

# Dynamic Shape Control of Soft Robots Enabled by Data-Driven Model Reduction

Iman Adibnazari<sup>1</sup>, Harsh Sharma<sup>2</sup>, Myungsun Park<sup>1</sup>, Jacobo Cervera-Torralba<sup>1</sup>,  
Boris Kramer<sup>1</sup>, Michael T. Tolley<sup>1,3</sup>

<sup>1</sup>Department of Mechanical and Aerospace Engineering, University of California San Diego, CA, USA

<sup>2</sup>Department of Mechanical Engineering, University of Wisconsin-Madison, WI, USA <sup>3</sup>Materials Science and Engineering Program, University of California San Diego, CA, USA

**Abstract**—Soft robots have shown immense promise in settings where they can leverage dynamic control of their entire bodies. However, effective dynamic shape control requires a controller that accounts for the robot’s high-dimensional dynamics—a challenge exacerbated by a lack of general-purpose tools for modeling soft robots amenable for control. In this work, we conduct a comparative study of data-driven model reduction techniques for generating linear models amenable to dynamic shape control. We focus on three methods—the eigensystem realization algorithm, dynamic mode decomposition with control, and the Lagrangian operator inference (LOpInf) method. Using each class of model, we explored their efficacy in model predictive control policies for the dynamic shape control of a simulated eel-inspired soft robot in three experiments: 1) tracking simulated reference trajectories guaranteed to be feasible, 2) tracking reference trajectories generated from a biological model of eel kinematics, and 3) tracking reference trajectories generated by a reduced-scale physical analog. In all experiments, the LOPInf-based policies generated lower tracking errors than policies based on other models.

## I. INTRODUCTION

Soft robotic systems have proven valuable in many settings, from enabling marine exploration [1] and novel medical technologies [2], to providing crucial insights into the biomechanics and ethology of biological organisms [3]. The benefits of soft robots often stem from the affordances provided by the compliance of the soft materials comprising them. For example, material compliance can be used to provide guarantees of mechanical safety in the event of collisions between a soft robot and delicate objects or environments (e.g., human tissue, coral reefs) [2], [4].

While the affordances provided by compliant materials and structures are central to the use of soft robotic systems, this compliance simultaneously poses key challenges to their development and broader adoption. One key challenge is a current lack of effective and general-purpose algorithmic tools for controlling soft systems [5]. General-purpose tools for controlling soft robots are challenging to develop because soft systems generally have highly underactuated continuum dynamics (i.e., the robot’s structure has infinite degrees of freedom and finite degrees of actuation) and suffer from myriad nonlinear effects, including material nonlinearities (e.g., hysteresis, viscoelastic effects) and geometric nonlinearities under loading.

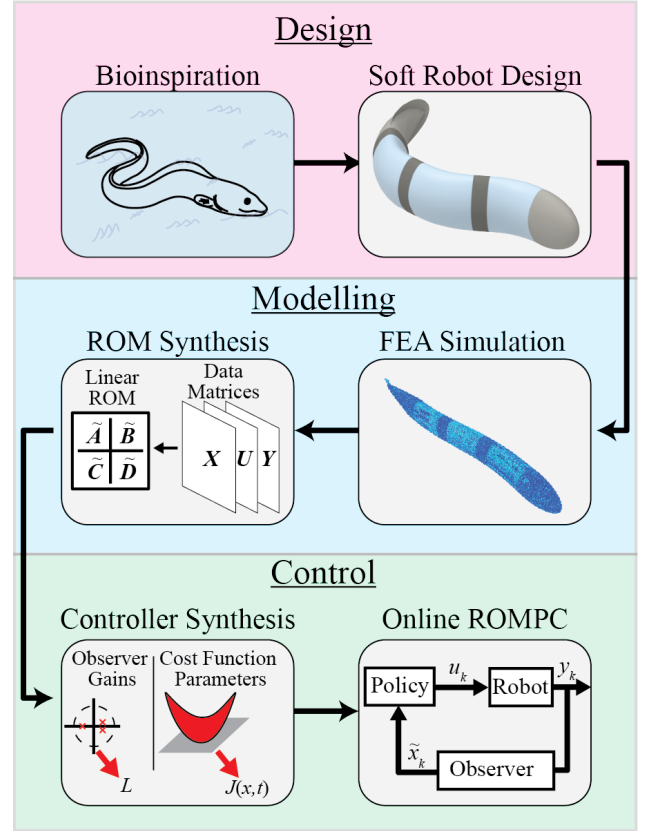


Fig. 1. Process flow for synthesizing a reduced-order model predictive controller for dynamic shape control of a soft robot. Our proposed process spans three stages—design (top), modeling (middle), and control (bottom)—that we defined to be broadly applicable to a variety of soft robots. In this work, we focus on an anguilliform-inspired soft robot, designed through a process of bioinspiration (top). Using data generated from a high-fidelity finite element simulation of the robot, we explored various techniques for data-driven model reduction to generate linear reduced-order models that are amenable to control. We then used these ROMs to construct a closed-loop state observer and reduced-order model predictive controller.

## A. Literature Review

Prior work on the dynamic control of soft robots has most frequently focused on the control of a single point of interest—typically the position of an end-effector [5]–[7]. However, there is increasing interest in dynamically controlling a soft robot’s full continuum structure in both manipulation and locomotion tasks [5], [6], [8]; snakes [9]–[11], fish [12],

[13], cephalopods [5], [14], and mammals with prehensile tails [15] frequently serve as inspiration for the benefits of dynamically controlled hybrid structures that combine rigid and soft materials. For these robots, dynamic shape control can be used to enable improved (e.g., more efficient) operation as well as richer interactions with their respective environments. These benefits sit in contrast to those provided by systems that use quasi-static shape control for locomotion tasks (e.g., quasi-static inchworm [16] and walking gaits [17] used to improve locomotion stability), and manipulation tasks in contexts where dynamic movement is undesirable (e.g., minimally invasive surgery [18]).

The dynamic shape control of a soft robot requires a control system that effectively accounts for the continuum mechanics governing the robot's behavior under finite degrees of actuation. However, past work on model-based control of soft robots has shown that the selection of a model frequently poses a tradeoff between model fidelity and model parsimony—generally affecting controller precision and speed [6], [19]. To address this challenge, four dominant paradigms have emerged for dynamic control of soft robotic systems: a) Model-based control with reduced physics models b) Reinforcement learning, c) Model-based control with numerical finite element simulations and intrusive reduced-order models thereof, and d) Linear control with models generated via extended dynamic mode decomposition methods. We will briefly survey these methods next.

*a) Model-based Control with Reduced Physics Models:*

The first of these paradigms emphasizes the use of reduced physics models in developing control systems for a given soft robot. Common examples of reduced physics models can be found in piecewise constant strain models of soft manipulators and models based on Cosserat rod theory [20]–[23]. Reduced physics models frequently assume quasi-static, discrete-state, and/or fully actuated behavior from a soft robot [5], [20], thereby diminishing, or even entirely removing, various dynamic phenomena that are otherwise realized by the physical system. In a control setting, the benefits of using reduced physics models lies in their interpretability, enabling the development of control systems with provable guarantees (e.g., stability, optimality, robustness) with respect to the underlying model [5]. For example, research on piecewise constant strain models has matured to the extent where these models admit many of the analytical and computational tools that have become common place for traditional, rigid-body systems [20], [22], [24]. However, the benefits of controllers based on reduced physics models are predicated on the accuracy of the assumptions underlying the model, which can degrade significantly as the complexity of the soft robot increases. Furthermore, constructing new reduced physics models requires significant expertise, is labor intensive, and results in a model that typically only applies to a single class of soft robot.

*b) Reinforcement Learning:* A second prominent paradigm for soft robot control circumvents the need for explicitly modeling the robot's dynamics by leveraging various reinforcement learning techniques [5], [11], [25], [26]. This approach has been employed to control soft manipulators [25], [27], soft swimming robots [26], [28], and

soft walking robots [29], often producing competent control policies that implicitly learn a model of the dynamics of the robot by training on data taken from an experimental or simulated system. However, the infinite dimensionality of soft robots implies requirements for large amounts of training data, especially as the number of inputs or outputs in the system increases [30], [31]. Moreover, control policies produced through this approach often provide few, if any, guarantees on stability, performance, and safety—factors crucial to operation in safety-critical environments like marine environments or around human bodies [30].

*c) Model-based Control with Numerical Finite Element Simulations and Intrusive Reduced Order Models:* Another paradigm for the control of soft robots focuses on developing model-based control policies around fast finite element simulations of a given system [19], [32]. A primary benefit of this approach lies in being broadly applicable to nearly arbitrary soft robots while also often providing accurate control of the system as a product of using high-fidelity models based on finite element methods (FEM) [33]. However, a fundamental challenge in using FEM-based models lies in their computational costs [32]. One approach to addressing this challenge has focused on developing real-time FEM simulation tools for the control the soft robots with sufficiently low-dimensional FEM models [34], [35]. For systems with higher-dimensional models however, various methods for projection-based model order reduction have been proposed to generate fast and low-dimensional numerical models suitable for use with linear control [35]–[39]. The techniques developed in these examples were shown to provide significant speed-ups to FEM simulations of soft robotic systems, enabling these models to be used in online control. Additionally, model order reduction and hyperreduction methods have also been shown to handle challenging settings where the robot might exhibit material and geometric nonlinearities as well as contacts with the environment [35], [39]. However, these methods are ultimately *intrusive* in the sense that they require the finite element simulator to provide access to the high-dimensional operators of the full-order model (FOM). This requirement poses challenges in settings where these operators are inaccessible and in systems that have limited computational resources, as is commonly the case for mobile robots.

*d) Linear Control with Models Generated via Extended Dynamic Mode Decomposition Methods::* The most recent paradigm to emerge for dynamic control of soft robots combines well-established tools for linear control with data-driven linear models generated through various extended dynamic mode decomposition (eDMD) methods [7], [40]–[42]. These eDMD-based models have been extensively explored and validated through the lens of Koopman operator theory [43], [44], wherein nonlinear lifting transformations are used to construct linear representations of otherwise nonlinear system dynamics. In practice, these transformations are applied to sensor data from a given robot, and the accuracy of learned models (and resulting quality of control) is often highly sensitive to one's selected library of lifting transformations [45], [46]. A common strategy for producing eDMD-based linear models involves generating a large library of lifting transformation

from a given family of basis functions (e.g., polynomials, time-delay functions). However, simply increasing the number of observables does not necessarily improve accuracy and the dimension of the learned model can scale quickly with the number of sensors used during operation and number of observable functions used for model synthesis [45].

When applied to the control of soft robots, a primary appeal of eDMD-based modeling and control lies in the ease of model synthesis directly from system data and the enforced linear structure of the resulting model. This linear structure enables the use of mature control tools such as linear quadratic regulator (LQR) control [6], and linear model predictive control (MPC) [7], [41]—wherein constraints on the input, output, and state of the system can be enforced during operation.

MPC frameworks are particularly well-suited for dynamic shape control of soft robots as they allow for explicit consideration of the robot’s dynamics, operating constraints, and external forcing from disturbances and interactions [47]. However, the manual selection of lifting transformations central to the control of soft robots with eDMD-based models often ignores the various physics and system-theoretic structure inherent to soft robot operation—structure that can be leveraged in a dynamic shape control setting. Alternative classes of *non-intrusive* data-driven model reduction techniques can enforce such underlying structure in fast and low-dimensional models synthesized from collected data. For example, some system identification techniques represent dynamics with guarantees on the controllability or observability of the resulting system [48], while others estimate models that preserve second-order mechanical structure of the system [49]–[51].

## B. Contributions

In this work, we explore the space of data-driven model reduction techniques to be used on model-predictive dynamic shape control for soft robots. We focus on the dynamic shape control of a simulated soft anguilliform robot (Fig. 1), motivated by the facts that anguilliform swimmers are among the most efficient natural swimmers, that their motion is highly dynamic in nature, and that their motion requires the careful coordination of the entire body of the fish to effectively produce thrust [52]–[54]. Therefore, a robot capable of similar motion requires effective dynamic shape control of its body.

Our contributions in this work are: 1) a generalizable pipeline for the modeling and dynamic shape control of soft robots enabled by data-driven linear model reduction, illustrated in Figure 1; 2) a comparative study of three non-intrusive model reduction methods, each assessed in estimation and control settings; 3) a high-fidelity, finite element structural simulation of a fluidically driven anguilliform soft robot to serve as a testbench for dynamic shape controllers; and 4) a freely available large-scale dataset of the dynamic behavior of the aforementioned simulated soft robot to facilitate further work on data-driven modeling of soft robotic systems [55]. Our simulated testbench only considers the structural mechanics of the robot under the forcing of pressure inputs that we assume propagate at much faster timescales than structural deformations. We made this choice because of the limited

available software tooling and immense computational expense related to simulating three-dimensional fluid-structure interaction (FSI) for soft robots. However, by focusing on structural mechanics, our comparative study addresses the challenges of dynamic shape control that are most relevant to the majority of soft robotic systems and accomplishes a necessary first step in effectively controlling soft robots that are subject to significant effects from fluid-structure interaction.

## C. Paper Outline

We begin in Section II by introducing the simulated anguilliform soft robot used as a testbench throughout this work followed by our formulation for reduced-order model predictive control (ROMPC) in Section III. We introduce the model reduction techniques used in our comparative study in Section IV and discuss their training and performance within an estimation setting in Section V. We then compare these methods in the context of dynamic shape control in Section VI on reference trajectories that are dynamically feasible. Following this comparison, we assess the performance of ROM-based controllers based on each model reduction method in two sets of control experiments wherein reference trajectories are generated from data collected from a physical system. In Section VII-A, we use our control scheme to track reference trajectories modeled after the gaits of anguilliform fish and in Section VII-B we track reference trajectories generated by a reduced-scale physical analog of our simulated platform. Finally, we discuss our results in Section VIII and present conclusions and potential avenues for future work in Section IX.

## II. ANGUILLIFORM SOFT ROBOT DESIGN AND SIMULATION

To center this work in a setting where dynamic shape control is necessary for successful operation, we constructed a large-scale, simulated testbench of the eel-inspired soft robot developed in [56] and [57] that models the dynamics of the robot using an open-source finite element simulation tool (SOFA, SOFA Consortium) [58], [59]. In this section, we briefly review the design and operation of the robot being simulated in Section II-A; we then describe our high-fidelity FEM simulation of its mechanics in Section II-B.

### A. Soft Robot Design and Operation

We considered the physical eel-inspired soft robot developed in [56] and [57]. Our simulated testbench models the structural mechanics of an increased-scale version of this soft robot, with dimensions  $1117 \text{ mm} \times 100 \text{ mm} \times 166.7 \text{ mm}$  ( $L \times W \times H$ )—three times the size of the physical system [56]. This robot is comprised of five soft segments—passive soft head and tail segments and three actuated body segments—all coupled via custom rigid clamps (Fig. 2.a). Each actuated body segment in the robot is driven by two antagonistic fluid elastomer actuators (FEAs) embedded inside of the body segment and separated by a thin, strain-limiting layer that is aligned with the sagittal plane of the robot [56]. Thus,

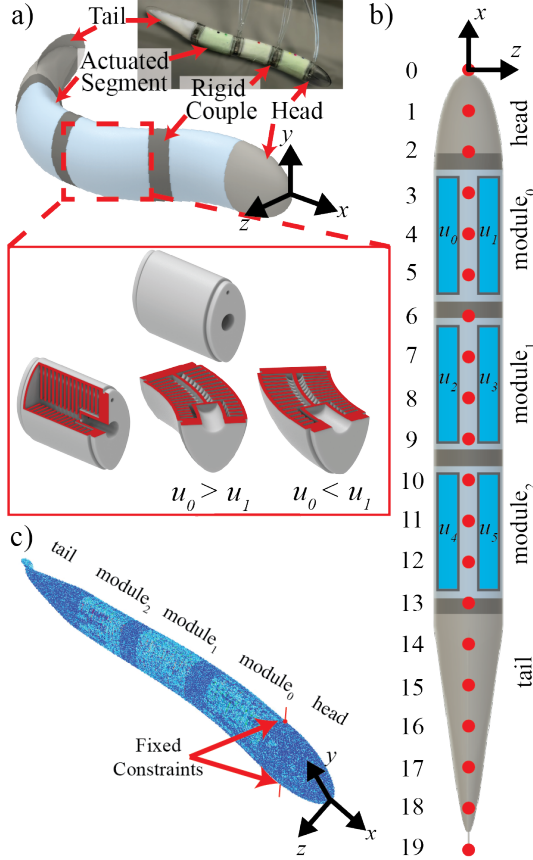


Fig. 2. Anguilliform soft robot design and simulation. (a) We demonstrate our proposed methods on a simulated version of an eel-inspired soft robot developed in [56], [57] (top right). This robot is comprised of soft, unactuated head and tail segments, three actuated segments driven by antagonistic fluid elastomer actuators, and rigid couples, used to securely connect the soft segments. As the two fluidic chambers in the actuated segments are pressurized with inputs,  $u_i$ , bending moments can be induced along the robot's body to affect its shape. (b) The robot accepts six pressure inputs that are physically coupled such that  $u_{2i} = -u_{2i+1}$  for  $i \in \{0, 1, 2\}$ . In our setting, we constructed the output of the robot as the  $x$ - $z$  position of 20 equally spaced points placed along the centerline of the robot's body, shown as red dots. (c) We simulated the system's dynamics through a custom finite element model constructed in an open source simulation framework (SOFA), with high-dimensional state  $\mathbf{x}(t) \in \mathbb{R}^n$  representing the spatial coordinates of each node in a high-dimensional mesh of the robot's geometry. We applied fixed constraints to two points at the base of the robot's head, allowing it to pivot about the line passing through these points (dorsal constrained point and line passing through both constrained points shown in red on the mesh of the robot). We selected this constraint based on past work on kinematics of anguilliform swimming indicating that this region of eels exhibits little lateral motion during forward swimming.

pressurizing the FEAs inside a given body segment induces a bending moment along the segment, and attaching multiple body segments together enables complex bending across the robot's body.

In [57], the antagonistic FEAs driving each body segment are pressurized by a gear pump in a closed hydraulic system. We approximate the behavior of this closed hydraulic drive system with a simulated constraint that when an FEA in a given body segment is actuated with pressure input,  $u$ , its antagonist FEA is actuated with the opposite pressure input,  $-u$ .

## B. Soft Robot Simulation in SOFA

Our numerical model builds on an open-source, finite element simulation framework (SOFA, SOFA Consortium) [58], [59]. To ensure mesh compatibility with the simulation, we generated a mesh of the system using an open-source meshing tool (Gmsh [60]) that enabled multi-resolution meshing across the robot. During meshing, we enforced a finer mesh resolution in regions with more intricate features, such as the internal chamber geometry of the FEAs, and coarser resolution elsewhere.

To avoid the computational expense and potential instability that can stem from resolving contact constraints, we meshed the robot as one contiguous body and defined regions of varying elastic moduli across the robot's body directly in SOFA. We used linear elasticity models across the entire system's structure, meaning that any nonlinear dynamic behavior would stem from geometric effects. We assigned three moduli to represent the three primary components of the robot: For regions containing rigid couples, we assigned an elastic modulus of 400 MPa to approximate rigid body behavior; for the strain-limiting layer of each body segment, we assigned an elastic modulus of 25 MPa, approximating that of the fiberglass composite (Garolite FR4) used in the physical system; and for the remainder of the robot, we assigned a modulus of 2 MPa—the 100% strain modulus of the silicone polymer used to construct the robot's body (Dragonskin 10, Smooth-On). Finally, we applied two, fixed Lagrangian constraints at nodes located at the dorsal and ventral extremes near the base of the robot's head, allowing the system to pivot about the line connecting these two constraints (Fig 2.c). Along with providing simulation stability, our placement of these boundary conditions was motivated by the fact that during straight-line locomotion, anguilliform swimmers tend to exhibit the least lateral motion in this region [52], [53].

The resulting simulation models the nonlinear full-order dynamics of the robot

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \quad (1)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \quad (2)$$

where at time  $t$ , the full-order state  $\mathbf{x}(t) \in \mathbb{R}^n$ , represents the spatial coordinates of every node in the meshed geometry of the robot, the control input,  $\mathbf{u}(t) \in \mathbb{R}^m$ , represents the pressure inputs to each of the fluidic chambers in the robot's actuators, and the output,  $\mathbf{y}(t) \in \mathbb{R}^p$  represents the  $x$ - $z$  coordinates of 20 equally spaced points placed along the dorsal centerline of the robot (see red dots in Fig. 2.b), computed by averaging the  $x$ - $z$  positions of the 20 mesh points nearest each control point's nominal position along the centerline. All spatial coordinates are expressed with respect to a stationary reference frame that is coincident with the anterior-most tip of the robot when in its neutral configuration (Fig. 2). The resulting full-order model of the robot's structural dynamics had state of dimension  $n = 243,789$ , output dimension  $p = 40$ , and input dimension  $m = 6$ , with the aforementioned antagonistic coupling constraints requiring only the selection of three decoupled control inputs at each timestep.

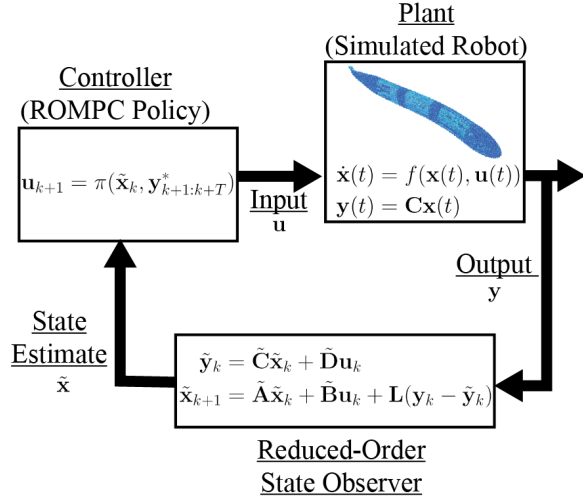


Fig. 3. Block diagram of proposed reduced-order model predictive control loop. For each tested ROM, we synthesized a closed-loop state observer that takes measurements of the simulated robot's centerline to produce estimates of the reduced-order state. These estimates and a desired reference trajectory are then used to seed an optimization-based control policy, which produces a trajectory of input pressures. At each timestep, the pressures at the first timestep of this trajectory are used as input to the robot.

To numerically evaluate this model, we used an implicit, first-order integration scheme (EulerImplicitSolver component in SOFA) and parallelized linear solver (ParallelCGLinearSolver component in SOFA's MultiThreading plugin) with a fixed timestep of  $\Delta t = 0.01$  seconds to maintain simulation stability. Due to the large size of the simulated robot's mesh, each simulation timestep required approximately 15 seconds of compute-time to evaluate on a custom desktop computer equipped with an Intel Core i9-10920X CPU, NVIDIA GeForce RTX 2080 Ti GPU, and 128.0 GiB of DDR4 memory, and running Ubuntu 22.04.5 LTS.

### III. DYNAMIC SHAPE CONTROL VIA REDUCED-ORDER MODEL PREDICTIVE CONTROL

Our numerical simulation provides a high-fidelity model of the anguilliform robot's structural dynamics. However, using this model in a real-time online control setting would be intractable, given the high-dimensionality of the full-order model.

To address this intractability, we propose a model-predictive control scheme that leverages non-intrusive ROMs produced from data generated by the full-order simulation, which we illustrate in Figure 3 [61], [62]. We focus on the use of linear ROMs as they are fast and enable the use of mature machinery for high-speed and constraint-aware online control [47].

Consider a discrete-time, linear ROM of the form

$$\begin{aligned}\tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}_k, \\ \tilde{\mathbf{y}}_{k+1} &= \tilde{\mathbf{C}}\tilde{\mathbf{x}}_{k+1} + \tilde{\mathbf{D}}\mathbf{u}_{k+1},\end{aligned}$$

with reduced-order state  $\tilde{\mathbf{x}}_k := \tilde{\mathbf{x}}(t_k) \in \mathbb{R}^r$  (with  $r \ll n$ ) at time  $t_k$ , estimate of the full-order output  $\tilde{\mathbf{y}}_k := \tilde{\mathbf{y}}(t_k) \in \mathbb{R}^p$ , and system matrices  $\tilde{\mathbf{A}} \in \mathbb{R}^{r \times r}$ ,  $\tilde{\mathbf{B}} \in \mathbb{R}^{r \times m}$ ,  $\tilde{\mathbf{C}} \in \mathbb{R}^{p \times r}$ ,

and  $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times m}$ . While the full-order simulation generates outputs as a function of only the full-order state, we include the matrix  $\tilde{\mathbf{D}}$  here to accommodate reduced-order modeling methods that produce a feedthrough term, such as the ERA method described in Section IV-D.

We use a Luenberger state observer that produces reduced-order state estimates to seed an optimization-based control policy (Fig. 3). At each timestep  $k$ , the centerline output  $\mathbf{y}_k := \mathbf{y}(t_k)$  from the nonlinear FOM (1)-(2) is used to update the observer's reduced-order state estimate  $\tilde{\mathbf{x}}_k$  under the estimation law

$$\tilde{\mathbf{y}}_k = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{D}}\mathbf{u}_k \quad (3a)$$

$$\tilde{\mathbf{x}}_k = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_{k-1} + \tilde{\mathbf{B}}\mathbf{u}_{k-1} + \mathbf{L}(\mathbf{y}_k - \tilde{\mathbf{y}}_k) \quad (3b)$$

The updated state estimate produced in (3b) is then used as the initial condition of the policy  $\pi(\tilde{\mathbf{x}}_k, \mathbf{y}_{k+1:k+T}^*)$ , that, given the reference trajectory  $\mathbf{y}_{k+1:k+T}^*$ , over control horizon  $T$ , computes an optimal control sequence  $\mathbf{u}_{k+1:k+T}^*$ , by solving the quadratic program at every timestep  $k$ ,

$$\begin{aligned}\mathbf{u}_{k+1:k+T}^* = \arg \min_{\mathbf{u}_{k+1:k+T}} \sum_{i=k+1}^{k+T} & \left( \|\tilde{\mathbf{y}}_i - \mathbf{y}_i^*\|_{\mathbf{W}_y}^2 + \|\mathbf{u}_i\|_{\mathbf{W}_u}^2 \right. \\ & \left. + \|\mathbf{u}_i - \mathbf{u}_{i-1}\|_{\mathbf{W}_{\delta u}}^2 \right) \quad (4a)\end{aligned}$$

s.t.

$$\tilde{\mathbf{x}}_{-1} = \tilde{\mathbf{x}}_k \quad (4b)$$

$$\mathbf{u}_{-1} = \mathbf{u}_k \quad (4c)$$

$$\tilde{\mathbf{x}}_i = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_{i-1} + \tilde{\mathbf{B}}\mathbf{u}_{i-1}, \quad (4d)$$

$$\tilde{\mathbf{y}}_i = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_i + \tilde{\mathbf{D}}\mathbf{u}_i, \quad (4e)$$

$$\|\mathbf{u}_i\|_{\infty} < u_{\max}, \quad (4f)$$

$$\mathbf{u}_{i,2j} = -\mathbf{u}_{i,2j+1} \quad \forall j \in \{0, 1, 2\} \quad (4g)$$

where  $\|\cdot\|_{\mathbf{W}_*}$  indicates the weighted 2-norm of a vector with diagonal weighting matrix  $\mathbf{W}_*$ . The objective of this optimization is to determine the sequence of inputs that simultaneously minimizes the weighted magnitudes of the tracking error  $\|\tilde{\mathbf{y}}_i - \mathbf{y}_i^*\|_{\mathbf{W}_y}^2$ , the weighted control energy  $\|\mathbf{u}_i\|_{\mathbf{W}_u}^2$ , and the discrete rate of change of the control input  $\|\mathbf{u}_i - \mathbf{u}_{i-1}\|_{\mathbf{W}_{\delta u}}^2$ . The constraints in (4b) and (4c) are used to set the initial state and input for the optimization, where the initial state constraint uses the estimate in (3b). These initial conditions are then rolled out over the control horizon via constraint (4d), with outputs evaluated at each timestep through constraint (4e). Finally, the constraint in (4f) is used to set a maximum possible pressure value for control and the constraint in (4g) is used to account for the antagonistic coupling of pressures present in the robot's closed fluidic drive system. After computing the optimizing input sequence  $\mathbf{u}_{k+1:k+T}^*$ , from this constrained optimization, we apply the first input in the sequence,  $\mathbf{u}_{k+1}^*$ , to the simulated robot, compute the system's response to the optimized control input, and generate the simulated systems state at the next timestep. We continue iterating this process until the end of the simulation horizon.

### IV. DATA-DRIVEN REDUCED-ORDER MODELING

For the controlled simulated system to effectively track full-body reference trajectories, the linear ROM used for state es-

timization and control optimization must effectively capture the controlled dynamics of the robot. Thus, we compared multiple methods for data-driven, linear model reduction with control-affine inputs with the goal of assessing their performance in a dynamic, full-body control setting.

In our comparative study, we considered three methods representing different classes of data-driven model reduction/system identification—the Lagrangian Operator Inference method (LOpInf) [49], dynamic mode decomposition with control (DMDc) [63], and a combination of the Eigensystem realization algorithm (ERA) with the observer Kalman filter identification algorithm (OKID) [48], [64]. In this section, we describe each method as used in our setting, beginning with details how we preprocessed data for model synthesis in Section IV-A, and then introducing formulations for the LOpInf, DMDc, and ERA methods in Section IV-B, Section IV-C, and Section IV-D respectively.

#### A. Data Preprocessing

We assume access to the full internal state,  $\mathbf{x}_k \in \mathbb{R}^n$ , control inputs that generated that state  $\mathbf{u}_k \in \mathbb{R}^m$ , and resulting output  $\mathbf{y}_k \in \mathbb{R}^p$ , where the subscript  $k$  indicates the value of a variable at time  $t_k$ . Since our main goal was to generate data-driven ROMs for controlling the simulated soft robot, we preprocessed the full-order states and outputs of all simulation data to compute centered data: to compute centered full-order state and output trajectories  $\mathbf{x}_{\text{centered},k} = \mathbf{x}_k - \mathbf{x}_{\text{neutral}}$  and  $\mathbf{y}_{\text{centered},k} = \mathbf{y}_k - \mathbf{y}_{\text{neutral}}$ , respectively. These centered variables are shifted such that nonzero coordinates of the neutral state  $\mathbf{x}_{\text{neutral}} \in \mathbb{R}^n$  and output  $\mathbf{y}_{\text{neutral}} \in \mathbb{R}^p$  of the robot map to the origin of the full-order state and output spaces, respectively. This centering of state and output trajectories was motivated by the system-theoretic insight that the only potential stable point of a linear system subject to zero input and zero disturbance is at its origin, and that this should coincide with the stable configuration of the robot when subject to zero input and zero disturbance. We synthesized all system models using these centered variables, un-centering them only for the sake of mesh visualization. From hereon, we simplify the notation for the centered variables: the centered state trajectory  $\mathbf{x}_{\text{centered},k}$  and the centered output trajectory  $\mathbf{y}_{\text{centered},k}$  at time  $t_k$  are denoted as  $\mathbf{x}_k$  and  $\mathbf{y}_k$ , respectively.

#### B. Lagrangian Operator Inference Method

The Lagrangian Operator Inference (LOpInf) method is a non-intrusive model reduction method that first projects high-dimensional snapshot data onto a low-dimensional subspace, and then fits linear reduced operators that preserve an assumed Lagrangian structure of the system's dynamics [49]. To learn a ROM with this method, we first construct snapshot matrices of the full-order state, control input, and output, as

$$\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_K] \in \mathbb{R}^{n \times K}, \quad (5)$$

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K] \in \mathbb{R}^{m \times K}, \quad (6)$$

$$\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_K] \in \mathbb{R}^{p \times K}, \quad (7)$$

where  $K$  is the number of snapshots.

This data is very high-dimensional ( $n = 243,789$ ), and we cannot learn from it directly. Thus, to project the high-dimensional snapshot data onto a low-dimensional subspace, we first construct a basis matrix  $\mathbf{U}_r$  via truncated singular value decomposition of  $\mathbf{X}$  with truncation value  $r$  as

$$\mathbf{X} \approx \mathbf{U}_r \mathbf{\Sigma}_r \mathbf{V}_r^*, \quad (8)$$

where  $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ ,  $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$ , and  $\mathbf{V}_r^* \in \mathbb{R}^{r \times K}$ .

We then construct a reduced snapshot matrix by projecting the full-order snapshot data onto the reduced basis as

$$\hat{\mathbf{X}} = \mathbf{U}_r^T \mathbf{X} = [\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_K] \in \mathbb{R}^{r \times K}, \quad (9)$$

where  $\hat{\mathbf{x}}_k := \mathbf{U}_r^T \mathbf{x}_k$  is the reduced state trajectory at time  $t_k$ . We also build snapshot matrices of the reduced first-order and second-order time-derivative data as

$$\dot{\hat{\mathbf{X}}} = [\dot{\hat{\mathbf{x}}}_1, \dots, \dot{\hat{\mathbf{x}}}_K] \in \mathbb{R}^{r \times K}, \quad \ddot{\hat{\mathbf{X}}} = [\ddot{\hat{\mathbf{x}}}_1, \dots, \ddot{\hat{\mathbf{x}}}_K] \in \mathbb{R}^{r \times K},$$

where  $\dot{\hat{\mathbf{x}}}_k$  and  $\ddot{\hat{\mathbf{x}}}_k$  are obtained from the reduced state trajectories via numerical approximation, here with an eighth-order central finite difference scheme. Given the snapshot data matrices of the state and its derivatives, we postulate a Lagrangian formulation for the ROM:

$$\ddot{\hat{\mathbf{x}}}(t) + \hat{\mathbf{D}}\dot{\hat{\mathbf{x}}}(t) + \hat{\mathbf{K}}\hat{\mathbf{x}}(t) = \hat{\mathbf{B}}\mathbf{u}(t), \quad (10)$$

$$\mathbf{y}(t) = \hat{\mathbf{C}}\hat{\mathbf{x}}(t), \quad (11)$$

with the reduced damping matrix  $\hat{\mathbf{D}} \in \mathbb{R}^{r \times r}$ , the reduced stiffness matrix  $\hat{\mathbf{K}} \in \mathbb{R}^{r \times r}$ , the reduced input matrix  $\hat{\mathbf{B}} \in \mathbb{R}^{r \times m}$ , and the reduced output matrix  $\hat{\mathbf{C}} \in \mathbb{R}^{p \times r}$ .

Given the reduced snapshot data  $\hat{\mathbf{X}}$  and the reduced time-derivative data  $\dot{\hat{\mathbf{X}}}$  and  $\ddot{\hat{\mathbf{X}}}$ , we infer the Lagrangian ROM operators in (10) by solving

$$\min_{\hat{\mathbf{K}}=\hat{\mathbf{K}}^T \succ 0, \hat{\mathbf{D}}=\hat{\mathbf{D}}^T \succ 0, \hat{\mathbf{B}}} \|\ddot{\hat{\mathbf{X}}} + \hat{\mathbf{D}}\dot{\hat{\mathbf{X}}} + \hat{\mathbf{K}}\hat{\mathbf{X}} - \hat{\mathbf{B}}\mathbf{U}\|_F. \quad (12)$$

We infer the reduced output operator  $\hat{\mathbf{C}}$  by solving the least-squares problem

$$\min_{\hat{\mathbf{C}}} \|\mathbf{Y} - \hat{\mathbf{C}}\hat{\mathbf{X}}\|_F. \quad (13)$$

The symmetric positive-definite constraints on  $\hat{\mathbf{K}}$  and  $\hat{\mathbf{D}}$  in (12) ensure that the linear ROM (10) preserves the underlying Lagrangian structure (see [49] for details).

The resulting ROM defines second-order, continuous-time dynamics that, for the sake of consistency in our MPC formulation, we recast as a system of first-order, discrete-time difference equations. To do this, we first define an augmented reduced state,  $\tilde{\mathbf{x}}^{(e)} = [\hat{\mathbf{x}}^T, \dot{\hat{\mathbf{x}}}^T]^T$ , with equivalent continuous-time dynamics

$$\dot{\tilde{\mathbf{x}}}^{(e)}(t) = \hat{\mathbf{A}}\tilde{\mathbf{x}}^{(e)}(t) + \hat{\mathbf{B}}\mathbf{u}(t), \quad (14)$$

$$\mathbf{y}(t) = \hat{\mathbf{C}}\tilde{\mathbf{x}}^{(e)}(t), \quad (15)$$

where, with a slight abuse of notation, we have

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{0}_{r \times r} & \mathbf{I}_r \\ -\hat{\mathbf{K}} & -\hat{\mathbf{D}} \end{bmatrix}, \quad \hat{\mathbf{B}} = \begin{bmatrix} \mathbf{0}_{r \times m} \\ \hat{\mathbf{B}} \end{bmatrix}, \quad \hat{\mathbf{C}} = \begin{bmatrix} \hat{\mathbf{C}} & \mathbf{0}_{p \times r} \end{bmatrix}.$$

Assuming zero-order hold conditions on the input,  $\mathbf{u}_k$ , and sampling time  $T_s = \Delta t = 0.01$  seconds, we derive an equivalent discrete-time system [65] as

$$\tilde{\mathbf{x}}_{k+1}^{(e)} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k^{(e)} + \tilde{\mathbf{B}}\mathbf{u}_k, \quad (16)$$

$$\tilde{\mathbf{y}}_k = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_k^{(e)}, \quad (17)$$

with  $\tilde{\mathbf{A}} = e^{\tilde{\mathbf{A}}T_s}$ ,  $\tilde{\mathbf{B}} = \tilde{\mathbf{A}}^{-1}(\tilde{\mathbf{A}} - \mathbf{I}_r)\tilde{\mathbf{B}}$ , and  $\tilde{\mathbf{C}} = \tilde{\mathbf{C}}$ .

### C. Dynamic mode decomposition with control

Dynamic mode decomposition with control (DMDc) [63] is a non-intrusive model reduction method that learns linear ROMs of high-dimensional nonautonomous systems purely from data. Similar to DMD [66], [67], we first build a snapshot data matrix  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_{K-1}] \in \mathbb{R}^{n \times K-1}$ , and a time-shifted snapshot data matrix  $\mathbf{X}' = [\mathbf{x}_2, \dots, \mathbf{x}_K] \in \mathbb{R}^{n \times K-1}$ . We also build the corresponding input snapshot matrix,  $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_{K-1}] \in \mathbb{R}^{m \times K-1}$ , and stack data matrices  $\mathbf{X}$  and  $\mathbf{U}$  to construct  $\tilde{\mathbf{\Omega}} := \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \end{bmatrix} \in \mathbb{R}^{(n+m) \times K-1}$ .

Given data matrices  $\tilde{\mathbf{\Omega}}$  and  $\mathbf{X}'$ , we compute the truncated singular value decompositions of  $\tilde{\mathbf{\Omega}}$  and  $\mathbf{X}'$  with truncation values of  $q$  and  $r$ , respectively, resulting in the decompositions

$$\tilde{\mathbf{\Omega}} \approx \begin{bmatrix} \tilde{\mathbf{u}}_1 \\ \tilde{\mathbf{u}}_2 \end{bmatrix} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{v}}^* \quad \mathbf{X}' \approx \tilde{\mathbf{u}} \tilde{\mathbf{\Sigma}} \tilde{\mathbf{v}}^*, \quad (18)$$

with  $\tilde{\mathbf{u}}_1 \in \mathbb{R}^{n \times q}$ ,  $\tilde{\mathbf{u}}_2 \in \mathbb{R}^{m \times q}$ ,  $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{q \times q}$ ,  $\tilde{\mathbf{v}}^* \in \mathbb{R}^{q \times K-1}$ ,  $\tilde{\mathbf{u}} \in \mathbb{R}^{n \times r}$ ,  $\tilde{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ , and  $\tilde{\mathbf{v}}^* \in \mathbb{R}^{r \times K-1}$ . Using the full-order state approximation  $\mathbf{x} \approx \tilde{\mathbf{u}} \hat{\mathbf{x}}$ , the resulting DMDc linear ROM is

$$\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}_k, \quad (19)$$

$$\tilde{\mathbf{y}}_{k+1} = \tilde{\mathbf{C}}\tilde{\mathbf{x}}_k, \quad (20)$$

with the reduced state matrix  $\tilde{\mathbf{A}} := \tilde{\mathbf{u}}^* \mathbf{X}' \tilde{\mathbf{v}} \tilde{\mathbf{\Sigma}}^{-1} \tilde{\mathbf{u}}_1^* \tilde{\mathbf{u}} \in \mathbb{R}^{r \times r}$ , the reduced input matrix  $\tilde{\mathbf{B}} := \tilde{\mathbf{u}}^* \mathbf{X}' \tilde{\mathbf{v}} \tilde{\mathbf{\Sigma}}^{-1} \tilde{\mathbf{u}}_2^* \in \mathbb{R}^{m \times r}$ , and the reduced output matrix  $\tilde{\mathbf{C}} := \tilde{\mathbf{C}} \tilde{\mathbf{u}}$  where

$$\tilde{\mathbf{C}} = \min_{\tilde{\mathbf{C}}} \|\mathbf{Y} - \tilde{\mathbf{C}} \mathbf{X}\|_F, \quad (21)$$

where  $\mathbf{Y}$  is as defined in (7).

### D. Eigensystem Realization Algorithm

The Eigensystem Realization Algorithm (ERA) [48] is a system identification method for identifying low-dimensional linear state-space models from impulse response data. For arbitrary input-output data, the ERA approach is used in combination with the Observer Kalman Filter Identification (OKID) algorithm [64], which estimates impulse response data from arbitrary input-output data. To employ this method, we first construct the output data matrix  $\mathbf{Y}$ , as defined in (7), as well as the input data matrix

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \cdots & \mathbf{u}_K \\ 0 & \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{K-1} \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & \mathbf{u}_1 \end{bmatrix} \in \mathbb{R}^{mK \times K} \quad (22)$$

using the input and output data collected from a single experiment. The input-output relationship of a linear system with arbitrary input can be expressed as

$$\mathbf{Y} = \mathbf{M}\mathbf{U}, \quad (23)$$

where

$$\mathbf{M} = [\mathbf{D} \quad \mathbf{CB} \quad \mathbf{CAB} \quad \cdots \quad \mathbf{CA}^{K-1}\mathbf{B}] \quad (24)$$

is a matrix of Markov parameters for the system, written in terms of the unknown  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$  and input-output feedthrough matrix,  $\mathbf{D}$ . Recognizing this relationship, we approximate  $\mathbf{M}$  as

$$\hat{\mathbf{M}} = \mathbf{Y}\mathbf{U}^\dagger, \quad (25)$$

where  $\mathbf{U}^\dagger$  denotes the Moore-Penrose pseudo-inverse of the input matrix,  $\mathbf{U}$  [68]. Using the estimated Markov parameters in  $\hat{\mathbf{M}}$ , we then construct two Hankel matrices

$$\mathbf{H} = \begin{bmatrix} \mathbf{CB} & \mathbf{CAB} & \cdots & \mathbf{CA}^{T_H-1}\mathbf{B} \\ \mathbf{CAB} & \mathbf{CA}^2\mathbf{B} & \cdots & \mathbf{CA}^{T_H}\mathbf{B} \\ \vdots & & \ddots & \\ \mathbf{CA}^{T_H-1}\mathbf{B} & \mathbf{CA}^{T_H}\mathbf{B} & \cdots & \mathbf{CA}^{2T_H-1}\mathbf{B} \end{bmatrix} \quad (26)$$

and

$$\mathbf{H}' = \begin{bmatrix} \mathbf{CAB} & \mathbf{CA}^2\mathbf{B} & \cdots & \mathbf{CA}^{T_H}\mathbf{B} \\ \mathbf{CA}^2\mathbf{B} & \mathbf{CA}^3\mathbf{B} & \cdots & \mathbf{CA}^{T_H+1}\mathbf{B} \\ \vdots & & \ddots & \\ \mathbf{CA}^{T_H}\mathbf{B} & \mathbf{CA}^{T_H+1}\mathbf{B} & \cdots & \mathbf{CA}^{2T_H}\mathbf{B} \end{bmatrix}, \quad (27)$$

where  $T_H = \lfloor K/2 \rfloor - 1$ . After taking an order- $r$  truncated singular value decomposition of  $\mathbf{H}$ ,

$$\mathbf{H} \approx \tilde{\mathbf{u}}_r \tilde{\mathbf{\Sigma}}_r \tilde{\mathbf{v}}_r^\top, \quad (28)$$

we estimate the order- $r$  system matrices as

$$\tilde{\mathbf{A}} = \tilde{\mathbf{\Sigma}}_r^{-1/2} \tilde{\mathbf{u}}_r^\top \mathbf{H}' \tilde{\mathbf{v}}_r \tilde{\mathbf{\Sigma}}_r^{-1/2}, \quad (29)$$

$$\tilde{\mathbf{B}} = \tilde{\mathbf{\Sigma}}_r^{1/2} \tilde{\mathbf{v}}_r^\top \mathbf{E}_l, \quad (30)$$

$$\tilde{\mathbf{C}} = \mathbf{E}_m^\top \tilde{\mathbf{u}}_r \tilde{\mathbf{\Sigma}}_r^{1/2}, \quad (31)$$

and  $\tilde{\mathbf{D}}$  is taken directly from the first block entry of the Markov parameter matrix  $\mathbf{M}$ . The matrices  $\mathbf{E}_l$  and  $\mathbf{E}_m$  are selection matrices constructed as

$$\mathbf{E}_l = [\mathbf{I}_l, \mathbf{0}_{l \times (l(T_H-1))}]^\top, \quad \mathbf{E}_m = [\mathbf{I}_m, \mathbf{0}_{m \times (m(T_H-1))}]^\top.$$

The result of this procedure is a linear, order- $r$  ROM of the form

$$\begin{aligned} \tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}_k, \\ \tilde{\mathbf{y}}_{k+1} &= \tilde{\mathbf{C}}\tilde{\mathbf{x}}_{k+1} + \tilde{\mathbf{D}}\mathbf{u}_{k+1}, \end{aligned}$$

with reduced state matrix  $\tilde{\mathbf{A}} \in \mathbb{R}^{r \times r}$ , reduced input matrix  $\tilde{\mathbf{B}} \in \mathbb{R}^{r \times m}$ ,  $\tilde{\mathbf{C}} \in \mathbb{R}^{p \times r}$ , and feedforward matrix  $\tilde{\mathbf{D}} \in \mathbb{R}^{p \times m}$ . Unlike LOPInf and DMDc, the ERA/OKID approach is based entirely on the input-output data and does not require a snapshot data matrix containing full-order state. This can be beneficial in settings where full-order state data is not available.

## V. ROM TRAINING AND ESTIMATION ACCURACY

Using the methods presented in the previous section, we generated a total of 70 ROMs and assessed their predictive accuracy as a function of their dimension and the amount of data used for training. In this section we detail this procedure, beginning with how we generated a large-scale dataset of the simulated robot's dynamics in Section V-A, then in Section V-B defining how we generated ROMs from each method with varying dimensions and amounts of training data. After, we introduce the estimation laws that we used to assess the predictive accuracy of each ROM in Section V-C, and characterize the predictive performance of each ROMs under these estimation laws in Section V-D.

### A. Dataset Generation

We constructed a dataset of  $N = 40$  simulated trials of the simulated soft robot, with each trial comprised of  $T = 1000$  timesteps, resulting in full-order state, input, and output trajectories that each span 10 seconds. For all trials, we prescribed sinusoidal pressure inputs to the channels in the soft actuators shown in Fig. 2 so as to excite cyclic behavior in the robot's dynamics. These inputs are defined as

$$u_0(t) = A_1 \sin(2\pi ft + \phi_0), \quad u_1(t) = -u_0(t), \quad (32a)$$

$$u_2(t) = A_2 \sin(2\pi ft + \phi_1), \quad u_3(t) = -u_2(t), \quad (32b)$$

$$u_4(t) = A_3 \sin(2\pi ft + \phi_2), \quad u_5(t) = -u_4(t), \quad (32c)$$

where  $A_1, A_2$ , and  $A_3$  denote the amplitudes of the sinusoidal pressure inputs to each segments,  $f$ , is the frequency of oscillation, and  $\phi_1 = 0^\circ$ ,  $\phi_2 = 120^\circ$ , and  $\phi_3 = 240^\circ$  are the relative phases between inputs to each segment. As mentioned above,  $u_{2i} = -u_{2i+1}, i \in \{0, 1, 2\}$  to approximate the physical constraints imposed by the closed fluidic system used to drive each actuated body segment antagonistically [57].

To excite a variety of dynamic behaviors, we simulated the anguilliform robot over  $N = 40$  trials using five different frequencies and eight combinations of amplitudes. We chose frequencies that spanned from an approximately quasi-static regime ( $f = 0.1$  Hz) to frequencies seen in anguilliform animals during dynamic swimming ( $f = 1.5$  Hz) [53]. Additionally, we chose amplitudes with the goal of exciting varying degrees of geometric nonlinearity. For this, we selected both low amplitude,  $A_L$ , and high amplitude,  $A_H$ , inputs to apply to active segments in a trial. We selected  $A_H$  such that the steady-state bending it induces in a given actuated segment is similar to that demonstrated by the physical platform in [57]. A full description of actuation parameters for generating the dataset is shown in Table I.

During each trial, we collected 1) the inputs signals applied to the robot; 2) the full-order state of the robot; and 3) the output of the system, defined by the  $x - z$  position of 20 equally spaced points along the dorsal centerline of the robot's body.

### B. ROM Training

Once we select the ROM model class, the fidelity of the data-driven ROMs generated by the ERA, DMDc, and LOPInf

TABLE I  
OPEN-LOOP ACTUATION PARAMETERS FOR DATASET GENERATION.  
TRIALS RESERVED FOR MODEL TRAINING ARE MARKED IN GREEN AND  
TRIALS USED IN CONTROL EXPERIMENTS ARE MARKED WITH \*.

|                                |                   | Actuation Frequency, $f$ [Hz] |     |     |     |     |
|--------------------------------|-------------------|-------------------------------|-----|-----|-----|-----|
|                                |                   | 0.1                           | 0.3 | 0.5 | 1   | 1.5 |
| Amplitudes ( $A_1, A_2, A_3$ ) | $(A_L, 0, 0)$     | 1                             | 2   | 3   | 4*  | 5   |
|                                | $(0, A_L, 0)$     | 6*                            | 7   | 8   | 9   | 10  |
|                                | $(0, 0, A_L)$     | 11                            | 12* | 13  | 14  | 15  |
|                                | $(A_L, A_L, A_L)$ | 16                            | 17  | 18  | 19* | 20  |
|                                | $(A_H, 0, 0)$     | 21                            | 22* | 23  | 24  | 25  |
|                                | $(0, A_H, 0)$     | 26*                           | 27  | 28  | 29  | 30  |
|                                | $(0, 0, A_H)$     | 31                            | 32  | 33* | 34  | 35  |
|                                | $(A_H, A_H, A_H)$ | 36                            | 37  | 38  | 39* | 40  |

methods are defined by two factors: 1) the amount and quality of data used during training, and 2) the dimension  $r$ , of the ROM being synthesized. To understand the effects of these factors on the estimation and control of a soft robot, we used each method to synthesize ROMs of dimension  $r = 2, 4, \dots, 20$  on datasets that included one, two, and three trials of simulated data. Due to the high dimensionality of the full-order system data, three trials proved to be the limit for local training on the custom desktop computer described in Section II-B and the use of an off-the-shelf parallel computation library (Dask, Dask Core). Due to the formulation of the ERA method, models could only be synthesized on one trial of timeseries data. Thus, our experimental study considered 70 different ROMs—30 each generated with DMDc and LOPInf, and 10 generated with ERA.

We reserved three trials for training each ROM (highlighted in green in Table I) based on the heuristic that the input parameters for these trials would produce a diverse set of dynamic and quasi-static behaviors that could then be represented by the resulting ROMs. We used the data generated by Trial 40 for all ROMs trained on a single trial of data, assuming the high-frequency and high-amplitude actuation would excite the richest set of structural dynamic behaviors—all ERA-based ROMs were trained on only this trial. We used Trials 40 and 17 for all ROMs trained on two trials of data, and additionally included Trial 36 for all ROMs trained on 3 trials of data. These trials provided additional information on lower-frequency dynamics with varying amplitudes. As geometric effects were the primary source of nonlinearity in the simulated system, we included trials with varied input amplitude to introduce varying amounts of nonlinearity into training.

### C. Open-loop and Closed-loop Estimation Laws

Our proposed ROMPC scheme (Fig. 3) leverages ROMs in both open-loop and closed-loop estimation settings described in [65], wherein “open-loop” estimation refers to using a given initial reduced-order state and input sequence with the system

matrices of a given ROM to estimate the reduced-order state and output trajectories of the simulated system. In contrast, “closed-loop” estimation refers to additionally using output measurements at each timestep to correct the state estimates produced by the state estimator. As our optimization-based control policy requires rolling out trajectories to account for the constraints in (4), it is necessary for a given ROM to provide accurate open-loop predictions of the output trajectories over the time horizon of control. These roll outs must also be seeded with accurate estimates of the reduced-order state at a given timestep, provided by a closed-loop state observer [65].

To assess the open-loop prediction accuracy of each ROM, we generated output estimates  $\tilde{\mathbf{y}}_k$  using a chosen timeseries of inputs  $\mathbf{u}_k$ , and initial condition  $\tilde{\mathbf{x}}_0 = \mathbf{0}$  with the estimation law

$$\begin{aligned}\tilde{\mathbf{x}}_{k+1} &= \tilde{\mathbf{A}}\tilde{\mathbf{x}}_k + \tilde{\mathbf{B}}\mathbf{u}_k \\ \tilde{\mathbf{y}}_{k+1} &= \tilde{\mathbf{C}}\tilde{\mathbf{x}}_{k+1} + \tilde{\mathbf{D}}\mathbf{u}_{k+1}.\end{aligned}\quad (33)$$

To assess the closed-loop estimation accuracy, we constructed a Luenberger observer defined in (3) with initial conditions  $\tilde{\mathbf{x}}_0 = \mathbf{0}$  and  $\tilde{\mathbf{y}}_0 = \mathbf{0}$ , and selecting the gain matrix  $\mathbf{L}$  for each ROM type and each dimension  $r$ , such that the  $r$  eigenvalues of  $\tilde{\mathbf{A}} - \mathbf{L}\tilde{\mathbf{C}}$  were real and evenly distributed between -0.5 and 0.5. The values of  $\mathbf{y}_k$  in (3) were taken from the output data of each trial.

#### D. Effects of Training Hyperparameters on Estimation Accuracy

We evaluated the open-loop and closed-loop estimation accuracy of each ROM over the 37 test trials specified in Table I, the results of which are summarized in Figure 4. For each trial, we computed the relative estimation error  $e_y$  of each ROM as

$$e_y = \frac{\|\mathbf{Y} - \tilde{\mathbf{Y}}\|_F}{\|\mathbf{Y}\|_F} \quad (34)$$

where  $\mathbf{Y} = [\mathbf{y}_0, \dots, \mathbf{y}_{K-1}]$  is a matrix of ground truth outputs from the simulated soft robot and  $\tilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_0, \dots, \tilde{\mathbf{y}}_{K-1}]$  is a matrix of output estimates produced by a given ROM and estimation law. Notably, a value of  $e_y = 1$  coincides with the quality of prediction that would result from an estimation law that only predicts  $\tilde{\mathbf{y}}_k = \mathbf{0}$  (i.e., the neutral position of the robot).

We computed  $e_y$  over  $K = 1000$  timesteps using both open-loop and closed-loop estimation laws with ROMs generated for each  $r = 2, 4, \dots, 20$  and one, two, or three training trials (Fig 4a). In open-loop rollouts with each trial, we found that all models produced an average of approximately  $e_y = 1$  for  $r = 2$ . On average, ROMs synthesized using the ERA and LOpInf methods improved in open-loop estimation accuracy as the ROM dimension increased, up to a limit within the tested values of  $r$ . Additionally, we found that as the average relative estimation error improved, the variance over test trials tended to increase. In contrast, the open-loop estimation error of ROMs generated by DMDc tended to increase with  $r$ .

In the closed-loop estimation setting, the estimation errors produced by each ROM tended to improve. We found that the values of  $e_y$  produced by ERA-based and DMDc-based

ROMs were lower than those produced by LOpInf-based ROMs for smaller  $r$ . However, the values of  $e_y$  generated from ERA-based and DMDc-based ROMs exhibited local optima between reduced-order state dimensions of  $r = 10$  to  $r = 16$ , depending on the number of trials used for training. In contrast, the values of  $e_y$  resulting from LOpInf-based ROMs demonstrated an approximately monotonic improvement with increasing  $r$ . Additionally, while increasing the amount of training data tended to improve closed-loop estimation accuracy for LOpInf-based ROMs (Fig 4b.iii), this tendency was less-so apparent in DMDc-based ROMs (Fig 4b.ii).

## VI. ROMPC IMPLEMENTATION AND EVALUATION ON FEASIBLE REFERENCE TRAJECTORIES

We next combined the closed-loop state observer (3) and model predictive control (4) to evaluate each model reduction method in our proposed ROMPC scheme (Fig. 3) on our simulated testbench. To implement the control scheme, we developed custom modules in SOFA using a commercially available solver (GUROBI, GUROBI Optimization LLC) for the large-scale quadratic programs that result from applying our ROMPC formulation over long time horizons. Using our simulated testbed, we investigated how ROM dimension and quantity of training data and objective function tuning affects shape tracking performance of the resulting controller.

We describe this empirical study in this section, beginning with how we generated feasible reference trajectories in Section VI-A, and then discussing preliminary objective function tuning in Section VI-B. Finally, we investigate the effects ROM hyperparameters and objective function tuning in Section VI-C and compare the best-case ROMPC policies generated with each model reduction method in Section VI-D.

### A. Feasible Reference Trajectory Generation

As with many soft robots, our simulated system exhibits highly underactuated dynamics [19], necessitating consideration for the dynamic feasibility of any given reference trajectory: infeasible reference trajectories can preclude accurate tracking, making it difficult to assess whether or not poor tracking is due to controller design and tuning or due to the infeasibility of the reference trajectory. In this first set of experiments, we addressed this problem of generating dynamically feasible reference trajectories by using the output trajectories from the eight trials of our dataset denoted by asterisks in Table I—as these output trajectories have already been realized, their feasibility is guaranteed.

### B. Objective Function Tuning

Our ROMPC formulation in Section III poses a multiobjective quadratic program where the coefficients of the diagonal matrices,  $\mathbf{W}_y$ ,  $\mathbf{W}_u$ , and  $\mathbf{W}_{\delta u}$ , determine the relative importance of reference tracking, magnitude of the control input, and discrete rate of change in the control input, respectively [47].

We iteratively tuned this objective following multiple heuristics. Because we used reference trajectories that were guaranteed to be feasible and all actuators were identical,

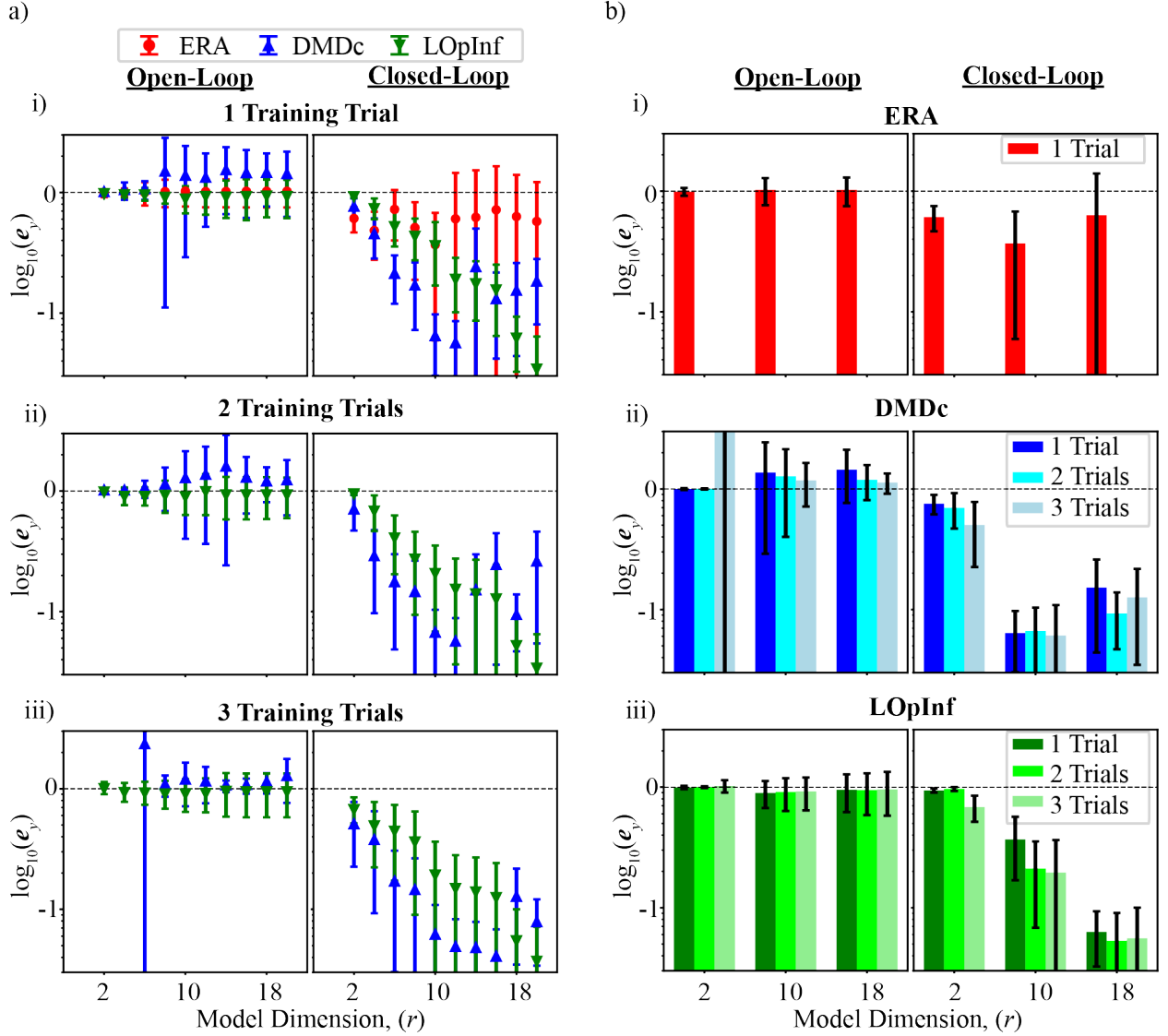


Fig. 4. Prediction error of open-loop and closed-loop estimation laws based on ROMs generated via the ERA (red), DMDc (blue), and LOPInf (green) methods. a, i-iii) Comparison across ROM estimation methods of relative estimation error,  $e_y$ , as a function of ROM dimension,  $r$ , grouped by the amount of training data used to train ROMs of each method: i) one trial, ii) two trials, iii) three trials. For nearly all synthesized ROMs, except higher dimensional ROMs generated via LOPInf, open-loop relative estimation error over a trial remained approximately at or above  $e_y = 1$  (black dotted line). Values for DMDc-based ROMs trained on three trials of data and with state dimensions of  $r = 2$  ( $e_y = 2946.6 \pm 3601.7$ ) and  $r = 4$  ( $e_y = 2150.3 \pm 2665.9$ ) were off the scale and are omitted for visual clarity. In contrast to the open-loop estimation schemes, the closed-loop state observers significantly improve estimation accuracy for nearly all models. b, i-iii) Estimation accuracy of each ROM as a function of the state dimensions tested in our proposed control loops, grouped by ROM synthesis method: i) ERA-based ROMs, ii) DMDc-based ROMs, iii) LOPInf-based ROMs.

we limited selection of  $\mathbf{W}_u = c_u \mathbf{I}_m$  and  $\mathbf{W}_{\delta u} = c_{\delta u} \mathbf{I}_m$ , i.e., scalar multiples of the identity matrix. Additionally, we selected the diagonal entries of  $\mathbf{W}_y$  that weigh the  $z$ -displacement tracking error of each control node be a constant  $c_z$ , with the remaining entries of  $\mathbf{W}_y$  being set to 0. We applied no tracking penalties to elements of the output corresponding to the  $x$ -displacements of each control node as we found that doing so could lead to undesirable behaviors (e.g., overly-aggressive control inputs). These undesirable behaviors could potentially be due to the fact that defining both the  $x$ - and  $z$ -displacements is redundant given the limited configuration space of the robot and the fixed arc lengths between control points.

Using the output trajectory realized in Trial 40 in Table I as reference, we iteratively tuned the objective function weights with the heuristic of finding a single set of values for  $\mathbf{W}_y$ ,  $\mathbf{W}_u$ , and  $\mathbf{W}_{\delta u}$  that could provide the best control across controllers operating on ROMs from each synthesis method. During tuning, all ROMs were of dimension  $r = 18$  and trained on three simulation trials for DMDc-based and LOPInf-based ROMs and on one trial of data for the ERA-based ROM (training trial selection is described in Section V).

Our iterative tuning process involved varying the scalar value defining each of the weighting matrices in our objective function with the goal of improving reference tracking as much as possible for a given class of ROM. We then used those

same objective function coefficients in trials with another class of ROM, repeating this iteration until tracking performance stopped improving for any controller. Notably, we found that manually tuning the ERA-based controller was the most challenging as small changes to weighting coefficients could result in aggressive or unstable behavior. This tuning process resulted in values of  $c_u = 1500$ ,  $c_{\delta u} = 1600$ , and  $c_z = 0.6$ , used for all three ROM classes and dimensions  $r$ .

### C. Effects of ROM Hyperparameters on Tracking Error

Using the tuned objective function, we conducted control experiments to determine the tracking performance of our ROMPC scheme as a function of the reduced-order state dimension and amount of training data used to generate the ROMs used for control. To assess tracking performance of various controllers, we selected eight reference trajectories from the 40-trial dataset presented earlier (denoted by asterisks in Table I). These trajectories were selected to represent a variety of dynamically feasible behaviors.

1) *Error Metrics*: To quantify the tracking performance, we considered two error metrics. First, we considered the relative tracking error of the robot's full body over the entirety of a control trial,

$$e_r = \frac{\|\mathbf{Y}^* - \mathbf{Y}\|_F}{\|\mathbf{Y}^*\|_F}, \quad (35)$$

where  $\mathbf{Y}^* = [\mathbf{y}_0^*, \dots, \mathbf{y}_{K-1}^*] \in \mathbb{R}^{p \times K}$  is reference output trajectory for a given trial and  $\mathbf{Y} = [\mathbf{y}_0, \dots, \mathbf{y}_{K-1}] \in \mathbb{R}^{p \times K}$  is the realized output trajectory for the same trial. Notably, a value of  $e_r = 1$  indicates a quality of tracking that would result from a "do-nothing" policy (i.e., one that produces  $\mathbf{u}_k = \mathbf{0}$  for all  $k$ ). We also considered a pointwise relative tracking error

$$e_{r,j} = \frac{\|\mathbf{Y}_j^* - \mathbf{Y}_j\|_F}{\|\mathbf{Y}_j^*\|_F} \quad (36)$$

where  $\mathbf{Y}_j^* = [y_{j,0}^*, \dots, y_{j,K-1}^*] \in \mathbb{R}^K$  are the reference (target) outputs at control point  $j$  and  $\mathbf{Y}_j = [y_{j,0}, \dots, y_{j,K-1}] \in \mathbb{R}^K$ . Similar to the definition of  $e_r$ , a value of  $e_{r,j} = 1$  indicates a tracking quality at point  $j$  equivalent to that which would result from a control policy that keeps that point stationary, such as the same "do-nothing" policy described previously.

2) *Tracking Performance as a Function of Reduced-Order State Dimension*: We first evaluated the tracking performance of our ROMPC scheme as we varied the value of  $r$  used to generate each ROM (Fig. 5.a). For this experiment, we considered ROMs of dimension 2, 10, and 18 and used models trained only on Trial 40 in Table I when computing  $e_r$  over the eight selected reference trajectories.

Within the tested values of  $r$ , the values of  $e_r$  produced by the LOPInf-based controllers were either on par with, or significantly lower than those produced by DMDc-based and ERA-based controller. Moreover, we observed a monotonic decrease, on average, in  $e_r$  with increasing  $r$  for LOPInf-based controllers. In contrast, DMDc-based and ERA-based controllers exhibited local minima in tracking error at values of  $r = 10$  and  $r = 2$ , respectively. Regardless, the best-case DMDc-based and ERA-based controllers in this experiment

still exhibited significantly higher full-body tracking error than the best-case LOPInf-based controller.

3) *Tracking Performance as a Function of ROM Training Data Quantity*: Next, we assessed how tracking performance varied as a result of the amount of data used to train each ROM (Fig. 5.b). For this experiment, we evaluated the tracking error  $e_r$ , produced by ROMs generated from the same sets of one, two, and three training trials presented previously, all of the same dimension,  $r = 18$ . Due to the formulation of ERA-based ROMs used here, we only assessed the performance of the ERA-based controller when the ERA-based ROM was trained on one trial of data.

We found that the LOPInf-based controllers produced approximately constant values of  $e_r \approx 0.45 \pm 0.08$  over all ROMs tested. The DMDc-based and ERA-based controllers produced tracking errors of  $e_r \approx 1$  for models trained on one trial of data. Notably however, the tracking performance of DMDc-based controllers degraded as the amount of data used for training increased.

4) *Tracking Performance as a Function of Varying Tracking Penalty Weights*: Finally, we considered how tracking performance was affected by varying the scheme used to define weighting coefficients in  $\mathbf{W}_y$  for tracking the reference  $z$ -displacements at control nodes across the soft robot's body (Fig. 5.c). For this experiment, we considered two weighting schemes: 1) an equal weighting of tracking penalties across the robot's body (i.e., the same scheme as described previously); and 2) a posterior-focused weighting scheme where tracking penalties were only non-zero for the posterior half of the robot. For the latter weighting scheme,  $\mathbf{W}_y$  was still kept as a diagonal matrix wherein the only nonzero diagonal values were those applying penalty to the  $z$ -displacement tracking error for the 10 posterior-most control points along the simulated robot's centerline. Using values of 1.2 for these nonzero entries of  $\mathbf{W}_y$ , we found that, relative to the equal weighting scheme, the posterior-focused scheme slightly increased the average and standard deviations of the full-body tracking errors resulting from the DMDc-based and LOPInf-based controllers, while negligibly affecting the average tracking performance of the ERA-based controller.

### D. Comparison of Best-Case Controllers

Based on the results from the previous section, we compared the pointwise tracking errors resulting from the best-case controllers representing each ROM synthesis technique in our comparative study (Fig. 5.d). The ERA, DMDc, and LOPInf ROMs used were of dimension 10, 2, and 18, respectively, and the ERA and DMDc ROMs were each trained on one trial of data while the LOPInf ROM was trained on 3 trials of data.

We first considered how each controller tracked the reference trajectories from trials on which their respective models were trained (Fig. 5.d.i). On these training trials, we found that over the entirety of the robot's body, the tracking errors of the DMDc-based and ERA-based controllers were approximately equal to, or higher than  $e_{r,j} = 1$ , qualitatively indicating performance that was on par with, or worse than, a "do-nothing" policy. In contrast, the best-case LOPInf-based

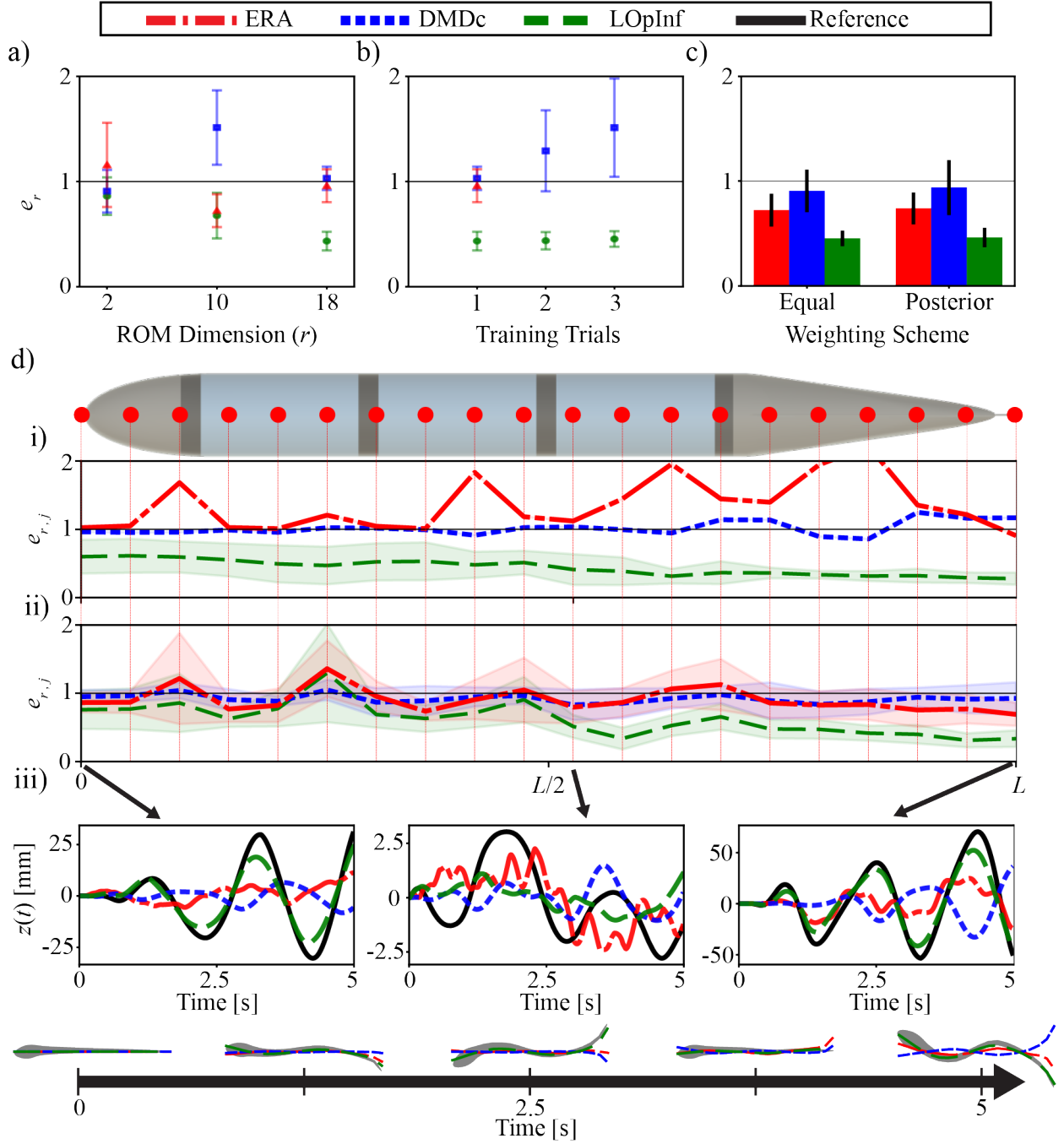


Fig. 5. Tracking performance of ROMPC scheme over ROM types, dimensions, quantities of training data, and objective function tuning schemes. The average tracking error for ERA-based (red), DMDc-based (blue), and LOpInf-based (green) controllers generated through this process followed from an extensive tuning process that considered the effects of (a) reduced-order dimension,  $r$ , for ROMs trained on one trial of data and (b) quantity of training data used for generated ROMs of dimension 18 on tracking performance. (c) Our tuning process also explored the effects of varying coefficients in the MPC objective function to apply equal weighting to tracking penalties across the body as well as weights defined by a posterior-focused scheme. (d) Relative tracking error of the best observed controllers based on each ROM generation method computed over the robot's body length. Tracking error was computed over known feasible trajectories from (i) the trials used to train each ROM and (ii) a set of eight test trials from our dataset. The average tracking error for each best-case controller over each set of trials is shown as dotted lines, with surrounding shaded regions representing one standard deviation of tracking error at a given point. (iii) Output trajectories generated by each controller are shown at control points located at the head (top left), midpoint (top middle), and tail (top right) of the robot for an example reference trajectory (black) extracted from Trial 33 of our dataset. The full reference centerline (black) of the simulated robot are shown for the same trial (bottom) overlaid with centerlines produced by each best-case controller shown in their respective colors (deflections in the  $z$ -direction amplified by a factor of five for sake of visualization).

controller consistently performed better than a “do-nothing policy” with an average tracking error of  $e_{r,0} = 0.60$  at the head of the robot and a value of  $e_{r,19} = 0.28$  near at the tail.

We then considered the pointwise relative tracking errors on the eight test trials used in our prior estimation studies (Fig. 5.d.ii). Notably, the best-case ERA-based and DMDc-based controllers generally exhibited *better* pointwise tracking performance on these test trials when compared to their tracking performance with reference trajectories from the training trials discussed previously. In contrast, the tracking performance of the best-case LOPInf controller degraded slightly near the anterior end of the robot. Furthermore, when considering the timeseries  $z$ -displacement for each of the control points, we found that at the anterior and posterior ends of the robot, the best-case LOPInf-based controller typically exhibited tracking performance that best matched the behavior of reference trajectories (Fig. 5.d.iii). In contrast, the best-case ERA-based and DMDc-based controllers often generated output trajectories with irregular oscillations at much higher frequencies than those of the oscillations prescribed by the reference trajectories.

## VII. CONTROL EXPERIMENTS WITH INCREASINGLY COMPLEX REFERENCE TRAJECTORIES

Using our simulated testbed and the best-case controllers described in Section VI-D, we conducted two control experiments wherein the simulated robot tracked full-body reference trajectories generated from systems of increasing complexity and realism. The first experiment, described in Section VII-A, focused on tracking trajectories generated from an empirical kinematic model of anguilliform swimming. The second experiment, described in Section VII-B, used reference trajectories extracted from experiments with a reduced-scale physical analog of our simulated robot. For each experiment, we describe how we generated reference trajectories as well as the tracking performance of the best-case controllers representing each model reduction method.

### A. Control Experiment 1: Tracking Bioinspired Full-Body Trajectories

We first evaluated how well the simulated platform can track the traveling wave kinematics integral to natural anguilliform locomotion during straight-line swimming [53]. Along with being a key element of anguilliform swimming, traveling wave propagation through a finitely actuated elastic structure is a uniquely dynamic phenomenon [53]. Thus, the goal of this control experiment to characterize how well the simulated system could track the gaits generated by this dynamic process.

1) *Reference Trajectory Generation*: We used the empirically derived model of centerline kinematics presented in [53], that describes the lateral position  $z_m$ , of an eel’s centerline at time  $t$ , as

$$z_m(s, t) = A e^{\alpha(\frac{s}{L}-1)} \sin(ks - 2\pi ft), \quad (37)$$

where  $s \in [0, L]$  indicates the axial position along the robot’s centerline (with a value of  $s = 0$  corresponding to the anterior-most point),  $L$  is the length of robot,  $A$  is the tail

TABLE II  
PARAMETERS DEFINING TRACKING EXPERIMENTS WITH BIOINSPIRED GAITS.

| Gait Parameter                | Parameter Values |
|-------------------------------|------------------|
| Tail beat amplitude, $A$ [mm] | 30               |
| Growth rate, $\alpha$         | 1.0, 3.5         |
| Tail beat frequency, $f$ [Hz] | 0.5, 1.0         |
| Body wavenumber, $k$          | 0.5, 1.0, 1.5    |

beat amplitude,  $k$  is the body wavenumber,  $f$  is the tail beat frequency, and  $\alpha$  is the amplitude growth rate [53].

We selected a broad set of gait parameters to generate reference trajectories from Table II, including values that have been empirically observed in anguilliform animals [53]. For each of the 12 combinations of gait parameters shown in Table II, we simulated tracking with each best-case ROMPC controller found in the previous section over a duration of 10 seconds (i.e., 1000 time steps). We quantified tracking performance with the same error metric defined in (35), considering only the tracking performance after the first 5 seconds of each trial. We ignored this transient portion of each trial to only account for the steady-state tracking performance of each controller.

2) *Tracking Performance*: For all gait parameters tested, the LOPInf-based controller produced the lowest tracking error, though the difference in tracking errors was nearly negligible in trials where  $\alpha = 1.0$  and  $f = 1.0$  Hz (Fig. 6.a). For most sets of parameters, all controllers demonstrated tracking performance better than that of a “do-nothing” control policy. We also found that under the DMDc-based and LOPInf-based ROMPC policies, tracking performance often significantly improved for the higher growth rate tested. This could potentially be attributed to the fact that larger values of  $\alpha$  significantly reduces the desired amount of lateral movement near the anterior end of the simulated robot (Fig. 6.a), and this reduced lateral movement better aligns with the boundary conditions we applied in our structural simulation that force lower displacement near the base of the robot’s head.

To further analyze the tracking behavior of our system, we considered the simulated robot’s kinematics over a single cycle of operation (Fig 6.b). We found that, over the selected gait parameters, the midbody of the robot often exhibited the least movement while the anterior and posterior ends of the robot contributed most significantly to the tracking performance. Also, all controllers typically exhibited the lowest tracking error near the posterior end of the robot, with the LOPInf-based controller consistently showing superior tracking performance at this point. The consistent improvement in tracking at the tail of robot can likely be attributed to the fact that the generated reference gaits prescribe the largest lateral displacement at the tail of the robot. This means that during control optimization, reducing error at the tail has the largest potential for reducing the loss prescribed by our objective function.

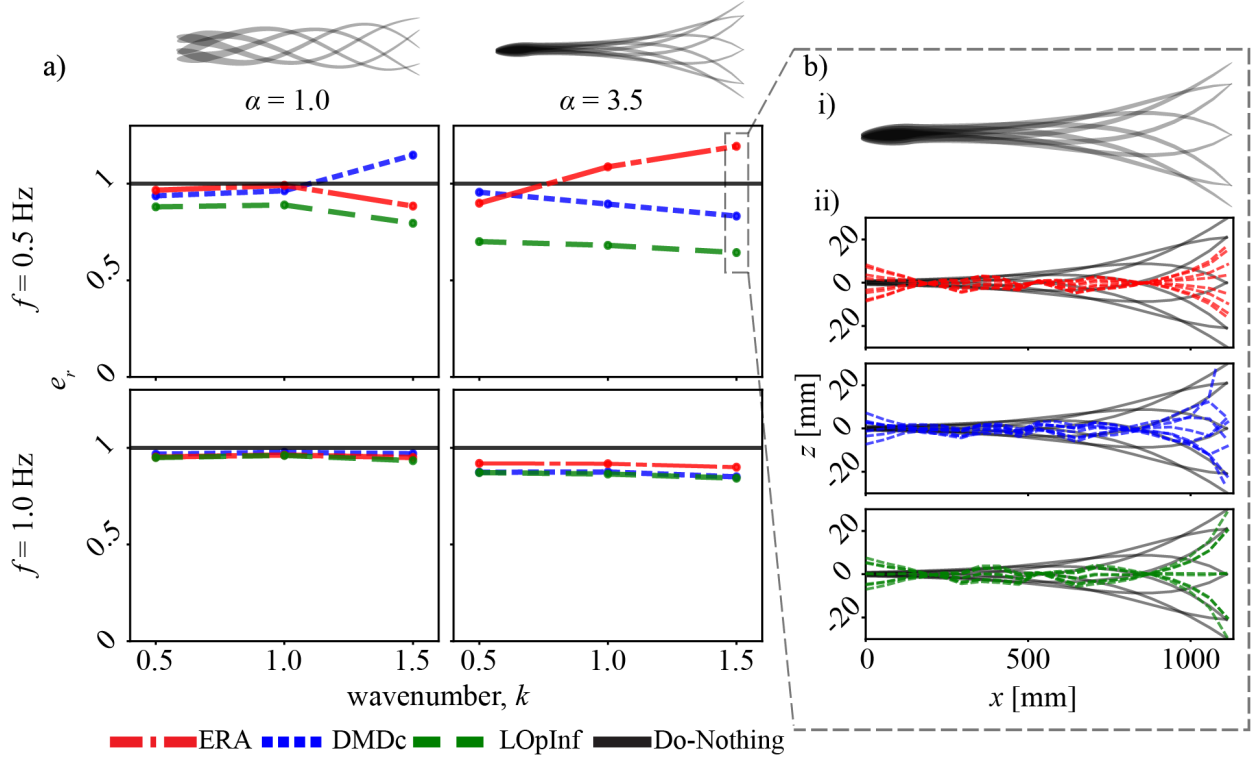


Fig. 6. Bioinspired trajectory tracking performance. a) Full-body relative tracking error of robot over all combinations of bioinspired gait parameters. Relative tracking errors were computed as defined in (35). Results are separated by gait frequencies  $f$  (rows) and growth rates  $\alpha$  (columns) with example centerlines produced with a given  $\alpha$  shown above each column. b) Centerlines of robot produced over a period of steady-state operation using gait parameters  $f = 0.5$  Hz,  $k = 1.5$ ,  $\alpha = 3.5$ , and  $A = 30$  mm. Centerline snapshots from the reference trajectory (i) for this gait are shown as black lines on each subplot and dotted color lines in (ii) show the centerline snapshots for the best-case ERA-based (top), DMDc-based (middle), and LOpInf-based (bottom) controllers.

### B. Control Experiment 2: Tracking Trajectories Generated by Experimental Robot

In our second set of control experiments, we assessed the capacity of the simulated robot to reproduce dynamic swimming behaviors exhibited by the experimental platform developed in [56], [57]. The physical system from which data was collected was comprised of the same materials described in Section II, but is one-third the scale of our simulated system and has minor modifications to enable tethered hydraulic actuation. Further details of these modifications and the system's physical realization can be found in [56] and [57].

1) *Reference Trajectory Generation*: To generate reference trajectories for the simulated robot, we extracted the centerline trajectories produced by the physical robot over the 20 thrust-characterization experiments conducted in [57]. These 20 experiments implemented open-loop control policies designed to approximate anguilliform swimming with a tail beat frequency of 0.5 Hz, nominal wavenumbers of  $k \in \{0.5, 0.6, 0.75, 1.0, 1.5\}$ , and inputs magnitudes meant to generate anterior-focused (AF), balanced (B), middle-focused (MF), and posterior-focused (PF) bending [57]. The open-loop control policies used in [57] were designed assuming deterministic, quasi-static, and constant-curvature deformation of each actuated segment, resulting in deformations like that shown in Figure 7.b. During these physical experiments, the robot's head segment was held fixed, which enforced similar boundary conditions to the simulated platform where the head

is allowed to pivot about its base as described in Section II-B.

For each of the physical experiments in [57], the robot's centerline kinematics were recorded via visual tracking of colored markers placed on each of the rigid couples of the robot. For each video frame, we used these colored markers to estimate the bend angle of each segment, which we then used to fit constant-curvature arcs across the centerline of the robot. We observed that the oscillation frequency of the system was too slow to induce significant bending in the passive tail. So we approximated the configuration of the tail as a line segment extending from the posterior-most actuated segment of the robot.

While our simulated control loop operated at a frequency of 100 Hz, the sample rate of the visually extracted centerlines was 12 Hz. To address this discrepancy, we linearly interpolated the extracted centerlines over time to produce reference trajectories suitable for our control loop. Finally, we spatially discretized the estimated centerlines into constant-length curves, with the end-points of these curves corresponding to the desired location of the simulated platform's control points. As the experimental data was only collected from the base of the physical robot's head to its tail, we discretized these curves into 17 segments to generate reference trajectories for control points 2-19 (as labeled in Fig 2).

2) *Tracking Performance*: Since the two anterior-most control points were fixed on the physical platform, we modified the cost function for the control optimization to enforce 0

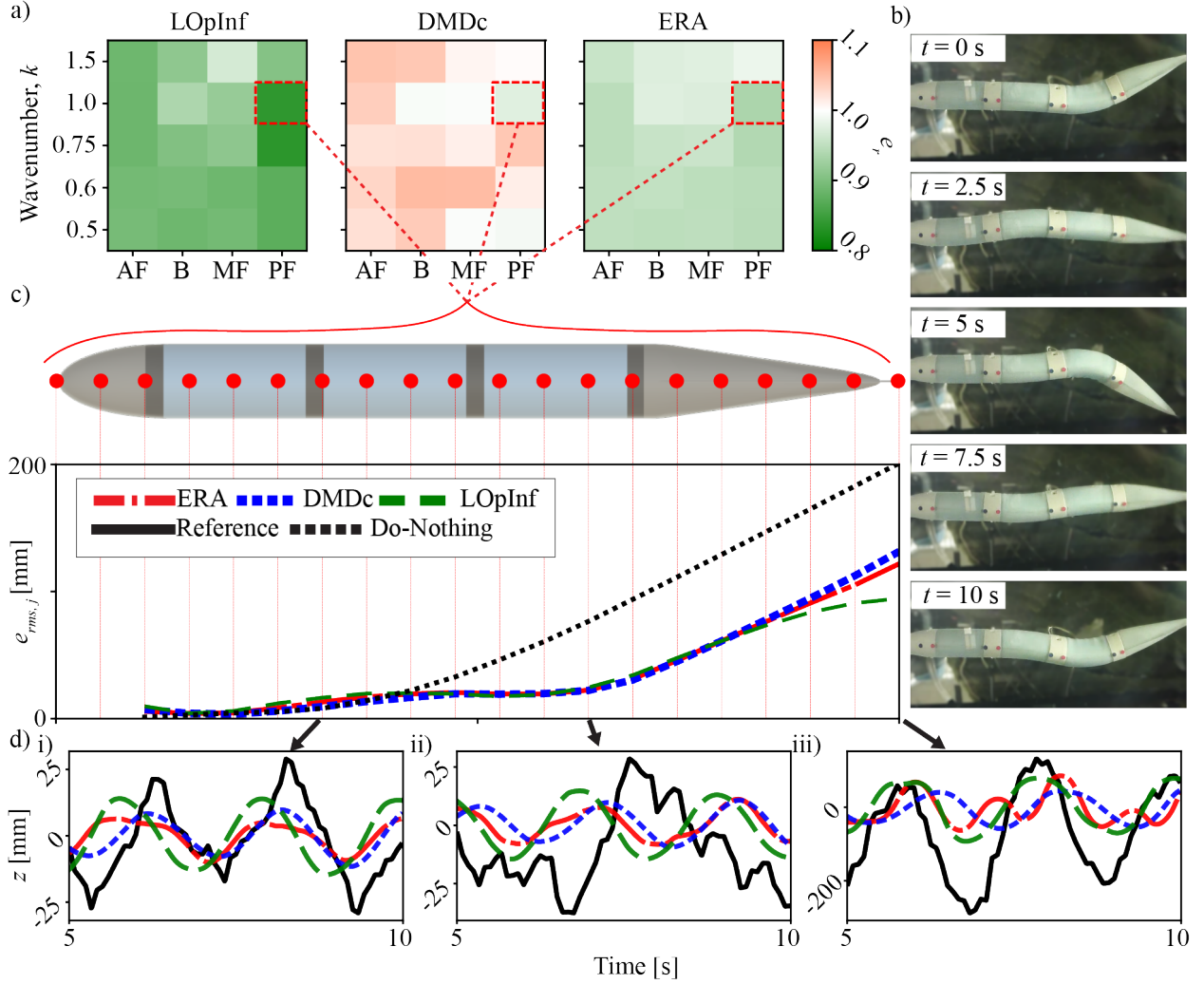


Fig. 7. Tracking reference trajectories extracted from experiments with physical robot platform. Using the data collected from the 20 physical experiments conducted in [57], we constructed reference trajectories with which we tested the tracking performance of our simulated system with the best-case controllers found with each ROM synthesis method. (a) Full-body tracking error maps resulting from tracking experiments over five gait wavenumbers and four sets of input amplitudes that are either anterior focused (AF), balanced (B), middle focused (MF), and posterior focused (PF). Design and selection of gait parameters are detailed in [57]. (b) Images of physical robot during a trial with wavenumber  $k = 1.0$  and posterior-focused input amplitudes. (c) We computed the pointwise relative error over the last 5 seconds of tracking this trial for each controller and (d) show the output trajectories at three points along the robot's body.

tracking penalty on the  $z$ -displacement error at these points, keeping the remaining coefficients of the cost function as defined in Section VI-D. As before, we considered the full-body relative tracking error metric defined in (35), ignoring the contributions of the two fixed control points, and again only considering the last 5 seconds of each simulated experiment to reduce the effect of transient behaviors expected from the simulated platform starting from rest. We also considered a pointwise root mean square (RMS) error across the length of the simulated robots body, defined as

$$e_{RMS,j} = \frac{\|\mathbf{Y}_j^* - \mathbf{Y}_j\|_F}{\sqrt{K-1}}, \quad (38)$$

We considered this RMS error in lieu of the pointwise relative error  $e_{r,j}$ , introduced in (36), because these experiments prescribed zero (or approximately zero) displacement anterior control points, meaning that even slight lateral movement from

the anterior control points of our simulated platform could produce extreme values of  $e_{r,j}$ , making it difficult to compare tracking quality between controllers.

In considering  $e_{RMS,j}$ , we found that the RMS tracking error near the posterior end of the robot often demonstrated the largest difference in tracking performance between each controller (Fig. 7.c). Even though the DMDc-based and ERA-based controllers sometimes exhibited tracking performance better than that of the best-case LOpInf-based controller near the anterior end of the robot, this difference was always minor in comparison to the improved tracking performance of the LOpInf based controller at the robot's posterior end.

For all trials, all controllers followed a similar trend in RMS tracking error over the length of the robot's body—RMS tracking error remained relatively low up to the posterior half of the robot, after which the error tended to increase more rapidly. We believe this behavior is best explained through the

timeseries responses of each control point under each controller (Fig. 7.d). Tracking performance most heavily degraded when the desired magnitude of the reference became large—a phenomenon that always happened near the posterior of the robot. We contend that large deformations led to degraded tracking because the maximum pressure constraints applied to our simulated system were likely lower than what was necessary to produce the bending prescribed by the reference trajectories. Simulating pressures beyond these saturation constraints negatively impacted simulation stability, limiting our capacity to reduce tracking error by expanding the saturation window. However, we validated this hypothesis by observing that the input trajectories produced by each controller would frequently saturate (Fig. 8). Further inspection of the input trajectories to each actuator showed that ERA-based controller often switched seemingly sporadically while the DMDC-based and LOPInf-based controllers oscillated at the expected input frequency of 0.5 Hz, with the LOPInf controller producing control inputs closest to the square-wave nominal signal used by the physical system, though with phase shifts.

### VIII. DISCUSSION

The primary objective of this comparative study is to quantitatively and qualitatively compare representative methods for data-driven model reduction in an identical setting requiring dynamic shape control of a soft robot. In this section, we use the results presented thus far to compare each model reduction method in Section VIII-A. We then describe various limitations of this study in Section VIII-B.

#### A. Comparison of Model Reduction Methods for ROMPC-based Shape Control

Two common heuristics guiding the use of data-driven models for control are 1) an expected tradeoff between model fidelity and computational cost, often linking higher-dimensional models with more accurate simulations; and 2) an expectation that increasing the amount or quality of training data will increase or maintain model accuracy [6], [20]. Within the evaluated range of ROM state dimensions  $r$ , our estimation experiments in Section V demonstrated that the LOPInf-based models best matched these expectations in both open-loop and closed-loop prediction settings. This behavior could be attributed to the second-order mechanical structure enforced by LOPInf models. Provided sufficiently expressive data, the Lagrangian form described by equation (12) has been shown to effectively capture the dominant dynamics of a soft system with purely linear dynamics [49], producing low open-loop prediction errors over long time horizons. As our simulated system integrates geometric nonlinearity, we expected that the linear ROMs produced via LOPInf would not capture these nonlinearities. However, compared to the DMDC and ERA based models, the LOPInf-based models nonetheless produced the most accurate open-loop estimates on average for every  $r$  except for when  $r = 2$  and models were trained on only one trial of training data.

In the closed-loop estimation setting, the ERA-based and DMDC-based ROMs often produced lower estimation error

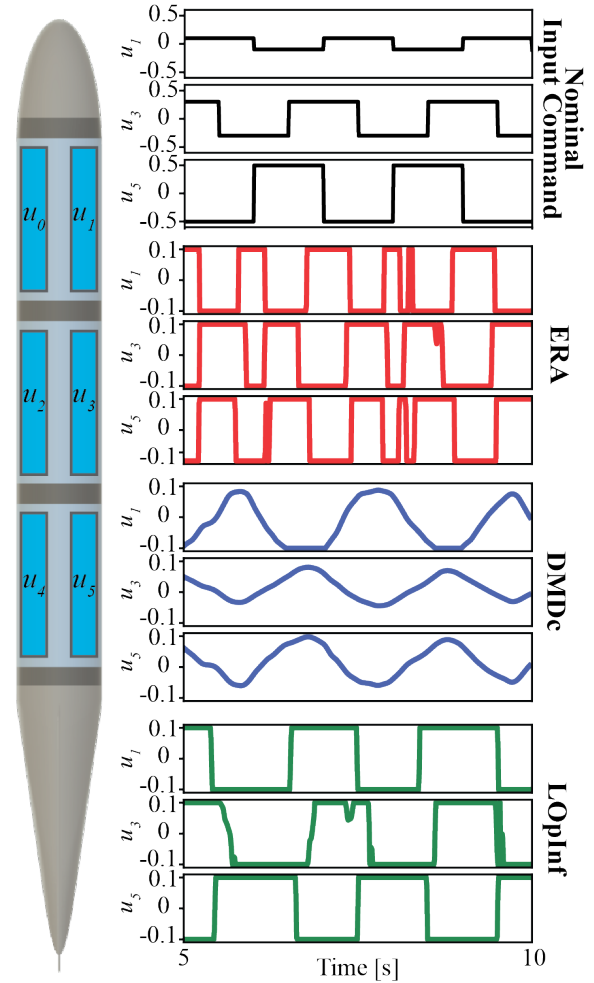


Fig. 8. Control signals produced by simulated controller when tracking reference trajectories generated from experiment with the physical robot in [57] with wavenumber,  $k = 1.0$  and (PF) bending amplitudes. The nominal input commands (black) used for the physical experiment represent the control signals sent from the microcontroller controlling the physical system to the electronic speed controllers driving the pumps connected to each actuator. We normalized these commands such that the maximum peak-to-peak amplitude of the command signal that the microcontroller produced has a value of 1.0. As these are the open-loop signals generated by the microcontroller, they do not account for the dynamics and delays induced by the electrical drivers, pumps, and fluidic channels of the physical system and are not pressures. The non-dimensionalized pressure control signals produced by the ERA-based (red), DMDC-based (blue) and LOPInf-based (green) policies for inputs  $u_1$ ,  $u_3$ , and  $u_5$  were fed directly into the full-order simulation.

than LOPInf-based ROMs for  $r \leq 12$  when one or two trials of training data were used and  $r \leq 18$  when three trials of training data were used. However, the DMDC-based and ERA-based models would exhibit sudden increases in estimation errors as the reduced-order state dimension increased beyond these values. We hypothesize that this behavior could be due to the fact that the DMDC algorithm presented in [63] extracts dominant spatial modes presented in the data and constructs a system representation where the state corresponds to a linear combination of those spatial modes. Therefore, it might be possible that by learning from datasets that inappropriately over-represent certain spatial modes, the prediction quality of the model can degrade even if a higher state dimension or

more data are used during training. In a similar vein, as the ERA-based models could only learn from a single trial of data, albeit generated with rich dynamics, it is possible that the trial we selected only effectively represented a limited set of observable modes amenable to closed-loop estimation. Increasing state dimension to include further modes could therefore have degraded closed-loop estimation accuracy.

When comparing the ROMs in our proposed ROMPC scheme, we initially hypothesized that ROMs producing the most accurate state reconstruction would admit controllers with the most accurate tracking. This hypothesis was based on the idea that the control optimization in (4) requires an accurate estimate of the current state, which is provided by the closed-loop state observer in our control loop. While we found that this hypothesis held true in the best-case ERA-based and LOPInf-based controllers, it did not hold true for the best-case DMDc-based controller. In fact, the best DMDc-based controller was based on a ROM trained on one trial of data with reduced-order state dimension  $r = 2$ , which notably produced the worst-case closed-loop estimation error among all hyperparameters tested.

Along with enabling a comparison between three classes of data-driven model reduction techniques in a control setting, our comparative approach additionally provided insights into the design and operation of the anguilliform robot that we simulated. When considering the pointwise tracking errors that resulted from tracking dynamically feasible reference trajectories, we found that all of the best-case controllers tended to exhibit spikes in tracking error just before each of the rigid couples comprising the simulated robot (Fig. 5.d.ii). This consistent pattern could indicate a potential reduction in either control authority or model fidelity (or both) near these regions, eliciting either design iteration or model refinement (or both) to improve tracking performance.

One of the most notable results from the control experiments in Section VII is that, even when tracking reference trajectories that are not known *a priori* to be dynamically feasible, the best-case LOPInf-based controller consistently produced superior tracking over the robot's body. This performance is best explained by the consistently lower tracking error near the tail of the robot, where reference displacements were often the highest and, therefore, of the highest priority to track given the ROMPC objective function. However, as the tail is unactuated, effectively tracking dynamic reference trajectories in this section of the robot indicates that the LOPInf-based controller is best accounting for the underlying structural dynamics of the robot during each trial. Notably, the best-case LOPInf-based controller produced this superior performance with an objective function that we had originally tuned to be identical for all of the ROMs tested. An objective function tuned specifically for a ROMPC policy based on a specific ROM instance will produce tracking errors that are equal to, or better than the ones shown in this work; we refrained from doing this to better compare the three model classes.

Moreover, the results from tracking bioinspired reference trajectories in Section VII-A illustrated that the selection of gait parameters can significantly affect tracking performance. This is attributed to some selections of gait parameters pro-

ducing reference trajectories that are closer to being feasible than those reference trajectories produced by other gait parameters. Nonetheless, none of these reference trajectories were expected to be fully feasible due to differences between our simulated platform and the physical systems from which reference trajectories were derived, as explained next.

### B. Limitations

Even though the LOPInf-based controller produced superior tracking performance in all of the experiments with feasible trajectories (Section VI), it regularly produced the highest pointwise relative tracking error in more anterior portions of the simulated robot's body—a trend that was also often apparent in the DMDc-based and ERA-based controllers, though less pronounced. In the setting of tracking feasible reference trajectories, this tracking error could be in part attributed to controller tuning, but it more likely stemmed from the fact that our ROMPC policy leveraged a single linear ROM to represent the dynamics of an otherwise nonlinear full-order system. Nonetheless, we considered this error a baseline for our ROMPC scheme against which we compared the results of the experiments that utilized infeasible reference trajectories.

Compared to this baseline, the experiments in Section VII regularly exhibited a higher full-body tracking error for all models. We hypothesize that the higher tracking errors in these experiments can be attributed to one of two sources. First, the reference trajectories used in these studies were generated from systems that involved forcing from FSI that were not included in our simulated testbench or datasets. Even though all of the model reduction methods used here could be made to account for FSI, our full-order model could not represent these effects due to current limitations in modern simulation tooling for fluidically driven soft robots operating underwater. A second potential source of tracking error beyond the aforementioned baseline may have stemmed from structural and other operational differences between the simulated platform and the physical systems from which reference trajectories were generated. Along with any differences in shape between our simulated robot, the eels from which trajectories were modeled in [53], and the physical platform in [57], there were additional discrepancies in boundary conditions between all of these systems. These differences are of particular significance to consider for the data-driven ROMs produced via DMDc and LOPInf, as the reduced-order states generated by these methods represent dominant spatial modes present in the training data. Because all of the training data in this work were produced from simulations with boundary conditions that allow the head to pivot about the constraints described in Section II-B, the LOPInf and DMDc methods can only project the full-order data onto bases where this pivoting is represented. As a result, the generated ROMPC policies are limited in their capacity to account for alternative boundary conditions, such as those seen in Section VII-B where the entire head of the physical system is held stationary.

## IX. CONCLUSIONS AND FUTURE WORK

We presented a comparative study of linear, data-driven model reduction techniques in the setting of dynamic shape

control for soft robots. We selected three methods—ERA, DMDc, and LOPInf—representing different families of data-driven model reduction techniques and compared them on a custom simulated testbench of the structural dynamics of an anguilliform-inspired soft robot. Using this testbench, we generated a large-scale dataset that we used to assess the open-loop and closed-loop estimation accuracy of ROMs generated by each method as a function of ROM dimension and amount of training data. This dataset is publicly available at [55]. Finally, we compared each class of ROM in multiple control settings—one where full-body reference trajectories were guaranteed to be dynamically feasible, one where reference trajectories came from previously established bioinspired models of anguilliform locomotion, and one where the reference trajectory came from a reduced-scale physical analog of our simulated platform.

In this work, we in part explored reference trajectories that were guaranteed to be feasible as they had previously been simulated. The broader problem of efficiently generating feasible full-body trajectories for a given soft robot and task remains largely unanswered in the literature. However, this question is of critical importance for continued work on dynamic shape control of soft and hybrid-soft robots. This problem is challenging because generating feasible trajectories for a given task requires an accurate dynamic model of the system, which is fundamentally precluded in our setting by the lack of available tooling for simulating fluid-structure interaction in our full-order model. Therefore, future extensions of this work would benefit most from the development of accurate full-order three-dimensional FEM models that incorporate fluid-structure interaction. Provided such full-order models, future work could then leverage data-driven model reduction techniques such as LOPInf to produce low-dimensional surrogate models with which feasible—or approximately feasible—reference trajectories could be generated online for a given task.

Additionally, a valuable avenue for further research on ROMPC-based shape control of soft robots could focus on accounting for various nonlinear effects, such as geometric nonlinearity and nonlinear material behaviors, as well as other sources of modeling error in a manner that is still amenable for online control. Accounting for these effects in both the simulation and data collection phase as well as in the modeling capacity of any synthesized ROMs is important for improving the performance of the resulting ROMs in both estimation and control settings. One potential approach relevant to the DMDc-based and LOPInf-based controllers could focus on developing alternative methods for identifying relevant bases onto which full-order state is projected during model synthesis. Another approach could leverage various extensions of the LOPInf method that account for nonlinear dynamics present in the full-order model while preserving the physics-based structure of the system [51], [69]; using these methods in an online dynamic shape control setting would then require the use of nonlinear model-based control techniques that appropriately incorporate the resulting ROMs.

A ROMPC formulation that can effectively capture residual error dynamics with low data requirements could also be lever-

aged to address the problem of simulation-to-reality transfer—a challenge that is necessary to address for most modern robotic systems. Similarly, adaptive control frameworks could be developed that leverage streaming-based methods for updating ROMs during operation of the robot, allowing an initial model to be trained in simulation and updated after transferring to the physical system.

## ACKNOWLEDGMENTS

I.A., M.P., J.C., and M.T.T. were supported by the U.S. Office of Naval Research (ONR), grant no. N00014-22-1-2595. B.K. and H.S. were supported by ONR under award number N00014-22-1-2624. 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 ONR.

## REFERENCES

- [1] S. Aracri, F. Giorgio-Serchi, G. Suaria, M. E. Sayed, M. P. Nemitz, S. Mahon, and A. A. Stokes, “Soft Robots for Ocean Exploration and Offshore Operations: A Perspective,” *Soft Robotics*, vol. 8, pp. 625–639, Dec. 2021.
- [2] M. Cianchetti, C. Laschi, A. Menciassi, and P. Dario, “Biomedical Applications of Soft Robotics,” *Nature Reviews Materials*, vol. 3, pp. 143–153, May 2018.
- [3] A. J. Ijspeert, “Biorobotics: Using Robots to Emulate and Investigate Agile Locomotion,” *science*, vol. 346, no. 6206, pp. 196–203, 2014.
- [4] S. M. Youssef, M. Soliman, M. A. Saleh, M. A. Mousa, M. Elsamanty, and A. G. Radwan, “Underwater Soft Robotics: A Review of Bioinspiration in Design, Actuation, Modeling, and Control,” *Micromachines*, vol. 13, p. 110, Jan. 2022.
- [5] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, “Control Strategies for Soft Robotic Manipulators: A Survey,” *Soft Robotics*, vol. 5, pp. 149–163, Apr. 2018.
- [6] D. A. Haggerty, M. J. Banks, E. Kamenar, A. B. Cao, P. C. Curtis, I. Mezić, and E. W. Hawkes, “Control of Soft Robots with Inertial Dynamics,” *Science Robotics*, vol. 8, p. eadd6864, Aug. 2023.
- [7] 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 *2021 Robotics Science and Systems*, (Freiburg im Breisgau, Germany), RSS, 2019.
- [8] A. Singh, J. Sun, and J. Zhao, “Controlling the Shape of Soft Robots Using the Koopman Operator,” in *2023 American Control Conference (ACC)*, (San Diego, CA, USA), pp. 153–158, IEEE, May 2023.
- [9] C. D. Onal and D. Rus, “Autonomous Undulatory Serpentine Locomotion Utilizing Body Dynamics of a Fluidic Soft Robot,” *Bioinspiration & Biomimetics*, vol. 8, p. 026003, Mar. 2013.
- [10] D. D. K. Arachchige, Y. Chen, and I. S. Godage, “Soft Robotic Snake Locomotion: Modeling and Experimental Assessment,” in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, (Lyon, France), pp. 805–810, IEEE, Aug. 2021.
- [11] X. Liu, C. D. Onal, and J. Fu, “Reinforcement Learning of CPG-Regulated Locomotion Controller for a Soft Snake Robot,” *IEEE Transactions on Robotics*, vol. 39, pp. 3382–3401, Oct. 2023.
- [12] D. Q. Nguyen and V. A. Ho, “Evaluation on Swimming Efficiency of an Eel-inspired Soft Robot with Partially Damaged Body,” in *2021 IEEE 4th International Conference on Soft Robotics, RoboSoft 2021*, pp. 289–294, Institute of Electrical and Electronics Engineers Inc., Apr. 2021.
- [13] D. Q. Nguyen and V. A. Ho, “Anguilliform Swimming Performance of an Eel-Inspired Soft Robot,” *Soft Robotics*, vol. 9, pp. 425–439, June 2022.
- [14] Z. Xie, F. Yuan, J. Liu, L. Tian, B. Chen, Z. Fu, S. Mao, T. Jin, Y. Wang, X. He, G. Wang, Y. Mo, X. Ding, Y. Zhang, C. Laschi, and L. Wen, “Octopus-inspired sensorized soft arm for environmental interaction,” *Science Robotics*, vol. 8, p. eadh7852, Nov. 2023.
- [15] M. C. Doerfler, K. Schaffer, and M. M. Coad, “Hybrid Soft-Rigid Continuum Robot Inspired by Spider Monkey Tail,” in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, (Singapore, Singapore), pp. 1–6, IEEE, Apr. 2023.

- [16] Y. Zhang, D. Yang, P. Yan, P. Zhou, J. Zou, and G. Gu, "Inchworm Inspired Multimodal Soft Robots With Crawling, Climbing, and Transitioning Locomotion," *IEEE Transactions on Robotics*, vol. 38, pp. 1806–1819, June 2022.
- [17] D. Drotman, S. Jadhav, D. Sharp, C. Chan, and M. T. Tolley, "Electronics-free Pneumatic Circuits for Controlling Soft-legged Robots," *Science Robotics*, vol. 6, p. eay2627, Feb. 2021.
- [18] Y. Xu, C. Watson, J.-T. Lin, J. T. Hwang, and T. K. Morimoto, "Shape Control of Concentric Tube Robots via Approximate Follow-the-Leader Motion," *IEEE Robotics and Automation Letters*, vol. 9, pp. 7198–7205, Aug. 2024.
- [19] C. Della Santina, C. Duriez, and D. Rus, "Model-Based Control of Soft Robots: A Survey of the State of the Art and Open Challenges," *IEEE Control Systems*, vol. 43, pp. 30–65, June 2023.
- [20] C. Armanini, F. Boyer, A. T. Mathew, C. Duriez, and F. Renda, "Soft Robots Modeling: A Structured Overview," *IEEE Transactions on Robotics*, vol. 39, pp. 1728–1748, June 2023.
- [21] H. Li, L. Xun, and G. Zheng, "Piecewise Linear Strain Cosserat Model for Soft Slender Manipulator," *IEEE Transactions on Robotics*, vol. 39, pp. 2342–2359, June 2023.
- [22] J. Till, V. Aloï, and C. Rucker, "Real-time Dynamics of Soft and Continuum Robots Based on Cosserat Rod Models," *The International Journal of Robotics Research*, vol. 38, pp. 723–746, May 2019.
- [23] A. Doroudchi and S. Berman, "Configuration Tracking for Soft Continuum Robotic Arms Using Inverse Dynamic Control of a Cosserat Rod Model," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, (New Haven, CT, USA), pp. 207–214, IEEE, Apr. 2021.
- [24] R. K. Katzschmann, C. D. Santina, Y. Tshimitsu, A. Bicchì, and D. Rus, "Dynamic Motion Control of Multi-Segment Soft Robots Using Piecewise Constant Curvature Matched with an Augmented Rigid Body Model," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, (Seoul, Korea (South)), pp. 454–461, IEEE, Apr. 2019.
- [25] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators," *IEEE Transactions on Robotics*, vol. 35, pp. 124–134, Feb. 2019.
- [26] Q. Wang, Z. Hong, and Y. Zhong, "Learn to Swim: Online Motion Control of an Underactuated Robotic Eel Based on Deep Reinforcement Learning," *Biomimetic Intelligence and Robotics*, vol. 2, p. 100066, Dec. 2022.
- [27] A. Centurelli, L. Arleo, A. Rizzo, S. Tolu, C. Laschi, and E. Falotico, "Closed-Loop Dynamic Control of a Soft Manipulator Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 7, pp. 4741–4748, Apr. 2022.
- [28] G. Li, J. Shintake, and M. Hayashibe, "Deep Reinforcement Learning Framework for Underwater Locomotion of Soft Robot," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, (Xi'an, China), pp. 12033–12039, IEEE, May 2021.
- [29] Q. Ji, S. Fu, K. Tan, S. Thorapalli Muralidharan, K. Lagrelius, D. Danelia, G. Andrikopoulos, X. V. Wang, L. Wang, and L. Feng, "Synthesizing the Optimal Gait of a Quadruped Robot with Soft Actuators Using Deep Reinforcement Learning," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102382, Dec. 2022.
- [30] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges," *Robotics*, vol. 8, p. 4, Jan. 2019.
- [31] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of Machine Learning Methods in Soft Robotics," *PLOS ONE*, vol. 16, p. e0246102, Feb. 2021.
- [32] P. Schegg and C. Duriez, "Review on Generic Methods for Mechanical Modeling, Simulation and Control of Soft Robots," *Plos one*, vol. 17, no. 1, p. e0251059, 2022.
- [33] O. Gourey and C. Duriez, "Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.
- [34] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *2013 IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), pp. 3982–3987, IEEE, May 2013.
- [35] E. Coevoet, A. Escande, and C. Duriez, "Optimization-based Inverse Model of Soft Robots with Contact Handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1413–1419, 2017.
- [36] R. K. Katzschmann, M. Thieffry, O. Gourey, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, "Dynamically Closed-loop Controlled Soft Robotic Arm Using a Reduced Order Finite Element Model with State Observer," in *2019 2nd IEEE international Conference on Soft Robotics (RoboSoft)*, pp. 717–724, IEEE, 2019.
- [37] M. Thieffry, A. Kruszewski, C. Duriez, and T.-M. Guerra, "Control Design for Soft Robots Based on Reduced-order Model," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 25–32, 2018.
- [38] S. Tonkens, J. Lorenzetti, and M. Pavone, "Soft Robot Optimal Control via Reduced Order Finite Element Models," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12010–12016, IEEE, 2021.
- [39] O. Gourey, B. Carrez, and C. Duriez, "Real-time simulation for control of soft robots with self-collisions using model order reduction for contact forces," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3752–3759, 2021.
- [40] 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," Nov. 2020.
- [41] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-Driven Control of Soft Robots Using Koopman Operator Theory," *IEEE Transactions on Robotics*, vol. 37, pp. 948–961, June 2021.
- [42] 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 *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1679–1685, IEEE, 2020.
- [43] I. Mezić, "Koopman Operator, Geometry, and Learning of Dynamical Systems," *Notices of the American Mathematical Society*, vol. 68, p. 1, Aug. 2021.
- [44] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern Koopman Theory for Dynamical Systems," *SIAM Review*, vol. 64, pp. 229–340, May 2022.
- [45] S. E. Otto and C. W. Rowley, "Linearly Recurrent Autoencoder Networks for Learning Dynamics," *SIAM Journal on Applied Dynamical Systems*, vol. 18, pp. 558–593, Jan. 2019.
- [46] 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.
- [47] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.
- [48] J.-N. Juang and R. S. Pappa, "An Eigensystem Realization Algorithm for Modal Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, pp. 620–627, Sept. 1985.
- [49] H. Sharma and B. Kramer, "Preserving Lagrangian Structure in Data-Driven Reduced-Order Modeling of Large-scale Dynamical Systems," *Physica D: Nonlinear Phenomena*, vol. 462, p. 134128, 2024.
- [50] H. Sharma, I. Adibnazari, J. Cervera-Torralba, M. T. Tolley, and B. Kramer, "Data-driven Model Reduction for Soft Robots via Lagrangian Operator Inference," *IFAC-PapersOnLine*, vol. 58, no. 17, pp. 91–96, 2024.
- [51] H. Sharma, D. A. Najera-Flores, M. D. Todd, and B. Kramer, "Lagrangian Operator Inference Enhanced with Structure-preserving Machine Learning for Nonintrusive Model Reduction of Mechanical Systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 423, p. 116865, 2024.
- [52] E. D. Tytell and G. V. Lauder, "The Hydrodynamics of Eel Swimming: I. Wake structure," *Journal of Experimental Biology*, vol. 207, pp. 1825–1841, May 2004.
- [53] E. D. Tytell, "The Hydrodynamics of Eel Swimming II. Effect of Swimming Speed," *Journal of Experimental Biology*, vol. 207, pp. 3265–3279, Sept. 2004.
- [54] I. Hess and P. Musgrave, "A Continuum Soft Robotic Trout with Embedded HASEL Actuators: Design, Fabrication, and Swimming Kinematics," *Smart materials and structures*, vol. 33, no. 10, p. 105043, 2024.
- [55] I. Adibnazari, H. Sharma, M. Park, J. Cervera-Torralba, B. Kramer, and M. Tolley, "Eel-inspired Soft Robot Dynamic Structural Simulation Dataset," <https://dx.doi.org/10.21227/cyd5-t91>, 2025.
- [56] J. Cervera-Torralba, Y. Kang, E. M. Khan, I. Adibnazari, and M. T. Tolley, "Lost-Core Injection Molding of Fluidic Elastomer Actuators for the Fabrication of a Modular Eel-Inspired Soft Robot," in *2024 IEEE 7th International Conference on Soft Robotics (RoboSoft)*, pp. 971–976, 2024.
- [57] M. Park, J. Cervera-Torralba, I. Adibnazari, and M. Tolley, "Analysis on Kinematics and Propulsion of a Self-sensing Multi-DoF Undulating Robotic Fish," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.
- [58] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin,

“SOFA: A Multi-Model Framework for Interactive Physical Simulation,” in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery* (Y. Payan, ed.), vol. 11, pp. 283–321, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

- [59] E. Coevoet, T. Morales-Bieze, F. Largilliere, Z. Zhang, M. Thieffry, M. Sanz-Lopez, B. Carrez, D. Marchal, O. Goury, J. Dequidt, and C. Duriez, “Software Toolkit for Modeling, Simulation, and Control of Soft Robots,” *Advanced Robotics*, vol. 31, pp. 1208–1224, Nov. 2017.
- [60] C. Geuzaine and J.-F. Remacle, “Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre-and Post-processing Facilities,” *International journal for numerical methods in engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.
- [61] J. Lorenzetti, A. McClellan, C. Farhat, and M. Pavone, “Linear Reduced-Order Model Predictive Control,” *IEEE Transactions on Automatic Control*, vol. 67, pp. 5980–5995, Nov. 2022.
- [62] Y. Li, Y. Xu, Z. Wu, L. Ma, M. Guo, Z. Li, and Y. Li, “A Comprehensive Review on Fish-inspired Robots,” *International Journal of Advanced Robotic Systems*, vol. 19, May 2022.
- [63] J. L. Proctor, S. L. Brunton, and J. N. Kutz, “Dynamic Mode Decomposition with Control,” *SIAM Journal on Applied Dynamical Systems*, vol. 15, pp. 142–161, Jan. 2016.
- [64] J.-N. Juang, M. Phan, L. G. Horta, and R. W. Longman, “Identification of Observer/Kalman Filter Markov Parameters - Theory and Experiments,” *Journal of Guidance, Control, and Dynamics*, vol. 16, pp. 320–329, Mar. 1993.
- [65] J. P. Hespanha, *Linear Systems Theory*. Princeton university press, 2018.
- [66] P. J. Schmid, “Dynamic Mode Decomposition of Numerical and Experimental Data,” *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.
- [67] C. W. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. S. Henningson, “Spectral Analysis of Nonlinear Flows,” *Journal of fluid mechanics*, vol. 641, pp. 115–127, 2009.
- [68] R. Penrose, “A Generalized Inverse for Matrices,” in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, pp. 406–413, Cambridge University Press, 1955.
- [69] H. Sharma, J. D. Draxl Giannoni, and B. Kramer, “Nonlinear Energy-preserving Model Reduction with Lifting Transformations that Quadraticize the Energy,” *Physica D: Nonlinear Phenomena*, vol. 483, p. 134954, 2025.



**Iman Adibnazari** (Member, IEEE) is a Ph.D. candidate in the Department of Mechanical and Aerospace Engineering at the University of California San Diego, CA, USA. Previously he received the B.S. degree in electrical engineering from the University of Utah, Salt Lake City, UT, USA, in 2018 and M.S. in electrical and computer engineering from the University of California San Diego, CA, USA, in 2020.

His research interests include design and fabrication of robotic systems, data-driven methods for modeling and control, and robot ethics.

He was recognized as a Science Policy Fellow from 2022-2025 from the School of Global Policy and Strategy at the University of California San Diego.



**Harsh Sharma** (Member, IEEE) is an Assistant Professor in the Department of Mechanical Engineering at the University of Wisconsin–Madison, WI, USA. Prior to this, he was a Postdoctoral Research Scholar at the University of California, San Diego, CA, USA. He received the Ph.D. degree in Aerospace Engineering and the M.S. degree in Mathematics from Virginia Tech, Blacksburg, VA, USA, and the B.Tech. and M.Tech. degrees in Mechanical Engineering from the Indian Institute of Technology Bombay, Mumbai, India.

His research interests include reduced-order modeling, scientific machine learning, and structure-preserving methods for the design, analysis, and control of complex dynamical systems, with applications in computational physics, structural dynamics, and soft robotics.



**Myungsun Park** (Member, IEEE) is a Post-doctoral researcher at University of California San Diego. She received the Ph.D. degree in 2024 from Seoul National University, Korea.

Her research interests include hand motion tracking, control of robot hands, development of soft sensors for proprioceptive, tactile and environmental sensing, and underwater robotic systems.



**Jacobo Cervera-Torralba** (Graduate Student Member, IEEE) received the B.Sc. degree in aerospace engineering from the Polytechnic University of Valencia, Valencia, Spain, in 2021. He is currently working toward the Ph.D. degree in design and fabrication of bio-inspired soft robots for underwater exploration with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, San Diego, CA, USA.

His research focuses on the design and fabrication of bioinspired soft robots for underwater exploration.

Mr. Cervera-Torralba was the recipient of the Charles Lee Powell Fellowship 2021.



**Boris Kramer** (Member, IEEE) received the M.S. and Ph.D. degrees in mathematics from Virginia Tech, Blacksburg, Virginia, in 2011 and 2015, respectively.

He is an Associate Professor in Mechanical and Aerospace Engineering with the University of California San Diego, San Diego, CA, USA. From 2015 to 2019, he has been a Postdoctoral Scholar with the Massachusetts Institute of Technology, Cambridge, MA, USA. His main research interests include model reduction, high-dimensional control,

data driven modeling, (multifidelity) uncertainty quantification, and design under uncertainty.

Dr. Kramer is a Member of SIAM and a Senior Member of AIAA. He was the recipient of the National Science Foundation Early CAREER Award in Dynamics, Control, and System Diagnostics in 2022 and the Department of Defense Newton Award in 2020. He is currently an Associate Editor for the SIAM/ASA Journal on Uncertainty Quantification.



**Michael T. Tolley** (Senior Member, IEEE) received the Ph.D. and M.S. degrees in mechanical engineering with a minor in computer science from Cornell University, Ithaca, NY, USA, in 2009 and 2011, respectively.

He is currently a Professor in Mechanical and Aerospace Engineering, and the Director of the Bioinspired Robotics and Design Lab, Jacobs School of Engineering, University of California San Diego (UCSD), San Diego, CA, USA. Before joining the Mechanical Engineering faculty with UCSD in the

fall of 2014, he was a Postdoctoral Fellow with the Wyss Institute for Biologically Inspired Engineering, Harvard University, Cambridge, MA, USA. His research seeks inspiration from nature to design robotic systems with the versatility, resilience, and efficiency of biological organisms.

Dr. Tolley's work has appeared in leading academic journals including *Science* and *Nature*, and has been recognized by awards including a U.S. Office of Naval Research Young Investigator Program award and a 3 M Nontenured Faculty Award.