_2'(s) \\ \vdots \\ c_N g_N'(s) \end{bmatrix}, \quad (31)$$

where $g_i'(s)$ denotes the time derivative of g_i , and thus

$$c_i g_i'(s) = c_i \left(\frac{\partial g_i}{\partial s_1} c_1 g_1 + \cdots + \frac{\partial g_i}{\partial s_N} c_N g_N \right) \text{ for } i \in \mathbb{Z} \cap [1, N]. \quad (32)$$

For ease of discussion, we limit the analysis to the first two time derivatives, but the same process can continue to generate higher-order derivatives and, thus, additional basis functions.

Using the states s_i , the first-order derivatives g_i and the individual terms $\frac{\partial g_i}{\partial s_j} g_j$ that appear in $g_i'(s)$, where $i, j \in \mathbb{Z} \cap$

$[1, N]$, we populate the basis functions of the Koopman matrix. In discrete time, the states are then propagated with

$$\begin{bmatrix} s_{1,k+1} \\ s_{2,k+1} \\ \vdots \\ s_{N,k+1} \end{bmatrix} = \begin{bmatrix} s_{1,k} + c_1 g_{1,k} \cdot \Delta t + c_1 g_{1,k}' \frac{\Delta t^2}{2!} \\ s_{2,k} + c_2 g_{2,k} \cdot \Delta t + c_2 g_{2,k}' \frac{\Delta t^2}{2!} \\ \vdots \\ s_{N,k} + c_N g_{N,k} \cdot \Delta t + c_N g_{N,k}' \frac{\Delta t^2}{2!} \end{bmatrix}. \quad (33)$$

Substituting for the $g_i'(s)$ terms, we show the expected form for a single state s_1 :

$$\underbrace{\begin{bmatrix} s_{1,k+1} \\ g_{1,k+1} \\ \left\{ \frac{\partial g_1}{\partial s_1} g_1 \right\}_{k+1} \\ \left\{ \frac{\partial g_1}{\partial s_2} g_2 \right\}_{k+1} \\ \vdots \\ \left\{ \frac{\partial g_1}{\partial s_N} g_N \right\}_{k+1} \end{bmatrix}}_{\Psi(s_{k+1})} = \underbrace{\begin{bmatrix} 1 & c_1 \Delta t & c_1^2 \frac{\Delta t^2}{2} & c_1 c_2 \frac{\Delta t^2}{2} & \cdots & c_1 c_N \frac{\Delta t^2}{2} \\ 0 & 1 & c_1 \Delta t & c_2 \Delta t & \cdots & c_N \Delta t \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{\tilde{\mathcal{K}}_d} \underbrace{\begin{bmatrix} s_{1,k} \\ g_{1,k} \\ \left\{ \frac{\partial g_1}{\partial s_1} g_1 \right\}_k \\ \left\{ \frac{\partial g_1}{\partial s_2} g_2 \right\}_k \\ \vdots \\ \left\{ \frac{\partial g_1}{\partial s_N} g_N \right\}_k \end{bmatrix}}_{\Psi(s_k)}. \quad (34)$$

As before, to improve the accuracy of the linear representation, we use data to approximate a Koopman operator.

D. Assessment of Error Bound Estimates

Using the single pendulum system, we demonstrate the error bound estimates for the data-driven linear approximation, both when the dynamics are known and when they are unknown. In particular, we show that $|f_{\max}^{(n+1)}|$ can be computed using the dynamics equations (model-based estimate) when those are available, or approximated using the residue error in the training process when the dynamics are unknown (data-driven estimate), as explained in Section III-B. In both cases, once $|f_{\max}^{(n+1)}|$ is calculated, the error bounds are computed using (22). The states are $s = [\theta, \omega]^T$ and dynamics are given by $\dot{s} = [\omega, \frac{g}{l} \sin(\theta) + u]^T$, where $g = 9.81 \text{ m/s}^2$ is the gravitational constant, $l = 1 \text{ m}$ is the pendulum length, and u is the control.

For the model-based estimate of the error bounds, the maximum magnitude of the n th derivative is (for each state) computed numerically by maximizing the symbolic expression over the domain of the state space that is used for training. For the data-driven estimate, the maximum magnitude is computed using (29) based on the training error. Note that, when propagating a data-driven operator instead of (25), the error bound estimates may in theory be violated.

We sample and forward-simulate 5000 initial states s_0 for $\Delta t = 0.01 \text{ s}$ to obtain a Koopman operator $\tilde{\mathcal{K}}_d$ via (5), which we then use on a different randomly selected set of 5000 states to propagate the dynamics for a time horizon T . In both the training and the testing sets, uniform distributions of the initial states $\mathcal{U}_{\theta_0}(-2\pi \text{ rad}, 2\pi \text{ rad})$ and $\mathcal{U}_{\omega_0}(-5 \text{ rad/s}, 5 \text{ rad/s})$ are used. For each sample, both in training and in testing, we apply random inputs generated from a uniform distribution given by $\mathcal{U}_u(-5 \text{ rad/s}^2, 5 \text{ rad/s}^2)$. The observables include the angle θ and its first three derivatives, derived analytically based on the dynamics equation.

The obtained structure of the data-driven Koopman operator resembles the Taylor-series structure (20) (see Fig. 2), which

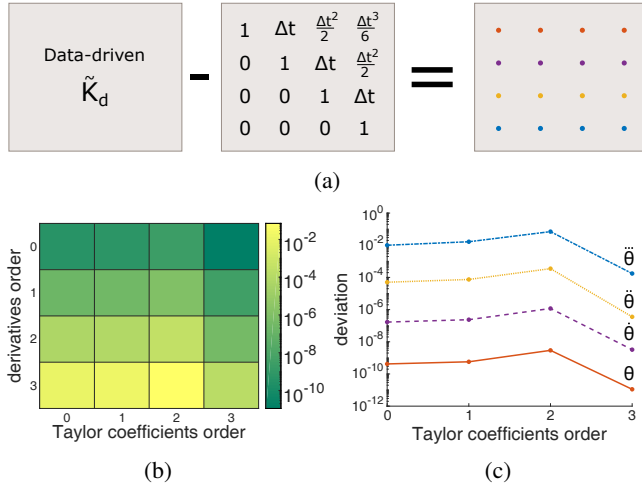


Fig. 2: The deviation of the data-driven Koopman operator from the Taylor-based matrix (20) for the single pendulum system, where the derivative basis functions are constructed analytically from the known dynamics. Fig. 2b shows that the non-zero coefficients (upper triangle) of the linear Taylor expansion are accurately recovered from the data-driven operator. The zero coefficients (lower triangle) are replaced by small values that help minimize the least-squares error for the part of the state space used in the training set. The deviation differs by orders of magnitude across the basis functions, as seen in Fig. 2c. As expected, the deviation is smallest for θ , as it is the one with the highest number of derivatives used in the basis functions.

adds validity to the data-driven error bound estimation. The error bound estimates and the actual errors are shown in Fig. 3. The error bound that is estimated from the structure of the data-driven operator without knowledge of the dynamics is reasonably accurate at predicting the maximum error. In addition, note that the maximum actual error and the error bound estimates have similar slopes with respect to the prediction horizon as well as the fact that both the actual error and the error bound estimates decrease with increasing order of derivatives used as basis functions.

Next, we demonstrate the performance of the derivative-based data-driven Koopman operator and the error bound estimates when dynamics are unknown. Using the single pendulum system, only the angle and angular velocity are measured; higher-order derivatives are estimated using central finite differences [69]. To illustrate the robustness of the approach to noise, we add zero-mean, Gaussian-distributed noise $\mathcal{N}(0, \sigma^2)$ to the measurements of θ and ω , which are then also filtered through a moving average of 15 periods for noise reduction. The higher-order derivatives and the Koopman operator are computed from the filtered measurements. The term $|f_{max}^{(n+1)}|$ is computed as the maximum magnitude of $|f^{(n+1)}|$, which is also calculated using central finite differences [69].

In Fig. 4 we show results for $n = 2$ and two levels of noise: low ($\sigma = \pi/180$) and high noise ($\sigma = 15\pi/180$). Note that the actual maximum error induced by the Koopman

operator remains almost identical, indicating that the presented error bounds can be used even with high levels of noise after using a simple denoising method. The error bound estimate for the low-noise scenario follows closely the error induced by the Koopman operator. In the high-noise scenario, the error bound estimate is more conservative. This is because the error bound formula is highly dependent on the calculated term $|f_{max}^{(n+1)}|$, which is likely to be miscalculated with noisy measurements. Note that if the training data do not represent the entire state space, the calculated value for $|f_{max}^{(n+1)}|$ is likely to underestimate the true value. On the other hand, including some safety margin in the $|f_{max}^{(n+1)}|$ term can make the error bounds more conservative. These results suggest that, although the error bound estimates may become less accurate with increasing levels of noise when dynamics are unknown, simple denoising methods can render the performance of the derivative-based Koopman operator robust to noise, suggesting that the proposed methodology is a promising candidate for the prediction of unknown systems.

In conclusion, the error bounds are derived with respect to the analytical Taylor-based Koopman form of (20). However, we show (Fig. 2) that, in practice, the data-driven Koopman model has a very similar structure to (20) such that the error bounds remain relevant estimates. In addition, we verify that the error bounds reflect reasonably well the prediction error induced by data-driven Koopman models (Fig. 3), even in the presence of noise (Fig. 4). In practice, the error bounds, which depend on the prediction time horizon and the magnitude of the derivatives of the dynamics, provide a systematic method to determine the basis functions for a desired balance between model accuracy and complexity.

Note that, in this work, we do not argue that one should always use additional basis functions than the original system states; instead, one could analyze when and how to augment the basis functions of Koopman operators to improve the modeling accuracy while being cognizant of increased system order and complexity. In particular, the derived error bound estimates can serve as a guide on whether one should do so and by how many derivatives.

E. Comparison to alternative system identification methods

Finally, we compare the performance of the derivative-based data-driven Koopman representation approach to alternative system identification methods using the single pendulum system. Specifically, we use SINDy [70], NARX [71], a data-driven linear model, and the proposed derivative-based data-driven Koopman approach to obtain models and design MPC control to invert the pendulum to the upright position.

To train the models we generate trajectories by forward simulating 500 uniformly-sampled initial states s_0 for 0.04 seconds using time steps of $\Delta t = 0.01$ s. That is, each of the 500 training trajectories contains four measurements, which are used to compute derivatives for the middle measurements needed for the SINDy algorithm and the proposed method, as well as make use of delays for NARX. The initial states are sampled from uniform distributions given by $\mathcal{U}_{\theta_0}(-2\pi \text{ rad}, 2\pi \text{ rad})$ and $\mathcal{U}_{\omega_0}(-5 \text{ rad/s}, 5 \text{ rad/s})$. We

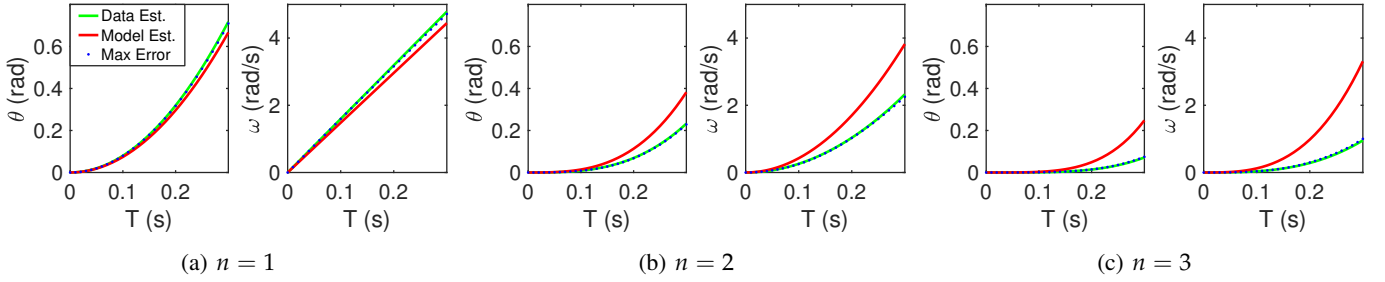


Fig. 3: Simulated error bound estimates and actual error bounds for the single pendulum system as a function of the prediction horizon and for increasing orders of derivatives used as Koopman basis functions. The derivative basis functions are constructed analytically from the known dynamics. Both error bound estimates are calculated using (22), but differ in how they compute $|f_{max}^{(n+1)}|$. **Data Est.** is the model-free error bound estimate and uses the data-driven Koopman operator and (29) to compute $|f_{max}^{(n+1)}|$; **Model Est.** is the model-based error bound estimate and uses the analytical dynamics equations to compute $|f_{max}^{(n+1)}|$; **Max error** is the measured largest deviation as a function of time between the actual value of the state and the one predicted by the data-driven Koopman operator across all trajectories that evolve from randomly sampled initial conditions. Results are shown for three different orders of derivatives of θ . Note that state θ has always one more derivative than ω . The data-driven bound estimates and actual errors can be generated for different parameter choices using a Jupyter notebook at https://colab.research.google.com/drive/1EPX1XVUhr9gix-pZD_3Ydw7Npzz9n3Jj.

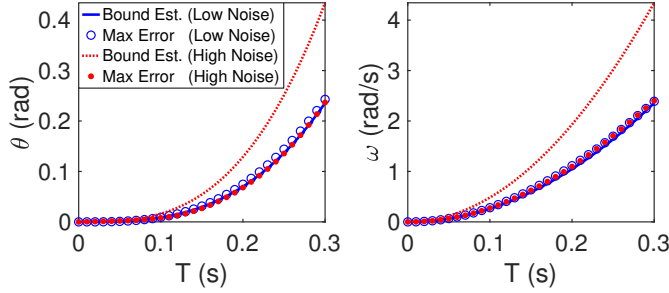


Fig. 4: Simulated error bound estimates and actual maximum errors induced by the data-driven Koopman operators for the single pendulum system when dynamics are unknown and measurements are noisy. The derivative basis functions are calculated numerically from the state measurements—no analytical model is used.

use random inputs generated from a distribution given by $\mathcal{U}_u(-10 \text{ rad/s}^2, 10 \text{ rad/s}^2)$. We train the SINDy model using the open-source package [72] and the NARX model using the MATLAB Deep Learning Toolbox [73]. The higher-order derivatives used for SINDy and the derivative-based Koopman model are numerically estimated from the angle and angular velocity measurements.

Fig. 5 shows simulation results for the inversion of the single pendulum system using MPC control computed from models obtained with NARX, SINDy, a data-driven linear model based only on the system states ($n = 1$, see Fig. 3), and a data-driven Koopman representation based on the state θ and its first- and second-order derivative ($n = 2$). In addition, we also test LQR control on the Koopman model to demonstrate that similar control performance can be achieved with LQR gains that are computed once. The desired states are given by $s_{des} = [0, 0]$, the weights for the states and control are $Q = \text{diag}(5, 0.01)$ and $R = 0.001$, respectively, and the prediction horizon is $T = 0.1$ s.

We also compare the control performance of these methods for 30 initial conditions θ_0 and ω_0 sampled from uniform distributions given by $\mathcal{U}_{\theta_0}(-\pi \text{ rad}, \pi \text{ rad})$ and $\mathcal{U}_{\omega_0}(-2 \text{ rad/s}, 2 \text{ rad/s})$. The average of the 30 errors is 6.00 (with a standard deviation of 6.60) for the Koopman-LQR approach; 6.05 (with a standard deviation of 6.59) for the Koopman-MPC; 6.48 (with a standard deviation of 7.00) for NARX; 6.69 (with a standard deviation of 7.18) for SINDy; and 7.44 (with a standard deviation of 7.70) for the linear model.

These results show that the control performance of the Koopman-MPC method is marginally better than the linear-MPC, NARX-MPC and SINDy-MPC. Note that in simulation the comparative performance of the methods considered is similar using prediction horizons up to $T = 1$ s. Given that NARX is computationally expensive, here we report the results for a short planning horizon so that the performance is closer to real-time execution. Also, the Koopman-LQR approach delivers control performance comparable to the Koopman-MPC method, with the additional benefit that it lends itself to efficient control computation, which makes it an attractive choice for online robotic control applications. This is why we use Koopman-LQR in the simulations and experiments with the robotic fish in Section IV.

IV. DATA-DRIVEN CONTROL OF TAIL-ACTUATED ROBOTIC FISH

We illustrate and validate the proposed data-driven modeling approach using a tail-actuated robotic fish. The states of the robotic fish are $s = [x, y, \psi, v_x, v_y, \omega]^T$, where x, y are the 2D world-frame coordinates, ψ is the orientation, v_x and v_y are the body-frame linear velocities (surge and sway, respectively), and ω is the body-frame angular velocity. We use α to indicate the angle of the tail, actuated with $\alpha(t) = \alpha_o + \alpha_a \sin(\omega_a t)$, where $\alpha_a, \alpha_o, \omega_a$ are the amplitude, bias, and frequency of the tail beat. To simplify the problem, we keep the frequency fixed at $\omega_a = 2\pi \text{ rad/s}$.

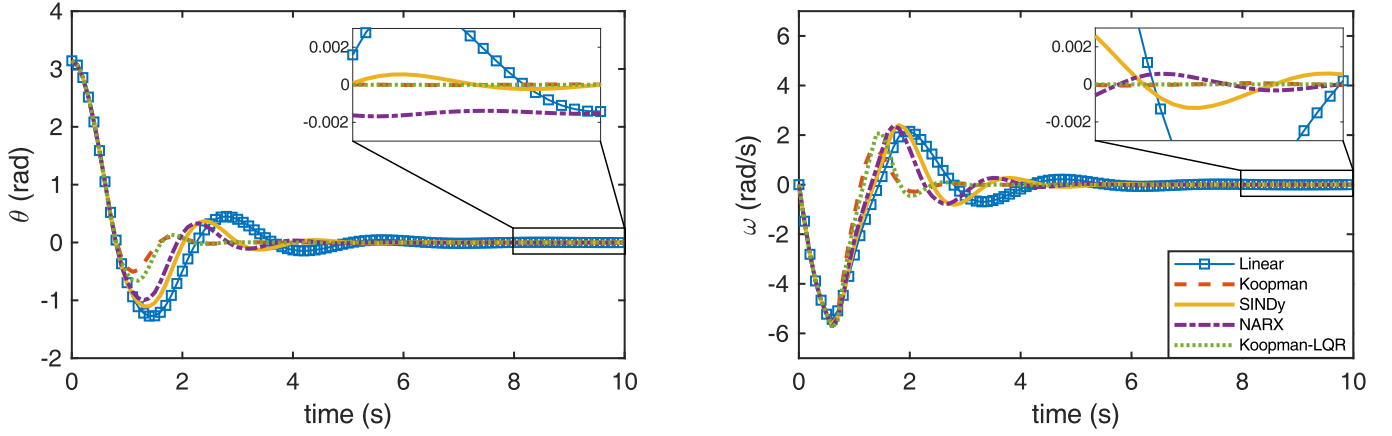


Fig. 5: Control of a pendulum system based on data-driven models obtained using SINDy, NARX, a linear model based only on the system states, and a derivative-based Koopman model whose observables contain the state θ and its first- and second-order derivatives. The derivative basis functions are numerically estimated from state measurements both for training the Koopman model and online to implement control—no analytical model is used. All models are used to design MPC control and, in addition, we use the Koopman model for LQR feedback (Koopman-LQR). Koopman with MPC has the lowest cost (19.71) for the 10 s simulation, while NARX, SINDy and the linear model result in errors that are 12.43%, 18.32%, and 30.61% higher, respectively. Koopman-LQR leads to the second best performance (0.97% higher error in comparison to Koopman-MPC).

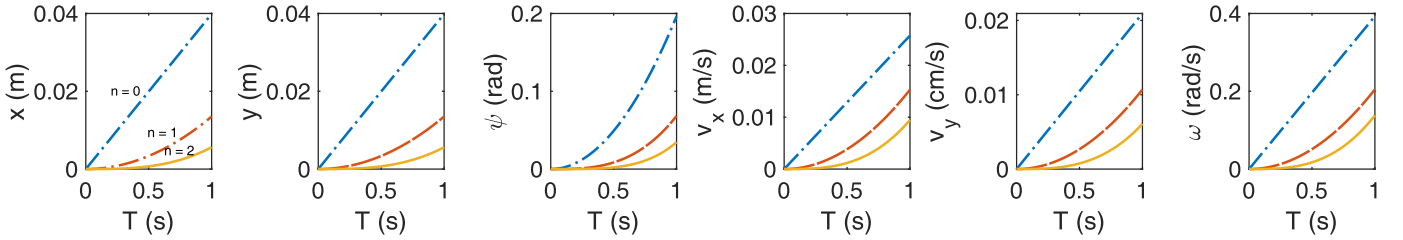


Fig. 6: Error bound estimates based on derivative-based Koopman models of the robotic fish dynamics (35) for increasing order of derivatives used in the basis functions. The derivative basis functions are constructed analytically from the known dynamics. Each additional order of derivatives improves the error bound estimates over the selected prediction horizon. The error bound estimates are computed using (22) where $|f_{max}^{(n+1)}|$ is calculated from the training data.

We describe the dynamics of the system with an average model [74] given by

$$\dot{s} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{\omega} \end{bmatrix} \triangleq \begin{bmatrix} v_x \cos(\psi) - v_y \sin(\psi) \\ v_x \sin(\psi) + v_y \cos(\psi) \\ \omega \\ f_1(s) + K_f f_4(\alpha_0, \alpha_a, \omega_a) \\ f_2(s) + K_f f_5(\alpha_0, \alpha_a, \omega_a) \\ f_3(s) + K_m f_6(\alpha_0, \alpha_a, \omega_a) \end{bmatrix}, \quad (35)$$

where

$$\begin{aligned} f_1(s) &= \frac{m_2}{m_1} v_y \omega - \frac{c_1}{m_1} v_x \sqrt{v_x^2 + v_y^2} + \frac{c_2}{m_1} v_y \sqrt{v_x^2 + v_y^2} \arctan\left(\frac{v_y}{v_x}\right) \\ f_2(s) &= -\frac{m_1}{m_2} v_x \omega - \frac{c_1}{m_2} v_y \sqrt{v_x^2 + v_y^2} - \frac{c_2}{m_2} v_x \sqrt{v_x^2 + v_y^2} \arctan\left(\frac{v_y}{v_x}\right) \\ f_3(s) &= (m_1 - m_2) v_x v_y - c_4 \operatorname{sgn}(\omega) \omega^2 \\ f_4(\alpha_0, \alpha_a, \omega_a) &= \frac{m}{12m_1} L^2 \omega_a^2 \alpha_a^2 (3 - \frac{3}{2} \alpha_o^2 - \frac{3}{8} \alpha_a^2) \\ f_5(\alpha_0, \alpha_a, \omega_a) &= \frac{m}{4m_2} L^2 \omega_a^2 \alpha_a^2 \alpha_o \\ f_6(\alpha_0, \alpha_a, \omega_a) &= -\frac{m}{4J_3} L^2 c \omega_a^2 \alpha_a^2 \alpha_o \end{aligned} \quad (36)$$

and $m_1 = m_b - m_{a_x}$, $m_2 = m_b - m_{a_y}$, $J_3 = J_{b_z} - J_{a_z}$, $c_1 = \frac{1}{2} \rho S C_D$, $c_2 = \frac{1}{2} \rho S C_L$, $c_4 = \frac{1}{J_3} K_D$, $c_5 = \frac{1}{J_3} L^2 m c$. Parameter

m_b is the mass of the robotic fish, m_{a_x} and m_{a_y} are the hydrodynamic derivatives that represent the added masses of the robotic fish along the x and y directions, respectively, J_{a_z} and J_{b_z} are the added inertia effect and the inertia of the body about the z -axis, respectively, m is the mass of the water displaced by the tail per unit length, ρ is the water density, L is the tail length, c is the distance from the body center to the pivot point of the actuated tail, C_D , C_L , K_D are drag force, lift, and drag moment coefficients, respectively, and K_f and K_m are scaling coefficients measured experimentally [21].

We train a Koopman operator using the control-affine form of the dynamics (35) that is obtained by substituting

$$u_1 = \alpha_a^2 (3 - \frac{3}{2} \alpha_o^2 - \frac{3}{8} \alpha_a^2) \quad u_2 = \alpha_a^2 \alpha_o. \quad (37)$$

Because the computed control is in terms of the variables u_1 and u_2 , it needs to be mapped to implementable values for the amplitude, α_a , and the bias, α_o , of the tail flapping. To convert u_1 and u_2 back to the physical actuation variables, we use a constrained global minimization solver based on Sequential Quadratic Programming (SQP) that finds the nearest, in the space of u_1 and u_2 , feasible actuation values for α_a and α_o .

Given u_1 and u_2 , the constrained optimization problem is posed as

$$\begin{aligned} \underset{\alpha_a, \alpha_o}{\operatorname{argmin}} & \sqrt{(u_1 - \alpha_a^2(3 - \frac{3}{2}\alpha_o^2 - \frac{3}{8}\alpha_a^2))^2 + (u_2 - \alpha_a^2\alpha_o)^2} \\ \text{subject to: } & \alpha_a \in [0, 30^\circ] \text{ AND } \alpha_o \in [-45^\circ, 45^\circ]. \end{aligned} \quad (38)$$

This minimization is solved before every control update.

Given the unilateral constraints on the forward motion of the tail-actuated robotic fish (it cannot move backwards), directly tracking position coordinates becomes rather challenging. For example, when the target lies behind the robotic fish, the control solution generates a negative amplitude (to generate backward motion) that is infeasible and thus the system stops moving. This behavior has been observed and tackled in [23] by translating position coordinates into different error states, associated with the body-frame velocities of the system. Similarly, in this work, we argue that one can express all feasible trajectories for the tail-actuated robotic fish in terms of an angle and a forward velocity profile.

A. Simulation Results

In this section, we present simulation results on the data-driven modeling and LQR control on the tail-actuated robotic fish. To decide the optimal order of derivative basis functions, we compare the error bound estimates over $T = 1$ s, which is the feedback rate, using different orders of derivatives. Results are shown in Fig. 6. We select $n = 2$ for a reasonable balance between the increasing complexity of calculating higher-order derivatives and model accuracy. Next, we populate the observable functions with the states, their first- (using (35)) and second-order derivatives, which are derived analytically from the average model. To allow the identification of unknown or changing coefficients (as discussed in Section III-C), we consider each term in the derivatives individually. For example, $\frac{d}{dt}v_y\omega = \dot{v}_y\omega + v_y\dot{\omega}$ generates multiple basis functions, where $\dot{v}_y$ and $\dot{\omega}$ are given by (35). Using separate functions for the time derivatives of each individual term that appears in the dynamics is similar to using the time derivatives of the entire equation of a state (e.g. $\ddot{v}_y(t)$). Despite increasing the number of basis functions, we prefer the first approach because it does not require knowing the coefficients of the individual terms in advance (e.g. $\frac{m_2}{m_1}$). As a result, we can readily train the Koopman operator on other robotic tail-actuated fish that have a different morphology. In this way, we end up with the system states, the control inputs, and 54 additional scalar functions, with $\Psi_s(s) \in \mathbb{R}^{60}$ and $\Psi_u(u) = u \in \mathbb{R}^2$.

Note that we choose control-dependent basis functions to be u so as to use LQR feedback. One can also choose basis functions that include nonlinear control terms in combination with a different control policy, such as NMPC [52], [75]. Alternatively, one can always convert dynamics that are nonlinear in control by introducing new dummy variables (e.g. v_i) as the control input that are the derivatives of the system actuation ($\dot{u}_i = c_i(v_i - u_i)$), where $c_i \in \mathbb{R}^+$ dictate the rate of change [52]. This is in practice also closer to the physical implementation of actuation that cannot instantly

TABLE I: Simulation parameters for the tail-actuated fish model dynamics (35).

Simulation Parameters			
Parameter	Value	Parameter	Value
m_b	0.725 kg	ρ	1000 kg/m ³
m_{ax}	-0.217 kg	S	0.03 m ²
m_{ay}	-0.7888 kg	C_D	0.97
J_{bz}	2.66×10^{-3} kg · m ²	C_L	3.9047
J_{az}	-7.93×10^{-4} kg · m ²	K_D	4.5×10^{-3}
L	0.071 m	K_f	0.7
d	0.04 m	K_m	0.45
c	0.105 m		

change values. In this way, it is then possible to include nonlinearities in u_i , while the system remains still linear with respect to the control input v_i designed by the user.

Next, we train an approximate Koopman operator using (5). To generate data s_k and s_{k+1} , we sample $P = 3000$ initial conditions for the states with uniform distributions given by $\mathcal{U}_{x_0}(-0.5 \text{ m}, 0.5 \text{ m})$, $\mathcal{U}_{y_0}(-0.1 \text{ m}, 0.1 \text{ m})$, $\mathcal{U}_{\psi_0}(-\pi/4 \text{ rad}, \pi/4 \text{ rad})$, $\mathcal{U}_{v_x}(0, 0.04 \text{ m/s})$, $\mathcal{U}_{v_y}(-0.0025 \text{ m/s}, 0.0025 \text{ m/s})$, $\mathcal{U}_{\omega}(-0.5 \text{ rad/s}, 0.5 \text{ rad/s})$. For each sample, we apply random inputs generated from a uniform distribution given by $\mathcal{U}_{\alpha_0}(-45^\circ, 45^\circ)$ for the tail angle bias and $\mathcal{U}_{\alpha_a}(0, 30^\circ)$ for the tail angle amplitude of oscillations. Then, for each sample of initial conditions s_k and controls u_k , we use dynamics (35) and parameters shown in Table I to propagate the states with the given control for $\Delta t = 0.005$ s and obtain the final states s_{k+1} . We use the set of s_k, s_{k+1}, u_k to compute the approximate discrete Koopman operator (5). Note that the value of u_{k+1} can be arbitrary, since we are not trying to predict the evolution of the control-dependent basis functions. Once we have trained the Koopman operator, we convert it to the continuous time via $\tilde{K} = \log(\tilde{K}_d)/\Delta t$, extract the state- and control-linearization matrices A and B , choose the weight matrices Q and R and compute the infinite-horizon LQR gains.

Fig. 7 shows the velocity tracking performance of the derivative-based Koopman model in comparison to a linear data-driven model for the system (35) (with the same set of states as in (35)) when using LQR-feedback. The desired trajectory is a figure-8 described by $s_{des} = [0, 0, 135 \cdot \pi/180 \cdot \sin(0.05t + \pi/2), 0.02, 0, 0.05 \cdot 135 \cdot \pi/180 \cdot \cos(0.05t + \pi/2)]$. The weights are $Q = \text{diag}(0, 0, 0.1, 4000, 0, 0)$ and $R = \text{diag}(0.01, 0.01)$. As is seen in the figure the proposed derivative-based Koopman modeling approach leads to improved control performance.

B. Experimental Results

1) Experimental Setup

We next use the physical robot shown in Fig. 8 to experimentally test our approach. An overhead camera captures the red and blue marks on the robotic fish (see Fig. 8) and calculates the coordinates of its center and its orientation at about 3 Hz. The body-frame velocities are estimated using a Kalman filter. The state derivative functions, provided by the average model, are then evaluated with the states. To simplify the modeling and control task, the tail-beat frequency used in

TABLE II: Amplitude and bias inputs used to collect training dataset.

	Actuation values																			
Amp ($^\circ$)	15					20					25					30				
Bias ($^\circ$)	0	± 20	± 30	± 40	± 50	0	± 20	± 30	± 40	± 50	0	± 20	± 30	± 40	± 50	0	± 20	± 30	± 40	± 50

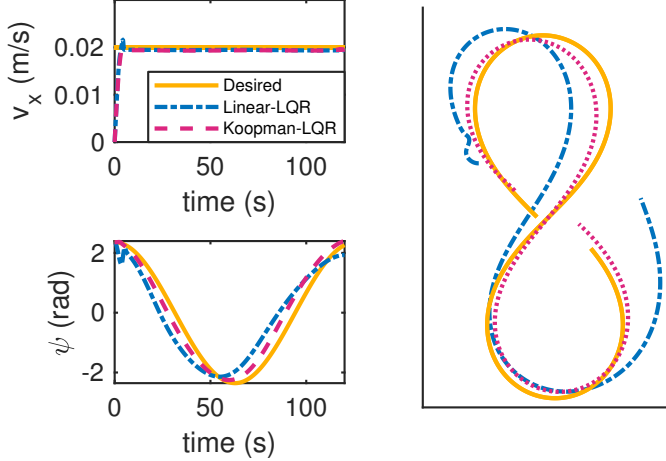


Fig. 7: LQR-controlled robotic fish in simulation. The LQR gains are generated once using the learned Koopman operator. The derivative basis functions are constructed analytically from the known dynamics. The desired trajectory is given in terms of the angle and the forward velocity. Since the position coordinates are not included in the performance objective (11), the controlled trajectories are individually shifted to align with the desired figure-8 shape as closely as possible. Despite using fixed LQR gains, the controlled systems successfully track the desired states that were designed to produce a figure-8 pattern. Koopman-LQR has the lower cost (3.35) for the 120 s duration, while the approach based on the data-driven linear model with the same set of states as in (35) results in an error that is 95% higher.



Fig. 8: Tail-actuated robotic fish used in experiments, developed by the Smart Microsystems Lab at Michigan State University. It maneuvers in water by oscillating its tail fin.

both the training phase and the testing phase is kept constant at $\omega_a = 2\pi$ rad/s. In order not to disturb the periodic movement of the tail oscillation during tracking, feedback control is updated at roughly one second. The control commands are communicated to the robot via Xbee (RF communication).

2) Training Phase

For the training phase, we collect experimental measurements using the robotic fish to compute the approximate Koopman operator. Throughout each run, we apply constant tail bias and amplitude for the oscillations of the tail fin. We conduct a total of 72 runs, with two trials for each of the 36 different combinations of actuation parameters, shown in Table II. We train the Koopman operator using the same basis functions as discussed in the simulation section.

To create a consistent mapping with the Koopman operator, all pairs of measurements s_k and s_{k+1} need to be spaced (in time) equally apart, as we explain in Section II. For this reason, and also to decrease the time between measurements to fine levels (without the constraints of our sampling and filtering methods), the obtained data is interpolated at $\Delta t = 0.005$ s. The interpolated data is then used to obtain an approximate Koopman operator according to (4).

To measure how well the Koopman model captures the nonlinear dynamics of the robotic fish, we use $\hat{K}$, learned from the experimental data, to propagate the identified model continuously based on the initial states of each of the 72 experimental runs. Then, the predicted simulated trajectories are compared against the corresponding experimental ones. For the purposes of illustration, two such comparisons are shown in Fig. 9. The linear Koopman model, despite not perfect, reasonably follows the experimental data for at least five seconds. Note that, because we only minimize the single-step prediction error (5), it is more likely (compared to minimizing a multi-step error) that we compute an unstable Koopman operator, even if the dynamics are stable. In that case, the long-term predictions would exponentially deviate and become inaccurate. Imposing stability properties on the operator is the focus of ongoing work [76], [77].

3) Testing Phase

We use the data-trained Koopman operator to implement linear feedback control (LQR) for tracking. Using the weight matrices Q and R , which penalize the tracking error and control effort, respectively, we define the minimization problem (11) and calculate the infinite-horizon LQR gains. Contrary to work in [49], and in order to illustrate the simplicity and robustness of the proposed scheme, we keep the same weights across all different tasks, such that the same LQR gains, (unless the model is updated) are used in every type of trajectory. The resulting feedback has the form shown in (13). As mentioned earlier, we argue that, to follow any trajectory, it suffices to track a desired orientation and forward velocity and so we design the LQR weights accordingly. Specifically,

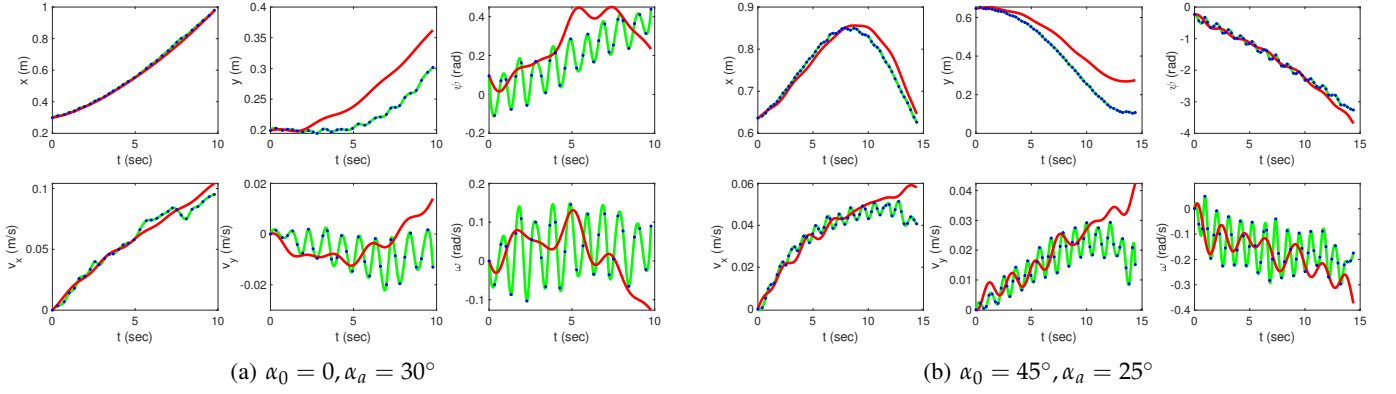


Fig. 9: Fitness between Koopman model and experimental measurements. The green line shows data interpolated from experimental measurements (blue dots) every $\Delta t = 0.005$ s. The red line shows the evolution of the states using the Koopman model. The actuation is constant for each of the two runs and is indicated in the caption.

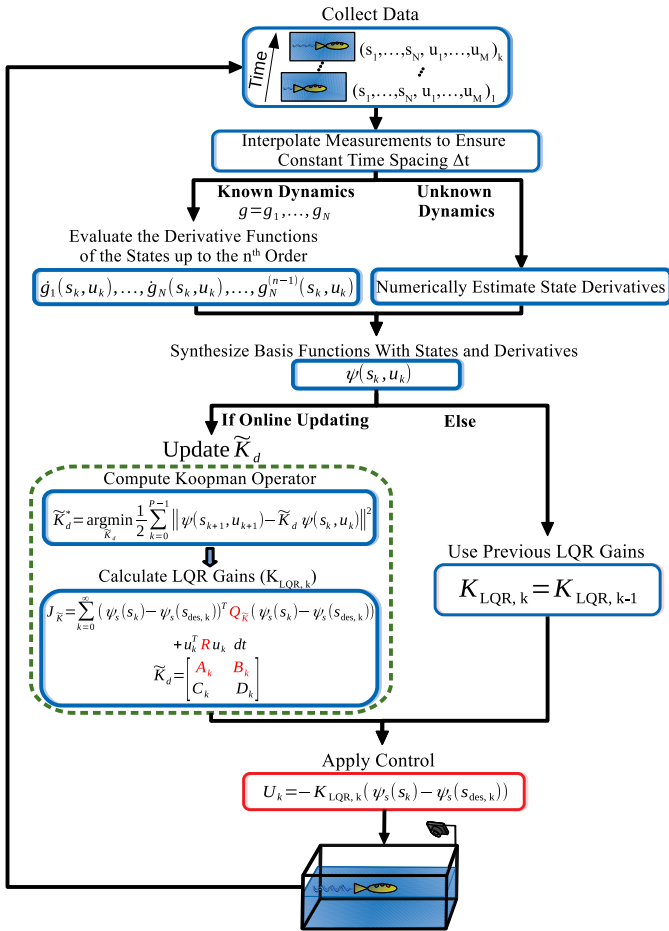


Fig. 10: Outline of the proposed methodology for LQR control using derivative-based Koopman operators.

the weights used are $Q = \text{diag}(0, 0, 0.1, 1000, 0, 0)$ and $R = \text{diag}(1, 1)$. The weights for the angle and forward velocity are disproportionate to account for the difference in scale between the velocity of the robotic fish, typically in the order of 0.01 m/s, and the body orientation, expressed in radians.

We implement the proposed data-driven Koopman methodology in two ways. One approach is computing the model and

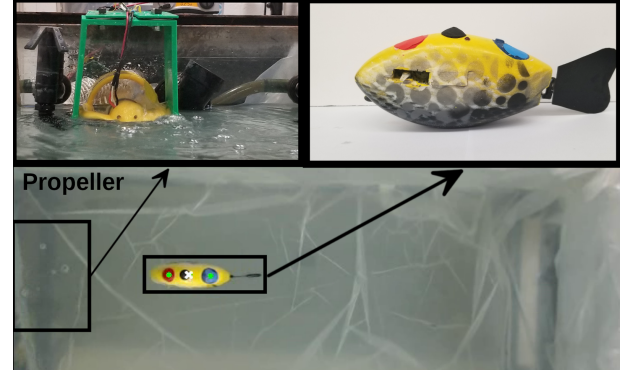


Fig. 11: Experimental setup for creating fluid disturbances. The motor is halfway submerged in water, generating ripples with its propellers.

LQR gains offline once; the other is updating the model and recalculating the LQR gains online in real time. To update the Koopman operator online in a memory-efficient manner, we do not store any previous measurements. Instead, we use (5) and split it into the P measurements used last to calculate the Koopman operator and the ΔP new measurements since the last update, where $P_{\text{total}} = P + \Delta P$ is the total number of measurements used. Then,

$$\tilde{K}_{d,\text{new}}^* = \mathcal{A}_{\text{new}} \mathcal{G}_{\text{new}}^+ \quad (39)$$

where

$$\begin{aligned} \mathcal{A}_{\text{new}} &= \frac{1}{P_{\text{total}}} \left(\mathcal{A}P + \sum_{k=P}^{P_{\text{total}}-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T \right) \\ \mathcal{G}_{\text{new}} &= \frac{1}{P_{\text{total}}} \left(\mathcal{G}P + \sum_{k=P}^{P_{\text{total}}-1} \Psi(s_k, u_k) \Psi(s_k, u_k)^T \right) \end{aligned} \quad (40)$$

and $\mathcal{A}$ and $\mathcal{G}$ are given by (6). The derivation of the formula is shown in Appendix B-A. Then, the LQR gains are recomputed as shown in Section II-D using the updated Koopman operator. We show an outline of the process in Fig. 10.

The proposed method is compared to a PID controller, a widely-used model-free control method. PID feedback is

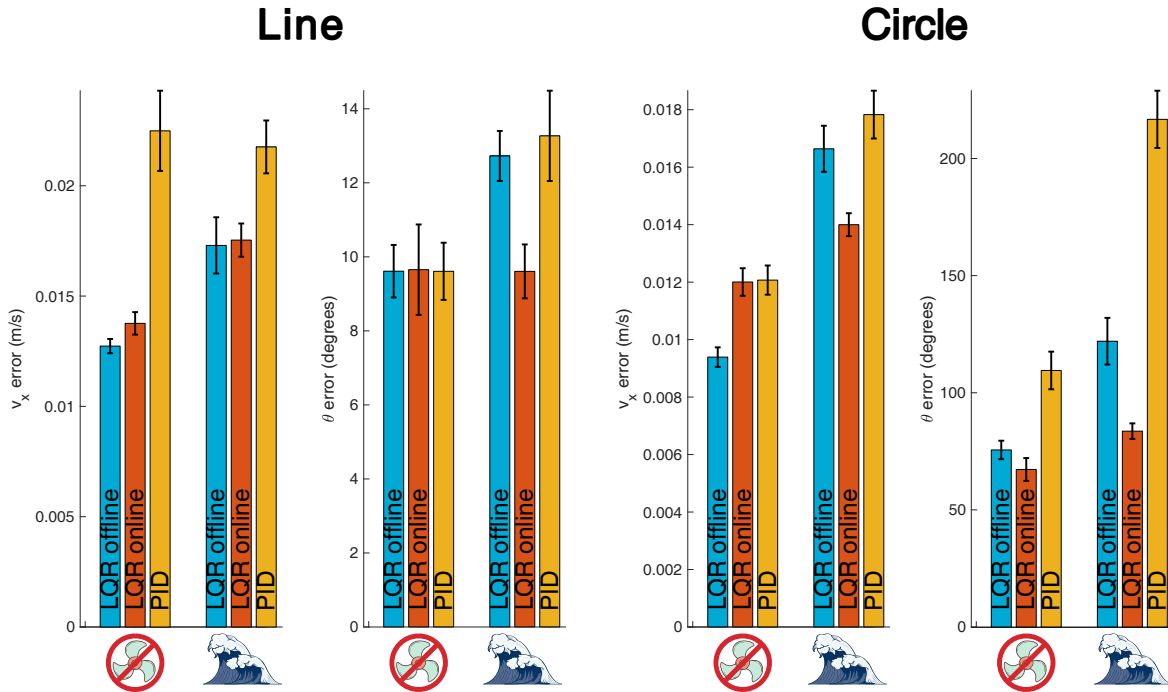


Fig. 12: Experimental results: Average error scores for velocity and angle tracking for PID and two variants of Koopman-LQR, trained offline and updated online, with and without fluid flow indicated respectively with the waves and no-fan icons. The four subplots compare the performance for the linear and circular trajectories for each state separately. The error bars indicate the standard error. The proposed Koopman operator scheme outperforms PID in all tests. Further, updating Koopman-LQR online improves the performance in the presence of the unmodeled fluid flow. A video of experimental runs is shown at https://youtu.be/9_wx0tdDta0.

quite effective, requires low computational effort, and does not require exact knowledge of the dynamics or any of the model coefficients [78]. In fact, it has been shown to perform remarkably well on motion and speed control of robotic fish [12], [25]. However, this method often requires intensive gain tuning. To tune the PID controller, we utilize the Ziegler-Nichols' closed-loop method [79]. By comparing our method to PID, we hope to demonstrate that our proposed method is a promising, data-driven feedback scheme that compares well against a popularly used, effective controller, yet without the additional overhead of intensive parameter tuning. The two methods are compared on tracking linear and circular trajectories that are described in terms of the desired orientation and forward velocity.

The comparison of the proposed Koopman-LQR method and the PID is conducted over ten trials for both types of trajectories. We further compare their performance in the presence of fluid disturbance, generated by a propeller (see Fig. 11). The results are presented in Fig. 12, which displays the average error in the two tracked states, together with the standard error. The standard error is a measure of the expected variability in the average error, contrary to the standard deviation that measures the expected variability away from the mean. Note that, for the case of LQR with gains computed offline, the data used to obtain the Koopman representation was collected in the absence of fluid disturbance.

From the results, the proposed data-driven Koopman-LQR scheme, regardless of whether it is updated online, outper-

forms PID in all tasks, with or without fluid disturbance, except for tracking the orientation in the linear trajectory, where both algorithms have similar performance. The difference in the performance between the Koopman-LQR and the PID is highlighted in the tracking of the circular trajectory in the presence of fluid flow, where the angle error is significantly higher for PID (Fig. 12). Given that angle tracking (in the linear trajectory) is the only metric where PID is comparable to our proposed method, we attribute the difference in performance in the presence of fluid flow to the robustness of our approach and its ability to track the desired trajectory in a confined space in the presence of unmodeled dynamics.

In the absence of fluid disturbance, the two implementations of the proposed method (that is, with and without updating the model and LQR gains in real time) have comparable results. It is only in tracking the circular trajectory that the offline implementation tracks the desired velocity better. We conjecture that this is due to the collisions of the robotic fish with the side wall (and the unmodeled boundary conditions) that take place because of the confined space. Introducing such discontinuous disturbances likely deteriorates the learned model temporarily; yet it still outperforms the well tuned PID controller. On the other hand, the major benefit of updating the model online is, as one would expect, in the presence of fluid disturbance. There, the real-time updated method significantly outperforms the offline-trained Koopman-LQR scheme, highlighting the importance of updating the model online in environments that constantly change.

V. DISCUSSION AND FUTURE WORK

In this paper, we use the Koopman operator framework to develop derivative-based data-driven linear representations of nonlinear systems, suitable for real-time feedback. The proposed synthesis of the observable functions aims at minimizing the representation error without requiring knowledge of the dynamics. Utilizing Taylor series error bounds, we characterize the approximation error induced by the Koopman operator and use the error bound formula to decide the order of derivatives for synthesizing the basis functions. LQR is then used as an example of efficient control synthesis based on the trained Koopman operator. In fact, unless the model is updated online, the LQR gains are computed only once. We demonstrate the efficacy of our approach with simulation and experimental results using a case study of a tail-actuated robotic fish.

One of the most promising aspects of the derivative-based synthesis of Koopman operators is the application to completely unknown dynamics. In that case, the proposed method relies on numerically estimating the state derivatives, amplifying any noise present in the measurements and possibly worsening the accuracy of the error bound estimates. As simulation results suggest, however, after implementing a simple moving average filter, the performance of the Koopman operator remains robust even for high noise levels. While the error bound estimates may be violated, they remain close to the true bounds. Such results merit further research on more complicated dynamics using more sophisticated noise reduction schemes. In fact, as part of early-stage efforts of underwater exploration, we have tested our derivative-based Koopman approach to the unknown dynamics of a soft robotic fish [80], where we numerically estimate state derivatives using high-gain observers and are able to predict with reasonable accuracy the evolution of the states.

The modeling performance of the proposed Koopman approach is comparable to state-of-the-art data-driven modeling methods, such as SINDy and NARX. This is in part shown via the comparison of MPC control performance, based on these different modeling schemes. On the other hand, as a linear model, the Koopman approach lends itself readily to efficient control design tools, an example of which is LQR. In fact, LQR based on a Koopman model delivers control performance comparable to these MPC-based methods. In short, this work develops a systematic approach for constructing the Koopman operator with high fidelity, which can be used for various model-based control schemes (as illustrated here with MPC and LQR). The advantage over other nonlinear data-driven methods is the linear nature of the model, which allows us to more efficiently compute the controller (e.g., LQR). This also justifies why we use LQR in the later simulation and experiments for the robotic fish example.

Although the proposed method can be used for any system that can benefit from data-driven methods or reduction of the nonlinearity, underwater robotics is perhaps the most suitable application for this method, due to the inherent environmental uncertainty, the highly nonlinear dynamics, and the need for controllers that use limited computation (to preserve battery use or due to limited computational power). While this method

could certainly be applied to other systems, it perhaps would not be as useful for low-dimensional systems, with known dynamics and few nonlinearities.

This work can be further expanded in multiple ways. We believe that the modeling is worsened by the average dynamics (35) used to describe the longer-term behavior of the tail-actuated fish. In fact, the average model borrowed from [74] is inaccurate, because it assumes that the tail bias has no effect on the angular velocity if the amplitude is zero ($u_2 = 0$ when $\alpha_a = 0$). However, when the system already has forward velocity, the orientation of the tail certainly induces a rotation to the movement. Thus, abandoning the average model, and instead using Kirchoff's equations for a rigid body in fluid environment, could improve the model fitness. Alternatively, one could also use a system identification algorithm, such as SINDy [38], to first obtain a model for the nonlinear dynamics of the system. Besides identifying the underlying dynamics, we are also interested in experimentally testing this methodology on entirely unknown dynamics where we use measurements to numerically compute the derivatives of the system states and populate the Koopman basis functions. Robotic fish with different morphologies, such as fish propelled with pectoral fish, are possible candidates for further testing. We are also interested in computing error bounds for general basis functions, not limited to derivatives of the dynamics.

Last, note that work in [81] uses Koopman operators with time delay observables to model and control (also using LQR feedback) a soft robotic arm. The authors comment that the time delay observables help better capture the momentum of the dynamics. We believe that time delay observables are equivalent to derivative-based observables given that the latter can be approximated numerically from past measurements [68]. Interestingly enough, the authors in [81] observe a steady increase in predictive power as they increase the number of time delays in the observables, which we consider to be analogous to including higher-order derivatives in the basis functions. Investigating the connection between time delay observables and derivative-based observables can reveal whether the error bound formula presented in this work can be also utilized in other applications of Koopman models, without the need to numerically estimate derivatives [82]–[85].

ACKNOWLEDGMENT

We would like to greatly thank the anonymous reviewers for their valuable feedback.

APPENDIX A GLOBAL ERROR FOR TAYLOR-BASED KOOPMAN OPERATORS

In each time step, there is error induced in the updated function. Let $e_k^{(m)}$ indicate the deviation from the accurate

value of the $f_k^{(m)}$ function at time t_k , where $m \in \mathbb{Z} \cap [0, n]$. That is,

$$\begin{aligned} e_k &= \tilde{f}_k - f_k \\ e'_k &= \tilde{f}'_k - f'_k \\ &\vdots \\ e_k^{(n)} &= \tilde{f}_k^{(n)} - f_k^{(n)} \end{aligned} \quad (41)$$

Next, consider how previous errors accumulate in the prediction of a function:

$$\begin{aligned} \tilde{f}_{k+1} &= \tilde{f}_k + \tilde{f}'_k \cdot \Delta t + \tilde{f}''_k \cdot \frac{\Delta t^2}{2!} + \cdots + \tilde{f}_k^{(n)} \frac{\Delta t^n}{n!} \\ &= (f_k + e_k) + (f'_k + e'_k) \cdot \Delta t + (f''_k + e''_k) \cdot \frac{\Delta t^2}{2!} \\ &\quad + \cdots + (f_k^{(n)} + e_k^{(n)}) \frac{\Delta t^n}{n!} \\ &= f_k + f'_k \cdot \Delta t + f''_k \frac{\Delta t^2}{2!} + \cdots + f_k^{(n)} \frac{\Delta t^n}{n!} \\ &\quad + \underbrace{e_k + e'_k \cdot \Delta t + e''_k \cdot \frac{\Delta t^2}{2!} + \cdots + e_k^{(n)} \frac{\Delta t^n}{n!}}_{\text{error terms}}, \end{aligned} \quad (42)$$

such that

$$\begin{aligned} e_{k+1} &= e_k + e'_k \cdot \Delta t + e''_k \cdot \frac{\Delta t^2}{2!} + \cdots + e_k^{(n)} \frac{\Delta t^n}{n!} \\ &\quad + \frac{\Delta t^{n+1}}{(n+1)!} f_{k,k+1}^{(n+1)}, \end{aligned} \quad (43)$$

where the last error term is from Lagrange's remainder formula and is added at each step. Remember, $f_{k,k+1}^{(n)}$ is the n th derivative of a function evaluated at some $t \in [t_k, t_{k+1}]$. Similarly,

$$\begin{aligned} e_k &= e_{k-1} + e'_{k-1} \cdot \Delta t + e''_{k-1} \cdot \frac{\Delta t^2}{2!} + \cdots + e_{k-1}^{(n)} \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f_{k-2,k-1}^{(n+1)} \\ &\vdots \\ e_3 &= e_2 + e'_2 \cdot \Delta t + e''_2 \cdot \frac{\Delta t^2}{2!} + \cdots + e_2^{(n)} \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f_{2,3}^{(n+1)} \\ e_2 &= e_1 + e'_1 \cdot \Delta t + e''_1 \cdot \frac{\Delta t^2}{2!} + \cdots + e_1^{(n)} \frac{\Delta t^n}{n!} + \frac{\Delta t^{n+1}}{(n+1)!} f_{1,2}^{(n+1)}, \end{aligned} \quad (44)$$

where $e_1 = f_{0,1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!}$; in the first iteration, we assume the knowledge of the exact values of all derivatives up to order n , such that the only numerical error comes from the inaccuracy of the Taylor series expansion. Without loss of generality, the functions are assumed to be known exactly at $k = 0$. That is, $e_0 = 0, e'_0 = 0, \dots, e_0^{(n)} = 0$. Thus, we can express the error of a function f as

$$\begin{aligned} e_k &= \left(\sum_{i=1}^{k-1} e'_i \cdot \Delta t + e''_i \cdot \frac{\Delta t^2}{2!} + \cdots + e_i^{(n)} \frac{\Delta t^n}{n!} + f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!} \right) \\ &\quad + f_{0,1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!} \end{aligned} \quad (45)$$

Similarly, for the errors associated with the higher-order terms, we have

$$\begin{aligned} e'_k &= \left(\sum_{i=1}^{k-1} e''_i \cdot \Delta t + e'''_i \cdot \frac{\Delta t^2}{2!} + \cdots + e_i^{(n)} \frac{\Delta t^{n-1}}{(n-1)!} \right. \\ &\quad \left. + f_{i,i+1}^{(n+1)} \frac{\Delta t^n}{(n)!} + f_{0,1}^{(n+1)} \frac{\Delta t^n}{(n)!} \right) \\ &\vdots \\ e_k^{(n-1)} &= \left(\sum_{i=1}^{k-1} e_i^{(n)} \cdot \Delta t + f_{i,i+1}^{(n+1)} \frac{\Delta t^{(n+1)-(n-1)}}{((n+1)-(n-1))!} \right. \\ &\quad \left. + f_{0,1}^{(n+1)} \frac{\Delta t^{(n+1)-(n-1)}}{((n+1)-(n-1))!} \right) \\ e_k^{(n)} &= \left(\sum_{i=1}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1-n}}{(n+1-n)!} + f_{0,1}^{(n+1)} \frac{\Delta t^{n+1-n}}{(n+1-n)!} \right) \end{aligned} \quad (46)$$

In general, the error of the p -th derivative of a function f at time t_k is given by

$$\begin{aligned} e_k^{(p)} &= \left(\sum_{i=1}^{k-1} \left(\left(\sum_{j=1}^{n-p} e_i^{(j)} \frac{\Delta t^j}{j!} \right) + f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!} \right) \right) \\ &\quad + f_{0,1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!}, \end{aligned} \quad (47)$$

which can be simplified to

$$e_k^{(p)} = \sum_{i=1}^{k-1} \sum_{j=1}^{n-p} e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1-p}}{(n+1-p)!}, \quad (48)$$

which is split into the error from the derivative inaccuracies and the error induced by the Taylor series expansion at each step. Note that n is the number of derivatives used to propagate f , where $p \in \mathbb{Z} \cap [0, n]$ indicates the order of the derivative of a function for which the error is calculated ($p = 0$ refers to the original function f).

APPENDIX B

GLOBAL ERROR BOUNDS FOR TAYLOR-BASED KOOPMAN OPERATOR

First, consider the error in f when no derivative basis functions are used, that is, $n = 0$. Then,

$$e_k = \sum_{i=0}^{k-1} f_{i,i+1}^{(1)} \frac{\Delta t^1}{1!} \quad (49)$$

$$|e_k| = \left| \sum_{i=0}^{k-1} f_{i,i+1}^{(1)} \frac{\Delta t^1}{1!} \right| \quad (50)$$

$$|e_k| \leq \frac{\Delta t^1}{1!} \sum_{i=0}^{k-1} |f_{i,i+1}^{(1)}| \quad (51)$$

To further simplify the analysis, we can assume a maximum value of $f^{(1)}$, $|f_{i,i+1}^{(1)}| \leq |f_{\max}^{(1)}|$ for all $i \in \mathbb{Z} \cap [0, k-1]$. Then,

$$|e_k| \leq \frac{\Delta t^1}{1!} \sum_{i=0}^{k-1} |f_{\max}^{(1)}| \quad (52)$$

$$|e_k| \leq k \cdot \Delta t \cdot |f_{\max}^{(1)}| = T \cdot |f_{\max}^{(1)}|, \quad (53)$$

where $T \triangleq k \cdot \Delta t$ is the time window that we are approximating the function over.

Similarly, we compute the error bound for $n = 1$ (one derivative of f in the basis functions). Then,

$$e_k = \sum_{i=1}^{k-1} e'_i \Delta t + \sum_{i=0}^{k-1} f_{i,i+1}^{(2)} \frac{\Delta t^2}{2!}, \quad (54)$$

where e'_i is given by (49). Note that $f^{(1)}$ becomes $f^{(2)}$; what was e_i is now e'_i and the first-order derivative of e'_i is the second-order derivative of e_i . Thus,

$$e_k = \sum_{i=1}^{k-1} \left(\sum_{j=0}^{i-1} f_{j,j+1}^{(2)} \frac{\Delta t^1}{1!} \right) \Delta t + \sum_{i=0}^{k-1} f_{i,i+1}^{(2)} \frac{\Delta t^2}{2!} \quad (55)$$

$$|e_k| \leq \sum_{i=1}^{k-1} \left(\sum_{j=0}^{i-1} |f_{j,j+1}^{(2)}| \frac{\Delta t^1}{1!} \right) \Delta t + \frac{\Delta t^2}{2!} \sum_{i=0}^{k-1} |f_{i,i+1}^{(2)}| \quad (56)$$

To further simplify things, we again use $|f_{i,i+1}^{(2)}| \leq |f_{\max}^{(2)}|$ for all $i \in \mathbb{Z} \cap [0, k-1]$, such that

$$|e_k| \leq \sum_{i=1}^{k-1} \left(\sum_{j=0}^{i-1} |f_{\max}^{(2)}| \frac{\Delta t^1}{1!} \right) \Delta t + \frac{\Delta t^2}{2!} \sum_{i=0}^{k-1} |f_{\max}^{(2)}| \quad (57)$$

$$|e_k| \leq \sum_{i=1}^{k-1} i |f_{\max}^{(2)}| \Delta t^2 + k \cdot \frac{\Delta t^2}{2} |f_{\max}^{(2)}| \quad (58)$$

$$|e_k| \leq |f_{\max}^{(2)}| \Delta t^2 \sum_{i=1}^{k-1} i + k \cdot \frac{\Delta t^2}{2} |f_{\max}^{(2)}|. \quad (59)$$

Using the property $\sum_{i=0}^n i = \sum_{i=1}^n i = \frac{n(n+1)}{2}$,

$$|e_k| \leq |f_{\max}^{(2)}| \Delta t^2 \frac{(k-1)k}{2} + k \cdot \frac{\Delta t^2}{2} |f_{\max}^{(2)}| \quad (60)$$

$$|e_k| \leq |f_{\max}^{(2)}| \Delta t^2 \frac{k^2 - k}{2} + k \cdot \frac{\Delta t^2}{2} |f_{\max}^{(2)}| \quad (61)$$

$$|e_k| \leq \frac{(k \cdot \Delta t)^2}{2} |f_{\max}^{(2)}| = \frac{T^2}{2} |f_{\max}^{(2)}|. \quad (62)$$

From $n = 0$ and $n = 1$, we notice a pattern in the error bound expression. Using proof by induction, we next show that the error bound is given by

$$|e_k| \leq \frac{T^{n+1}}{(n+1)!} |f_{\max}^{(n+1)}|. \quad (63)$$

Base Case: $n = 0$

From (53), it is true that, for $n = 0$,

$$|e_k| \leq T \cdot |f_{\max}^{(1)}|. \quad (64)$$

Induction Step:

Assuming that the relationship holds for $n - 1$, we show it is also true for n . For the case of using basis functions with derivatives up to order n , the error in the derivative function of order $p = 0$ is, using (48), given by

$$e_k = \sum_{i=1}^{k-1} \sum_{j=1}^n e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!}. \quad (65)$$

Taking the absolute value of the error,

$$|e_k| = \left| \sum_{i=1}^{k-1} \sum_{j=1}^n e_i^{(j)} \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} f_{i,i+1}^{(n+1)} \frac{\Delta t^{n+1}}{(n+1)!} \right| \quad (66)$$

$$\leq \sum_{i=1}^{k-1} \sum_{j=1}^n |e_i^{(j)}| \frac{\Delta t^j}{j!} + \sum_{i=0}^{k-1} |f_{i,i+1}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}, \quad (67)$$

where $\frac{\Delta t^j}{j!}$ and $\frac{\Delta t^{n+1}}{(n+1)!}$ are non-negative. Then, we use the relationship to substitute for the terms $|e_i^{(j)}|$. Note that $|e_i^{(1)}|$ is equivalent to $|e_k|$ using $n - 1$ derivatives. Similarly, each term $|e_i^{(j)}|$ for $j \in \mathbb{Z} \cap [1, n]$ is equivalent to $|e_k|$ for $n - j$ derivatives. Thus, we use the relationship that holds for up to $n - 1$ derivatives, such that

$$\begin{aligned} |e_k| &\leq \sum_{i=1}^{k-1} \sum_{j=1}^n \left| \frac{(i\Delta t)^{n+1-j}}{(n+1-j)!} f_{\max}^{(n+1)} \right| \frac{\Delta t^j}{j!} \\ &\quad + \sum_{i=0}^{k-1} |f_{i,i+1}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!} \\ &= |f_{\max}^{(n+1)}| \Delta t^{n+1} \sum_{i=1}^{k-1} \sum_{j=1}^n \frac{i^{n+1-j}}{(n+1-j)! j!} \\ &\quad + |f_{\max}^{(n+1)}| \sum_{i=0}^{k-1} \frac{\Delta t^{n+1}}{(n+1)!}, \end{aligned} \quad (68)$$

where $|f_{\max}^{(n+1)}| \geq |f_{i,i+1}^{(n+1)}|$. Next, we use $j' = n + 1 - j$ to simplify the inner sum term

$$\sum_{j=1}^n \frac{i^{n+1-j}}{(n+1-j)! j!} = \sum_{j'=n}^1 \frac{i^{j'}}{(j')! (n+1-j')!}, \quad (69)$$

which can be rewritten for j and the summation order can also be reversed such that

$$\begin{aligned} |e_k| &\leq |f_{\max}^{(n+1)}| \Delta t^{n+1} \sum_{i=1}^{k-1} \sum_{j=1}^n \frac{i^j}{(n+1-j)! j!} \\ &\quad + |f_{\max}^{(n+1)}| \sum_{i=0}^{k-1} \frac{\Delta t^{n+1}}{(n+1)!}. \end{aligned} \quad (70)$$

We can simplify

$$|f_{\max}^{(n+1)}| \sum_{i=0}^{k-1} \frac{\Delta t^{n+1}}{(n+1)!} = k |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}, \quad (71)$$

such that

$$\begin{aligned} |e_k| &\leq |f_{\max}^{(n+1)}| \Delta t^{n+1} \sum_{i=1}^{k-1} \sum_{j=1}^n \frac{i^j}{(n+1-j)! j!} \\ &\quad + k |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!} \\ &= \left[\sum_{i=1}^{k-1} \sum_{j=1}^n \frac{i^j}{(n+1-j)! j!} + \frac{k}{(n+1)!} \right] |f_{\max}^{(n+1)}| \Delta t^{n+1}. \end{aligned} \quad (72)$$

Using the binomial coefficient

$$\frac{a!}{(a-b)! b!} = \binom{a}{b}, \quad (73)$$

to rewrite the term $(n+1-j)!j!$,

$$|e_k| \leq \left[\sum_{i=1}^{k-1} \sum_{j=1}^n \frac{j!}{(n+1)!} \binom{n+1}{j} + \frac{k}{(n+1)!} \right] |f_{\max}^{(n+1)}| \Delta t^{n+1}. \quad (74)$$

Switching the summation order,

$$|e_k| \leq \left[\sum_{j=1}^n \binom{n+1}{j} \sum_{i=1}^{k-1} j! + k \right] |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}. \quad (75)$$

To use Pascal's identity [86]:

$$\sum_{p=0}^k \binom{k+1}{p} \sum_{j=1}^n j^p = (n+1)^{k+1} - 1, \quad (76)$$

we rewrite the error bound as

$$|e_k| \leq \left[\sum_{j=0}^n \binom{n+1}{j} \sum_{i=1}^{k-1} j! - \binom{n+1}{0} \sum_{i=1}^{k-1} i^0 + k \right] |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!}, \quad (77)$$

such that

$$\begin{aligned} |e_k| &\leq \left[(k^{n+1} - 1) - (k-1) + k \right] |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!} \\ &= k^{n+1} |f_{\max}^{(n+1)}| \frac{\Delta t^{n+1}}{(n+1)!} \\ &= \frac{T^{n+1}}{(n+1)!} |f_{\max}^{(n+1)}|, \end{aligned} \quad (78)$$

where $T = k \cdot \Delta t$. Therefore,

$$|e_k| \leq \frac{T^{n+1}}{(n+1)!} |f_{\max}^{(n+1)}| \text{ for all } n \in \mathbb{Z}^{\geq 0}. \quad (79)$$

A. Incremental Update of Koopman Operator

Consider P measurements used to last update the Koopman operator and ΔP new measurements. We incrementally compute a Koopman operator using all $P_{\text{total}} = P + \Delta P$ measurements as follows. The Koopman operator is computed using (5), which we split the expression into past and new measurements, such that

$$\begin{aligned} \mathcal{A}_{\text{new}} &= \frac{1}{P_{\text{total}}} \sum_{k=0}^{P_{\text{total}}-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T \\ &= \frac{1}{P_{\text{total}}} \left(\sum_{k=0}^{P-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T \right. \\ &\quad \left. + \sum_{k=P}^{P_{\text{total}}-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T \right) \end{aligned} \quad (80)$$

and, using

$$\mathcal{A} = \frac{1}{P} \sum_{k=0}^{P-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T, \quad (81)$$

we can rewrite $\mathcal{A}_{\text{new}}$ as

$$\mathcal{A}_{\text{new}} = \frac{1}{P_{\text{total}}} (P\mathcal{A} + \sum_{k=P}^{P_{\text{total}}-1} \Psi(s_{k+1}, u_{k+1}) \Psi(s_k, u_k)^T). \quad (82)$$

Similarly,

$$\mathcal{G}_{\text{new}} = \frac{1}{P_{\text{total}}} (\mathcal{G}P + \sum_{k=P}^{P_{\text{total}}-1} \Psi(s_k, u_k) \Psi(s_k, u_k)^T). \quad (83)$$

REFERENCES

- [1] D. Mayne, *Nonlinear Model Predictive Control: Challenges and Opportunities*, F. Allgöwer and A. Zheng, Eds. Basel: Birkhäuser Basel, 2000, vol. 26.
- [2] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.
- [3] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*. New York: Wiley, 1972, vol. 1.
- [4] A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming—the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.
- [5] F. Allgöwer and A. Zheng, *Nonlinear Model Predictive Control*. Basel: Birkhäuser, Basel, 2012, vol. 26.
- [6] A. Isidori, *Nonlinear Control Systems*, 3rd ed. London, UK: Springer, 1995.
- [7] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. New York: American Elsevier, 1970.
- [8] A. R. Ansari and T. D. Murphey, "Sequential action control: closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1196–1214, 2016.
- [9] E. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proceedings of the American Control Conference*, Baltimore, MD, USA, 2010, pp. 1125–1132.
- [10] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the American Control Conference*, Portland, OR, USA, 2005, pp. 300–306.
- [11] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, "Feedback synthesis for underactuated systems using sequential second-order needle variations," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1826–1853, 2019.
- [12] J. Yu, M. Tan, S. Wang, and E. Chen, "Development of a biomimetic robotic fish and its control algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 4, pp. 1798–1810, 2006.
- [13] H. Deng, W. Wang, W. Luo, and G. Xie, "Yaw angle control of a boxfish-like robot based on cascade PID control algorithm," in *Proceedings of the International Conference on Power Electronics Systems and Applications*, Hong Kong, China, 2015, pp. 1–5.
- [14] P. Suebsaiprom and C.-L. Lin, "Maneuverability modeling and trajectory tracking for fish robot," *Control Engineering Practice*, vol. 45, pp. 22–36, 2015.
- [15] G. Mamakoukas, M. A. MacIver, and T. D. Murphey, "Sequential action control for models of underactuated underwater vehicles in a planar ideal fluid," in *Proceedings of the American Control Conference*, Boston, MA, USA, 2016, pp. 4500–4506.
- [16] N. Kato, "Control performance in the horizontal plane of a fish robot with mechanical pectoral fins," *IEEE Journal of Oceanic Engineering*, vol. 25, no. 1, pp. 121–129, 2000.
- [17] L. Wen, T. Wang, G. Wu, J. Liang, and C. Wang, "Novel method for the modeling and control investigation of efficient swimming for robotic fish," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3176–3188, 2011.
- [18] K. A. Morgansen, B. I. Triplett, and D. J. Klein, "Geometric methods for modeling and control of free-swimming fin-actuated underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1184–1199, 2007.
- [19] K. A. Morgansen, P. A. Vela, and J. W. Burdick, "Trajectory stabilization for a planar carangiform robot fish," in *Proceedings of the International Conference on Robotics and Automation*, Washington, DC, USA, 2002, pp. 756–762.
- [20] P. Suebsaiprom and C.-L. Lin, "Sliding mode path tracking control for fish-robot under ocean current perturbation," in *Proceedings of the International Conference on Control and Automation*, Kathmandu, Nepal, 2016, pp. 972–977.
- [21] M. Castaño and X. Tan, "Model predictive control-based path-following for tail-actuated robotic fish," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 7, pp. 1–11, 2019.

- [22] S. Zhang, B. Jiang, X. Chen, J. Liang, P. Cui, and X. Guo, “Modeling and Dynamic Control of a Class of Semibiomechanical Robotic Fish,” *Complexity*, vol. 2018, 2018.
- [23] M. L. Castaño and X. Tan, “Backstepping control-based trajectory tracking for tail-actuated robotic fish,” in *Proceedings of the International Conference on Advanced Intelligent Mechatronics*, Hong Kong, China, 2019, pp. 839–844.
- [24] S. Saimek and P. Y. Li, “Motion planning and control of a swimming machine,” *The International Journal of Robotics Research*, vol. 23, no. 1, pp. 27–53, 2004.
- [25] Q. Ren, J. Xu, and X. Li, “A motion control approach for a robotic fish with iterative feedback tuning,” in *Proceedings of the International Conference on Industrial Technology*, Seville, Spain, 2015, pp. 40–45.
- [26] A. Mauroy and J. Goncalves, “Linear identification of nonlinear systems: A lifting technique based on the Koopman operator,” in *Proceedings of the Conference on Decision and Control*, Las Vegas, NV, USA, 2016, pp. 6500–6505.
- [27] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, “A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition,” *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.
- [28] E. Kaiser, J. N. Kutz, and S. L. Brunton, “Data-driven approximations of dynamical systems operators for control,” *arXiv preprint arXiv:1902.10239*, 2019.
- [29] B. O. Koopman, “Hamiltonian systems and transformation in Hilbert space,” *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.
- [30] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, “Extending data-driven Koopman analysis to actuated systems,” in *Proceedings of the IFAC Symposium on Nonlinear Control Systems*, 2016, pp. 704–709.
- [31] J. L. Proctor, S. L. Brunton, and J. N. Kutz, “Generalizing Koopman theory to allow for inputs and control,” *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.
- [32] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, “Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control,” *PloS One*, vol. 11, no. 2, p. e0150171, 2016.
- [33] I. Mezić, “On applications of the spectral theory of the Koopman operator in dynamical systems and control theory,” in *Proceedings of the Conference on Decision and Control*, Osaka, Japan, 2015, pp. 7034–7041.
- [34] M. Budišić, R. Mohr, and I. Mezić, “Applied Koopmanism,” *Chaos*, vol. 22, no. 4, p. 047510, 2012.
- [35] M. Korda and I. Mezić, “Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control,” *Automatica*, vol. 93, pp. 149–160, 2018.
- [36] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, “On dynamic mode decomposition: Theory and applications,” *Journal of Computational Dynamics*, vol. 1, no. 2, pp. 391–421, 2014.
- [37] A. Mauroy and I. Mezić, “Global stability analysis using the eigenfunctions of the Koopman operator,” *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.
- [38] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the National Academy of Sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.
- [39] B. Lusch, J. N. Kutz, and S. L. Brunton, “Deep learning for universal linear embeddings of nonlinear dynamics,” *Nature Communications*, vol. 9, no. 1, p. 4950, 2018.
- [40] I. Abraham, G. De La Torre, and T. D. Murphey, “Model-based control using Koopman operators,” in *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, USA, 2017.
- [41] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan, “Modeling and control of soft robots using the Koopman operator and model predictive control,” in *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, 2019.
- [42] A. Salova, J. Emenheiser, A. Rupe, J. P. Crutchfield, and R. M. D’Souza, “Koopman operator and its approximations for systems with symmetries,” *Chaos*, vol. 29, no. 9, p. 093128, 2019.
- [43] A. S. Dogra and W. T. Redman, “Optimizing Neural Networks via Koopman Operator Theory,” in *Neural Information Processing Systems (NeurIPS)*, 2020.
- [44] S. Peitz and S. Klus, “Koopman operator-based model reduction for switched-system control of PDEs,” *Automatica*, vol. 106, pp. 184–191, 2019.
- [45] S. Sinha, U. Vaidya, and E. Yeung, “On computation of Koopman operator from sparse data,” in *Proceedings of the American Control Conference*, Philadelphia, PA, USA, 2019, pp. 5519–5524.
- [46] B. Huang, X. Ma, and U. Vaidya, “Data-driven nonlinear stabilization using koopman operator,” *The Koopman Operator in Systems and Control*, pp. 313–334, 2020.
- [47] E. Yeung, S. Kundu, and N. Hodas, “Learning deep neural network representations for Koopman operators of nonlinear dynamical systems,” in *Proceedings of the American Control Conference*, Philadelphia, PA, USA, 2019, pp. 4832–4839.
- [48] H. Lu and D. M. Tartakovsky, “Prediction accuracy of dynamic mode decomposition,” *SIAM Journal on Scientific Computing*, vol. 42, no. 3, pp. A1639–A1662, 2020.
- [49] G. Mamakoukas, M. L. Castaño, X. Tan, and T. D. Murphey, “Local Koopman operators for data-driven control of robotic systems,” in *Proceedings of Robotics: Science and Systems*, Freiburg, Germany, 2019.
- [50] P. J. Antsaklis and A. N. Michel, *A Linear Systems Primer*. Springer Science & Business Media, 2006, ch. 3, pp. 116–119.
- [51] 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.
- [52] E. Kaiser, J. N. Kutz, and S. L. Brunton, “Data-driven discovery of Koopman eigenfunctions for control,” *Machine Learning: Science and Technology*, 2021.
- [53] S. P. Nandanoori, S. Sinha, and E. Yeung, “Data-driven operator theoretic methods for global phase space learning,” in *2020 American Control Conference (ACC)*, Denver, CO, USA, 2020, pp. 4551–4557.
- [54] M. Haseli and J. Cortés, “Efficient identification of linear evolutions in nonlinear vector fields: Koopman Invariant Subspaces,” in *Conference on Decision and Control (CDC)*, Nice, France, 2019, pp. 1746–1751.
- [55] —, “Learning Koopman Eigenfunctions and Invariant Subspaces from Data: Symmetric Subspace Decomposition,” *arXiv preprint arXiv:1909.01419*, 2020.
- [56] N. Takeishi, Y. Kawahara, and T. Yairi, “Learning Koopman invariant subspaces for dynamic mode decomposition,” in *Proceedings of the Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1130–1140.
- [57] K. P. Murphy, *Machine learning: A Probabilistic Perspective*. Cambridge, MA: The MIT Press, 2012.
- [58] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York: Springer, 2013, vol. 112.
- [59] L. Ljung, *System Identification*. Upper Saddle River, NJ: Prentice Hall, 1998.
- [60] B. C. Daniels and I. Nemenman, “Automated adaptive inference of phenomenological dynamical models,” *Nature Communications*, vol. 6, p. 8133, 2015.
- [61] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos, “Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis,” *Communications in Mathematical Sciences*, vol. 1, no. 4, pp. 715–762, 2003.
- [62] Z. Hou and S. Jin, “A novel data-driven control approach for a class of discrete-time nonlinear systems,” *IEEE Transactions on Control Systems Technology*, vol. 19, no. 6, pp. 1549–1558, 2011.
- [63] J. R. Cloutier, “State-dependent Riccati equation techniques: an overview,” in *Proceedings of the 1997 American control conference (Cat. No. 97CH36041)*, vol. 2. Albuquerque, NM, USA: IEEE, 1997, pp. 932–936.
- [64] T. Çimen, “State-dependent Riccati equation (SDRE) control: A survey,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3761–3775, 2008.
- [65] T. Carleman, “Application de la théorie des équations intégrales linéaires aux systèmes d’équations différentielles non linéaires,” *Acta Mathematica*, vol. 59, no. 1, pp. 63–87, 1932.
- [66] K. Kowalski and W.-H. Steeb, *Nonlinear Dynamical Systems and Carleman Linearization*. Teaneck, NJ: World Scientific, 1991.
- [67] S. Banks, “Infinite-dimensional Carleman linearization, the Lie series and optimal control of non-linear partial differential equations,” *International Journal of Systems Science*, vol. 23, no. 5, pp. 663–675, 1992.
- [68] G. V. Bard, “Numerically estimating derivatives during simulations,” in *Proceedings of the International Conference on Modelling, Simulation, and Visualization Methods*, 2011, pp. 341–347.
- [69] P. J. Olver, *Introduction to partial differential equations*. Springer, 2014.
- [70] S. L. Brunton, J. L. Proctor, and J. N. Kutz, “Discovering governing equations from data by sparse identification of nonlinear dynamical systems,” *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

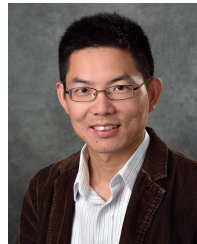
- [71] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*. Springer, 2001, pp. 547–577.
- [72] B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, "Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data," *Journal of Open Source Software*, vol. 5, no. 49, p. 2104, 2020. [Online]. Available: <https://doi.org/10.21105/joss.02104>
- [73] I. The MathWorks, *MATLAB Deep Learning Toolbox*, Natick, Massachusetts, United State, 2019b. [Online]. Available: <https://www.mathworks.com/help/deeplearning>
- [74] J. Wang and X. Tan, "Averaging tail-actuated robotic fish dynamics through force and moment scaling," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 906–917, 2015.
- [75] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear Koopman realizations for the modeling and control of systems with unknown dynamics," *arXiv preprint arXiv:2010.09961*, 2020.
- [76] G. Mamakoukas, O. Xherija, and T. D. Murphey, "Memory-Efficient Learning of Stable Linear Dynamical Systems for Prediction and Control," in *Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada, 2020.
- [77] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning data-driven stable Koopman operators," in <https://arxiv.org/abs/2005.04291>.
- [78] J. Quevedo, "Digital control: past, present and future of pid control," in *Proceedings of IFAC Workshop*, 2000.
- [79] F. Høgen, *PID Control*. Tapir Academic Press, 2004, ch. 4, pp. 94–98.
- [80] M. L. Castaño, A. Hess, G. Mamakoukas, T. Gao, T. Murphey, and X. Tan, "Control-oriented modeling of soft robotic swimmer with Koopman operators," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Boston, MA, USA, 2020, pp. 1679–1685.
- [81] D. A. Haggerty, M. J. Banks, P. C. Curtis, I. Mezić, and E. W. Hawkes, "Modeling, reduction, and control of a helically actuated inertial soft robotic arm via the Koopman operator," *arXiv preprint arXiv:2011.07939*, 2020.
- [82] H. Arbabi and I. Mezić, "Computation of transient Koopman spectrum using hankel-dynamic mode decomposition," in *APS Division of Fluid Dynamics Meeting Abstracts*, 2017, pp. G1–009.
- [83] H. Eivazi, L. Guastoni, P. Schlatter, H. Azizpour, and R. Vinuesa, "Recurrent neural networks and Koopman-based frameworks for temporal predictions in turbulence," *arXiv preprint arXiv:2005.02762*, 2020.
- [84] M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz, "Time-delay observables for Koopman: Theory and applications," *SIAM Journal on Applied Dynamical Systems*, vol. 19, no. 2, pp. 886–917, 2020.
- [85] M. Korda, M. Putinar, and I. Mezić, "Data-driven spectral analysis of the Koopman operator," *Applied and Computational Harmonic Analysis*, vol. 48, no. 2, pp. 599–629, 2020.
- [86] K. MacMillan and J. Sondow, "Proofs of power sum and binomial coefficient congruences via Pascal's identity," *The American Mathematical Monthly*, vol. 118, no. 6, pp. 549–551, 2011.



Giorgos Mamakoukas received the B.A. degree in Physics at Grinnell College, IA, USA in 2014 and the M.S. in Mechanical Engineering at Northwestern University, Evanston, IL, in 2017. He is currently pursuing the Ph.D. degree in Mechanical Engineering at Northwestern University under the supervision of Todd Murphey. His interests include control, robotics, optimization, and machine learning.



Maria L. Castaño received her B.S. in Electrical Engineering from Florida International University, Miami, FL, USA in May 2014. She is currently pursuing her Ph.D degree at Michigan State University, East Lansing, MI, USA under the supervision of Xiaobo Tan. Her research interest include modeling, controls, and underwater robotics.



Xiaobo Tan received the B.Eng. and M.Eng. degrees in automatic control from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 2002. He is currently an MSU Foundation Professor and the Richard M. Hong Endowed Chair in the Department of Electrical and Computer Engineering (ECE) at Michigan State University (MSU). His research interests include control systems, smart materials, underwater robotics, and soft robotics.

Dr. Tan is a Senior Editor for IEEE/ASME Transactions on Mechatronics. He has coauthored over 250 peer-reviewed journal and conference papers, and holds four US patents. He is a Fellow of IEEE and ASME, and a recipient of NSF CAREER Award (2006), MSU Teacher-Scholar Award (2010), MSU College of Engineering Withrow Distinguished Scholar Award (2018), Distinguished Alumni Award from the ECE Department at University of Maryland (2018), and multiple best paper awards.



Todd Murphey received his B.S. degree in mathematics from the University of Arizona and the Ph.D. degree in Control and Dynamical Systems from the California Institute of Technology. He is a Professor of Mechanical Engineering at Northwestern University. His laboratory is part of the Center for Robotics and Biosystems, and his research interests include robotics, control, computational methods for biomechanical systems, and computational neuroscience. Honors include the National Science Foundation CAREER award in 2006, membership in the

2014-2015 DARPA/IDA Defense Science Study Group, and Northwestern's Professorship of Teaching Excellence. He was a Senior Editor of the IEEE Transactions on Robotics.