Controlling the Shape of Soft Robots Using the Koopman Operator

Ajai Singh, Jiefeng Sun, and Jianguo Zhao

Abstract— **In nature, animals with soft body parts can control the parts to different shapes, e.g., an elephant trunk can wrap on a tree branch to pick it up. But most research on manipulators only focuses on how to control the end effector, partly because the arm of the manipulator is rigidly articulated. With recent advances in soft robotics research, controlling a soft manipulator into many different shapes will significantly improve the robot's functionality, such as medical robots morphing their shape to navigate the digestive system and then delivering drugs to the required location. However, controlling the shape of soft robots is challenging since the dynamics of soft robots are highly nonlinear and computationally intensive. In this paper, we leverage a physics-informed data-driven method using the Koopman operator to realize shape control of soft robots. The dynamics of a soft manipulator are simulated using a physics-based simulator (PyElastica) to generate the input-output data, and the data is used to identify an approximated linear model based on the Koopman operator. We then formulate the shape-control problem as a convex optimization problem that is computationally efficient. We demonstrated the linear model is over 12 times faster than the physics-based model in simulating the manipulator's motion. Further, we can control a soft manipulator into different shapes using model predictive control (MPC). We envision that the proposed method can be effectively used to control the shapes of soft robots to interact with uncertain environments or the shapes of shape-morphing robots to fulfill different tasks.**

I. Introduction

Despite the fact that robots made from rigid bodies are the heart of various industries such as manufacturing, soft robots made from soft materials have recently emerged in robotics research [1], [2]. Unlike rigid ones, soft robots can leverage their inherent mechanical compliance to interact with humans or external environments, leading to many applications such as grasping or manipulation, locomotion, rehabilitation, assistance, medical devices, etc [3], [4].

Recently, the soft robotics community has started to investigate how a robot's shape can enhance the functional capabilities of the robot with inspiration from biological organisms [5]. In fact, various living organisms can change their body shape to cope with different environments and response to external stimuli. For example, an octopus can squeeze its body through gaps much smaller than body size [6], and moth larvae can curl up to roll away from predators [7]. Inspired by biological organisms, researchers have developed various robots to leverage different shapes for different functions. For instance, Shah *et al.* investigated how a soft robot can use different shapes for either crawling or rolling in different environments [8]. Hwang *et al.* leveraged a novel kirigami composite to develop a morphing drone that can autonomously transform from ground to air vehicle [9]. Many other recent research on how to shape morphing can enhance a robot's functionality is reported in the review paper [5].

To leverage different shapes to enhance functions, we need to control the shape of a soft robot to the desired ones. But it is challenging to control a soft robot's shape since soft robots exhibit highly nonlinear dynamics. Researchers have developed various physics-based models using different methods such as Cosserat Rod theory [10], and model reduction method [11], among many others [12]. Although such models can achieve high-fidelity simulation of various types of soft robots [13], they generally require a long computational time, making them unsuitable for the shape control of soft robots.

In this paper, we aim to leverage existing physics-based models to obtain computationally efficient data-driven models and then use the resulting model for controlling the shape of soft robots. Specifically, we will use the PyElastica [10] simulation software to generate sufficient input-output data for a given soft robot. Using the data, we will then establish a data-driven model based on Koopman operator theory [14] to obtain a finite-dimensional approximation of a soft robot [15]. The Koopman operator can represent a nonlinear dynamical system with a finite-dimensional linear model to approximate the original dynamics of a soft robot. With such a linear model, we can directly use existing control methods such as model predictive control to control a soft robot's shape.

Note that researchers have recently used the Koopman operator theory to control soft robots [15]–[19]. The work in [15] shows promising results in controlling the tip of the robot to trace a desired trajectory while [16] used the Koopman operator approach in modeling of soft robotic swimmer. But controlling the shape of soft robots is different from existing problems (e.g., tip position control). Therefore, our work adds another application of Koopman operator-based system identification and control to control the shape of soft robots. In other words, *the contribution of this paper* is to develop a data-driven method to control a soft robot's shape by leveraging existing physics-based models and Koopman operator theory.

The rest of the paper is organized as follows. In section II, we formulate the shape control problem for a soft robot. In Section III, we provide the mathematical underpinnings of the Koopman operator, its approximation from data, and the algorithm used for system identification using the Koopman

Ajai Singh, Jiefeng Sun, and Jianguo Zhao are with the Department of Mechanical Engineering, Colorado State University, Fort Collins, CO 80523, USA. `Ajai.Singh@colostate.edu`, `J.Sun@colostate.edu`, `Jianguo.Zhao@colostate.edu`

operator approach. In section IV, we describe the Model Predictive Controller used with the identified linear system. Section V describes the simulation setup: how to generate the input-output data for a soft robot using the PyElastica [10]. Section VI shows the results of system identification and shape control.

II. Shape Control Problem

In this section, we formulate the shape control problem by using a general soft manipulator. We will show how to use this framework to solve the shape control problem in subsequent sections.

Given a soft manipulator of length L, we divide it into $N-1$ segments with equal length. The shape for the i-th ($i = 2, \ldots, N$) segment is specified by section at its top. At a discrete time step t_k, we use $g_i(t_k) \in SE(3)$ to represent the top section's position and orientation in the inertia frame Figure 1.

$$g_i(t_k) = \begin{bmatrix} R_i(t_k) & p_i(t_k) \\ 0 & 1 \end{bmatrix} \tag{1}$$

where $R_i(t_k) \in SO(3)$ represents the orientation and $p_i(k) \in \mathbb{R}^3$ presents the position for the section's centroid. Denote the i-th segment as L_i. For each segment, we assume we can apply input u in terms of forces/torques at each segment's top. In reality, such forces/torques may be generated by artificial muscles embedded inside soft materials [20]. With such a setup, the shape of the soft manipulator can be approximated by $g_i(t_k) \in SE(3)$ at time step t_k.

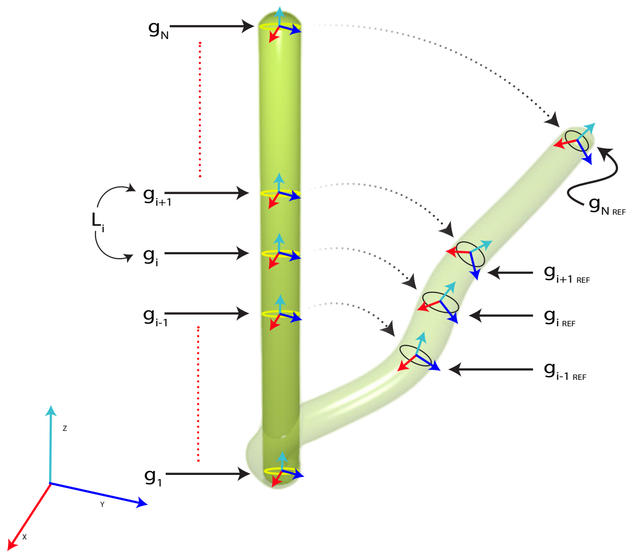

Fig. 1. Illustration for the shape control problem for a soft manipulator divided into $N-1$ equal length segments with the top of each segment shown as a yellow cross-section. The manipulator shown in solid green color is its initial shape ($t = 0$) and the shape shown in faded green shows the target shape.

Given a desired shape for the soft manipulator represented by $g_{i_{ref}}$ ($i = 1, \ldots, N-1$), the shape control problem can be formulated as finding the control input $u(t_k)$ that minimizes the distance between $g_i(t_k)$ and $g_{i_{ref}}$. Note that the distance in $SE(3)$ can be defined separately for the position and orientation with the Euclidean distance for position and geodesic distance for orientation. In this paper, we will focus on a simplified problem by only considering the position distance. In this case, the problem can be formulated as:

$$\underset{u(t_k)}{\text{minimize}} \quad \sum_{i=1}^{N-1} ||p_{ref} - p_i(t_k)||_2^2 \tag{2a}$$

$$\text{subject to}$$

$$x(t_{k+1}) = f(x(t_k), u(t_k)), \tag{2b}$$

$$h(x(t_k), u(t_k)) \leq 0 \tag{2c}$$

where $p_{ref} \in \mathbb{R}^3$ is the desired position, $x(t_k) \in \mathbb{R}^n$, $u(t_k) \in \mathbb{R}^m$ are the states and control input of the system at time instant t_k, respectively. $h(x(t_k), u(t_k)) \leq 0$ are the various constraints applied to the state and control variables which are commonly known as polyhedral constraints.

Generally, the dynamics for a given soft robot (i.e., $x(t_{k+1}) = f(x(t_k), u(t_k))$) is highly nonlinear, involving complicated physics-based models [12]. Such models can only be solved numerically with considerable computation time, preventing them from real-time shape control of soft robots. Inspired by recent work on using the data-driven method to identify approximated models from either numerical or experimental data for controlling soft robots for manipulation [15]–[19], we aim to obtain a data-driven model using Koopman operator theory and then use the model to *control the shapes* of soft robots.

III. System Identification using Koopman Operator Theory

Given the complicated dynamics of a soft robot, we will use Koopman Operator theory to directly identify a computationally efficient linear model using the input-output data generated by physics-based models (e.g., PyElastica [10]). In this section, we briefly review the preliminaries for Koopman operators.

Koopman operator theory can be used to construct a linear model of a forced nonlinear system in an infinite dimensional Hilbert space from input-output data of the nonlinear system. With the constructed linear model, we can directly use existing linear system control techniques. The Koopman operator approach is undeniably becoming an increasingly popular data-driven method for the control of nonlinear dynamical systems. The theoretical underpinnings of the Koopman Operator were formulated by Bernard Koopman in [14], and this framework became popular after the development of the Dynamic Mode Decomposition algorithm [21].

Given a nonlinear dynamical system, the Koopman operator first maps the states of the original system using scalar functions (also called observables) of the states into a so-called lifted space with new state variables. The new system in the lifted space with the new state variables is an infinite dimensional linear system. Unlike the linearization about a point that becomes inaccurate when operating away from the linearizing point, the Koopman operator describes the evolution of the scalar observable throughout the state space in a linear fashion. This makes the Koopman operator approach preferable when realizing linear representation of nonlinear systems [22].

We briefly review the Koopman operator framework for control systems, as described in [15]. Assume a discrete nonlinear dynamical system given by:

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k)) \\ y(t_k) &= g(x(t_k)) \end{aligned} \tag{3}$$

where $x(t_k)$, $x(t_{k+1}) \in \mathbb{R}^n$ are the state vector at time step t_k, t_{k+1} respectively, $y(t_k) \in \mathbb{R}^r$ is the output of the system at time instant t_k. To simplify notations, in the following, we use $x(t_k)$ and x_{t_k} interchangeably.

To map the state x to a lifted space, we use a basis or observation function $\phi(x(t_k)) : \mathbb{R}^n \to \mathbb{R}^c \in \mathcal{F}$, where $\mathcal{F}$ is the space of all basis functions. The Koopman Operator $\mathcal{K} : \mathcal{F} \to \mathcal{F}$ is defined as:

$$(\mathcal{K}\phi)(x_{t_k}) = \phi(f(x_{t_k})) \tag{4}$$

which can be rewritten as:

$$(\mathcal{K}\phi)(x_{t_k}) = \phi(x_{t_k+1}) \tag{5}$$

which means the Koopman operator simply updates the observation of the state in the lifted space from the current time step to the next step.

$\mathcal{K}$ is an infinite dimensional linear operator, but we can use a finite subspace to approximate it. Let the finite dimensional approximation of $\mathcal{K}$ be $\bar{\mathcal{K}}$. $\bar{\mathcal{K}}$ operates on $\bar{\mathcal{F}} \subset \mathcal{F}$ which is the subspace spanned by a finite set of basis functions. $\bar{\mathcal{K}}$ can be obtained using EDMD as discussed in [23], and this approximation is achieved by solving the following optimization problem

$$\underset{A}{\text{minimize}} \quad \sum_{k=0}^{K-1} ||\psi(x_{t_{k+1}}) - A\psi(x_{t_k})||_2^2 \tag{6}$$

where $\psi(x) = [\psi_1(x) \ , \psi_2(x) \ , \psi_3(x) \ , \dots , \ \psi_{N_c}(x)]^\top$ with $\{\psi_i : \mathbb{R}^n \to \mathbb{R}\}_{i=1}^{N_c}$ represents the N_c basis functions, $A \in \mathbb{R}^{N_c \times N_c}$ is the finite dimensional approximation of the Koopman Operator, N_c is the total number of basis functions, K is the cardinality of the dataset given by $\mathcal{D} = \{x_{t_k}\}_{k=0}^{K}$ and $^\top$ is the transpose operator.

Solving the minimization problem of (6), we can represent the nonlinear dynamical system given by (3) as the following discrete linear dynamical system

$$\begin{aligned} z(t_{k+1}) &= Az(t_k) \\ \tilde{y}(t_k) &= Cz(t_k) \end{aligned} \tag{7}$$

where $z(t_k) = \psi(x(t_k)) \in \mathbb{R}^{N_c}$ and $\tilde{y}(t_k)$ is the output. The matrix $C \in \mathbb{R}^{r \times N_c}$ is obtained just like A by solving the following minimization problem

$$\underset{C}{\text{minimize}} \quad \sum_{k=1}^{K} ||y(t_k) - C\psi(x(t_k))||_2^2 \tag{8}$$

Similarly, for a nonlinear dynamical system with control inputs, the methodology discussed can be used. Consider a discrete nonlinear dynamical system given by

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k), u(t_k)) \\ y(t_k) &= g(x(t_k)) \end{aligned} \tag{9}$$

where $u(t_k) \in \mathbb{R}^m$ where m is the dimension of control inputs of the dynamical system. Then in this case to approximate the Koopman Operator, the minimization problem in (6) changes to

$$\underset{A, B}{\text{minimize}} \quad \sum_{k=0}^{K-1} ||\psi(x(t_{k+1})) - (A\psi(x(t_k)) + Bu(t_k))||_2^2 \tag{10}$$

Thus, by solving the minimization problem of (10), the finite approximation of the Koopman operator is approximated by *A* and *B*, and it acts as one step predictor of the nonlinear dynamical system described by (9). The minimization problem for the output equation i.e. for *C* matrix remains the same as given by (8). It is advisable to include velocities in the states when identifying the dynamics of a mechanical system [15]. These can be included by modifying the domain of the basis function such that $\{\psi_i : \mathbb{R}^{n+nd+md} \to \mathbb{R}\}_{i=1}^{N_c}$, where d is the number of delays.

We will use the Extended Dynamic Mode Decomposition (EDMD) [15], [23] to construct the linear model of a soft robot since EDMD is a data-driven method that approximates the leading Koopman eigenfunctions, eigenvalues and modes where as other data-driven methods such as generalized Laplace analysis, Ulam Galerkin method, and Dynamic Mode Decomposition cannot approximate the three quantities [23]. One could also use neural networks to approximate the three quantities but neural networks require a lot of training data and tuning.

IV. Model Predictive Control for Koopman operator based linear system

In this section, we introduce the basic idea of model predictive control and then develop on the concept to show how MPC can be used alongside Koopman Operator.

Many model-based controllers have been developed for soft robots [24], [25], but most of them rely on simplifying assumptions and mostly are static. Such controllers have been proven to be very efficient for the static control of soft robotic manipulators. The disadvantage of these controllers is that they are only limited to static control and they are not suitable for dynamic control of complex soft robotic systems. Dynamic control of soft robots is achieved by supplementing a piece-wise constant curvature model with data-driven trajectory optimization but again the downside of this approach is that the training is very task-specific [15]. There also have been some more realistic physics-based models, but they are computationally very expensive.

From Section III, we know that the Koopman operator approximates a linear system of a nonlinear dynamical system from data. In [15], they constructed MPC controller from a linear Koopman representation of a nonlinear dynamical system. We use a similar approach demonstrated in [15] to construct a MPC for shape control of a soft manipulator. Since the identified model is linear, the MPC optimization has computational advantages over nonlinear ones as the MPC optimization problem is convex whereas, for the non-linear models, it is not. Since the optimization problem is convex, it can be solved very efficiently with any method for convex optimization. For Koopman-based MPC, we first define the objective function as follows:

$$J = z(t_{N_h})^\top Q(t_{N_h}) z(t_{N_h}) + q(t_{N_h})^\top z(t_{N_h}) + \sum_{i=0}^{N_h-1} \{ z(t_i)^\top Q(t_i) z(t_i) + u(t_i)^\top R(t_i) u(t_i) + q(t_i)^\top z(t_i) + r(t_i)^\top u(t_i) \}$$

where $N_h \in \mathbb{N}$ is the prediction horizon, $Q(t_i) \in \mathbb{R}^{N_c \times N_c}$, $R(t_i) \in \mathbb{R}^{m \times m}$ are positive semidefinite matrices, $q(t_i) \in \mathbb{R}^{N_c}$, and $r(t_i) \in \mathbb{R}^m$. Here Q, R, q, r are constant matrices or vectors. Then we can iteratively solve a convex quadratic program over a receding horizon as shown below:

$$\underset{u(t_k)}{\text{minimize}} \quad J \tag{11a}$$

subject to

$$z(t_{k+1}) = Az(t_k) + Bu(t_k), \tag{11b}$$

$$E(t_i)z(t_i) + F(t_i)u(t_i) - b(t_i) \leq 0, \tag{11c}$$

$$z(0) = \psi(x(t_k)) \tag{11d}$$

where $E(t_i) \in \mathbb{R}^{c \times N_c}$ and $F(t_i) \in \mathbb{R}^{c \times m}$ and the vector $b(t_i) \in \mathbb{R}^c$ define state and input polyhedral constraints where c denotes the number of constraints. Every time the optimization routine is called the predictions need to be set to the current lifted state $\psi(x(t_k))$. While the size of the cost and constraint matrices depend on the dimension of the lifted state N_c, [26] shows that these can be rendered independent of N_c by transforming the problem into its so-called dense-form [15]. We use the above framework to solve the shape control problem proposed in section II.

V. Simulation Setup

In this paper, we use the PyElastica, a physics-based simulator for soft robots based on the Cosserat rod theory, to generate the data for the Koopman operator. We choose PyElastica because it is relatively accurate, open source, and easy to set up.

A. Generating data from PyElastica

When using PyElastica, we need to set up a simulation where the user is required to define a system of rods, set up initial and boundary conditions on the rod, run the simulation, and collect data for post-processing. The detailed process can be found in [27].

The physical parameters defined for our manipulator are listed in Table I. We fix the base of the robot as the boundary condition. The actuation of the manipulator is achieved by applying torques distributed along the length of the arm. The torques are decomposed into orthogonal torque functions in the local normal and bi-normal directions. The magnitude of the torques in a direction is obtained via continuous splines characterized by N independent control points. In our case, we don't apply the torque in the tangent (axial) direction to twist the robot.

TABLE I
List of physical parameters for the simulation setup

Physical Parameter	**Value**	**Unit**
Number of tracking points	6	N/A
Starting Position of the rod (vector)	[0.0, 0.0, 0.0]	N/A
Direction in which rod extends(vector)	[0.0, 0.0, 1.0]	N/A
Normal vector of rod	[1.0, 0.0, 0.0]	N/A
Length of rod	1.00	meter (m)
Radius of the tip	0.05	meter (m)
Radius of the base	0.05	meter (m)
Density of the rod	1.0×10^3	kg/m^3
Energy dissipation constant	10.00	N/A
Youngs Modulus	1.00×10^7	Pa
Poisson's Ration	0.50	N/A

*The vectors are defined with respect to the standard right-hand coordinate system.

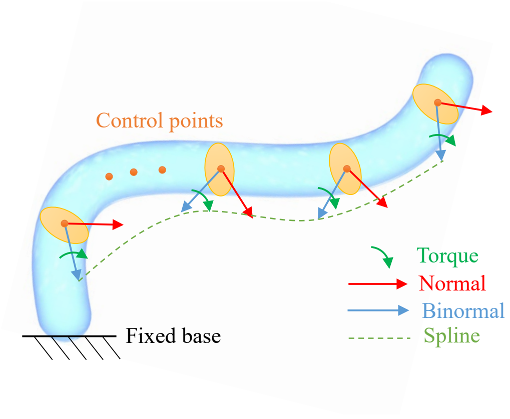

Fig. 2. The schematic for the simulation setup.

Note that spline control points for normal and binormal directions are different, and two splines are needed. To generate the two splines for each simulation case, we randomly generate two torques in normal and binormal directions for each control point. In the initial state, the robot is in a straight and upright shape. In the simulation, the two torque splines are kept constant as a step input for the system. We use the position Verlet algorithm as the time stepping algorithm and time step $\Delta t = 0.0001$ s to simulate 20 s for each case, which can ensure the soft robot reaches steady state final shape.

B. Data collection for system identification

Koopman Operator can be used to construct a linear model of the soft robotic system constructed in section III. To do this, we collect data for system identification by simulating the robot with random input.

We simulate the manipulator for a total of 30 cases, For each simulation case, we collect 2000 snapshots with a sampling time of $T_s = 0.01$ s. These data sets are used for approximating the Koopman Operator. The system identification was carried out by lifting the collected snapshots using the basis function with delays defined in section III. We used first-order polynomials with delay $d = 1$ as the basis function and then performed least square regression as shown in (10) and (8) to obtain the A, B, and C matrices, respectively [15] with $A \in \mathbb{R}^{47\times47}$, $B \in \mathbb{R}^{47\times10}$, and $C \in \mathbb{R}^{18\times47}$. For the problem, we have chosen a total of 47 basis functions with degree 1. For the 47 functions, we defined the first 18 to be the output of the system, then 28 are the delay coordinates, and a constant 1 has been added so that we do not lift the inputs. One might simply think to add higher degree polynomials or more number of basis functions to minimize the prediction error by the Koopman-based linear model $\dot{\phi}(x) = A\phi(x)$, but with the increase in the order of the polynomial, there are more functions in ϕ, then more derivatives $\dot{\phi}$ have to be expressed by ϕ. As a consequence of increasing the order of the polynomials the derivatives $\dot{\phi}$ grow in complexity which makes it harder for $\dot{\phi}$ to be expressed by ϕ [28].

VI. Results

In this section, we first quantify the identified linear model in terms of accuracy and speed by comparing it with the physics-based model. We demonstrate that it can realize much faster tip and shape control with the identified linear model and an MPC.

A. *The Accuracy and Speed of the Identified Linear Model*

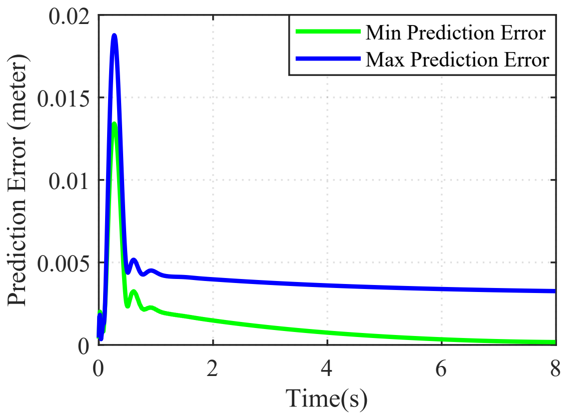

Fig. 3. The maximum prediction error (in blue) and minimum prediction by the Koopman based linear model (in green). The error at each time-step is called using (12).

The accuracy of the Koopman Model is estimated by calculating the error that is defined as the Euclidean distance between the predicted output and the output of the physics-based model. The accuracy of the model approximated by the Koopman operator depends on the number of basis functions and the types of basis functions. We validated the identified linear model against 6 different data sets generated with the same method as mentioned above.

The linear model predicted by the Koopman operator has a maximum mean RMSE of 35E-4 meters and minimum mean RMSE of 6.78E-4 across all states. Fig. 3 shows the maximum and minimum mean prediction error between the Koopman-based linear model and Physics based model. The prediction error for other validation cases lies between the maximum mean and minimum mean error mentioned. Here the mean prediction error at time step t_i is defined as:

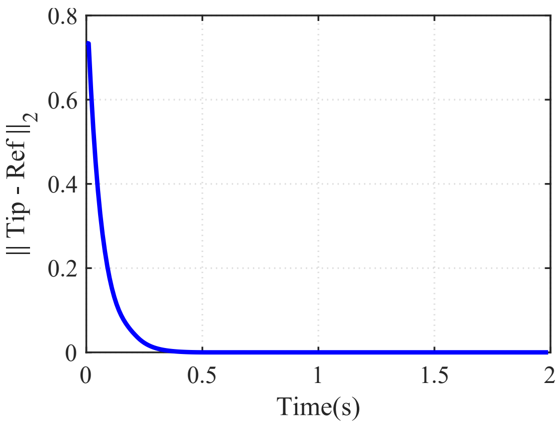

Fig. 4. Results for the control of the tip of the manipulator. The plot shows the Euclidean distance (in meters) between the tip and the reference set point. Here $||.||_2$ denotes the Euclidean norm.

$$Error_{mean}(t_i) = \frac{1}{L}\sqrt{\frac{1}{N_x}\sum_{j=1}^{N_x}((x^j(t_i) - x^{j_{ref}}(t_i))^2)} \tag{12}$$

where $L = 1$ is the length of the soft manipulator in meter, $N_x = 18$ is the number of states, $x^j(t_i)$, $x^{j_{ref}}(t_i)$ is the j^{th} state vector at time t_i approximated by Koopman based model and actual physics-based model, respectively.

For the computation time, the Koopman-based linear model is much more time efficient compared to its Physics-based counterpart. To ignore the effect of other operating system tasks interfering with simulation we measure the mean time. For the Koopman-based model the mean time was calculated by running the model 7 times and each run consisting of 100 loops. Similarly, the physics-based model was 7 times and each run consisted of 10 loops, and $dt = 10^{-4}$ was fixed when running the time performance test. The time comparison tests are run on a computer having 16 GB Random Access Memory (RAM) and a 2.6 GHz CPU. Table II shows the comparison between the mean time taken by both the models to run for a finite number of time steps. We can see the Koopman-based linear model is much faster compared to the actual physics-based model: over 12 times faster except in the case of the single step.

Controlling the tip of the soft robot is a very special case of the shape control problem that we proposed in section II. Unlike controlling the shape of the soft robot where we are required to control multiple points, we control only one point, i.e., the tip of the manipulator. The problem of controlling the tip of the soft manipulator can also be called as set point tracking in three-dimensional space. We use the linear Model Predictive Controller with a prediction horizon

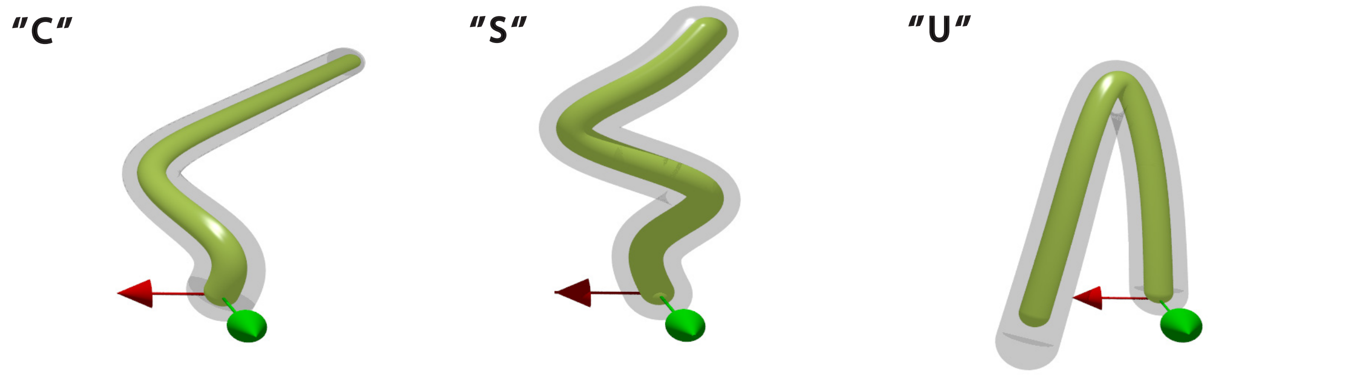

Fig. 5. Results for controlling the soft robot to three different shapes. The object in solid green is the shape of the actual soft robot and the gray envelope is the reference shape. For reference X axis has been color coded as Green, similarly, the Y axis has been color-coded as Red.

TABLE II
MEAN TIME COMPARISON OF PHYSICS AND KOOPMAN-BASED MODEL

# Time steps	Physics-based	Koopman-based
1	1.7 ms ± 353 µs	188 µs ± 29.5 µs
10	2.21 ms ± 112 µs	159 µs ± 50.7 µs
100	9.14 ms ± 386 µs	696 µs ± 23.5 µs
1000	60.8 ms ± 1.13 ms	4.75 ms ± 111 µs
10000	568 ms ± 7.2 ms	44.6 ms ± 640 µs
100000	5.6 s ± 49.3 ms	435 ms ± 5.79 ms

* The time shown is the mean time per loop.
* For Koopman-based model the mean is over 7 runs, 100 loop each
* For Physics-based model the mean is over 7 runs, 10 loop each

$N_h = 25$ steps. The desired reference set points as [0.1, 0.2, 0.3] shown in section IV. To demonstrate the control of tip, we move the tip of the soft manipulator from the rest position, i.e., from [0, 0, 1] to [0.1, 0.2, 0.3] where the elements of the vector are the x, y, z coordinates (in meter) of the manipulator. The result is shown in Figure 4, which shows the Euclidean distance between the tip of the robot and the reference point with time. From the plot is clear that the MPC controller can move the tip to the desired location within 0.5 s.

B. Shape Control with Koopman-based MPC

We further demonstrate the shape control of a soft manipulator by controlling it to different shapes, specifically three letters: 'C', 'S', and an inverted 'U'. The linear Model Predictive Controller is used to derive the optimal control input over the prediction horizon of $N_h = 25$. For different shapes, the controller has a similar cost function and no input or state polyhedral constraint. The objective of the controller is to move the points being tracked to the desired reference location in the workspace of the robot. Hence a cost function is chosen in such a way that it penalizes the distance between the reference point and points on the robot's body. To generate the reference or desired shapes, we use the shooting method with the physics-based model. Then these shapes are supplied as the reference shapes to the MPC

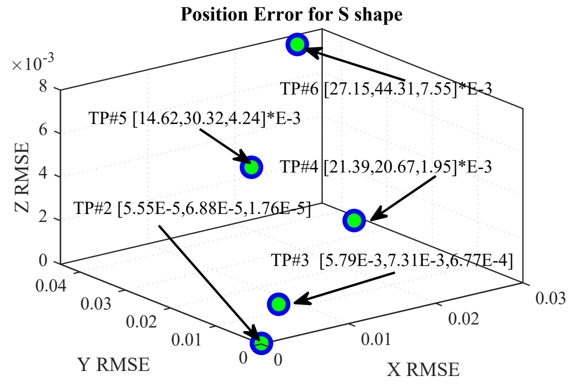

Fig. 6. Position error of the tracking points corresponding to the U shape acquired by the soft manipulator as shown in Figure 5. The term TP in the plot stands for Tracking Point and the vector following TP shows RMSE for X, Y, and Z for a particular tracking point. Here the Root Mean Squared Error (RMSE) is measured in meters. (Note that the error for Tracking point 1 was always 0 as it was static, hence it was ignored in the error plot.)

controller. Note that since the bottom of the manipulator is rigidly fixed to the ground, the segment close to the ground needs to resemble a vertical shape due to the spline used to interpolate the torques in PyElastica. If we do not consider this segment, however, the desired shapes are indeed close to letters 'C' and 'S'.

The results for the three shapes are illustrated in Fig. 5, where we plot the robot's final shape and the desired shape. From the figure, we can see the robot can accomplish the desired shape. To quantify the error between the final and desired shape, we plot the RMSE in meters for different tracking points for the morphed shapes in Figure 6. Root Mean Squared Error is calculated as the RMSE between the final position generated by the controller for a tracking point and the corresponding reference point. As we can see from the error in Figure 6, the final position error can be quite small ($< 5\%$ with respect to the manipulator's length).

VII. CONCLUSION

In this paper, we have used a data-driven method based on the Koopman operator to establish an approximated linear model of a soft manipulator using the input-output data from the physics-based model that is accurate but not computationally efficient. The linear model is accurate and 12 times faster than the physics-based model. Combining with a linear Model Predictive Controller (MPC), we successfully control the shape of the soft robot, which is very difficult for a physics-based model.

This approach shows promising results, but significant improvement can be made. For example, a much more accurate model approximation can be obtained if the eigenfunctions of the Koopman Operator can be approximated from data or we can approximate the robot with more number of tracking points. In the future, we will extend this method to control a soft robot made from a combination of several rods to form a polyhedron (e.g., tetrahedron, Octahedron, etc.) to generate more diverse shapes, which can be potentially experimentally verified through soft robot prototypes driven by artificial muscles.

References

[1] G. M. Whitesides, "Soft robotics," *Angewandte Chemie International Edition*, vol. 57, no. 16, pp. 4258–4273, 2018.

[2] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, 2015.

[3] M. Cianchetti, C. Laschi, A. Menciassi, and P. Dario, "Biomedical applications of soft robotics," *Nature Reviews Materials*, vol. 3, no. 6, pp. 143–153, 2018.

[4] P. Polygerinos, N. Correll, S. A. Morin, B. Mosadegh, C. D. Onal, K. Petersen, M. Cianchetti, M. T. Tolley, and R. F. Shepherd, "Soft robotics: Review of fluid-driven intrinsically soft devices; manufacturing, sensing, control, and applications in human-robot interaction," *Advanced Engineering Materials*, vol. 19, no. 12, p. 1700016, 2017.

[5] D. Shah, B. Yang, S. Kriegman, M. Levin, J. Bongard, and R. Kramer-Bottiglio, "Shape changing robots: bioinspiration, simulation, and physical realization," *Advanced Materials*, vol. 33, no. 19, p. 2002882, 2021.

[6] K. J. Quillin, "Kinematic scaling of locomotion by hydrostatic animals: ontogeny of peristaltic crawling by the earthworm lumbricus terrestris," *Journal of Experimental Biology*, vol. 202, no. 6, pp. 661–674, 1999.

[7] R. Armour and J. Vincent, "J bionic eng. 2006, 3, 195-208; c) l. van griethuijsen, b. trimmer," *Biol. Rev*, vol. 89, pp. 656–670, 2014.

[8] D. S. Shah, J. P. Powers, L. G. Tilton, S. Kriegman, J. Bongard, and R. Kramer-Bottiglio, "A soft robot that adapts to environments through shape change," *Nature Machine Intelligence*, vol. 3, no. 1, pp. 51–59, 2021.

[9] D. Hwang, E. J. Barron III, A. T. Haque, and M. D. Bartlett, "Shape morphing mechanical metamaterials through reversible plasticity," *Science Robotics*, vol. 7, no. 63, p. eabg2171, 2022.

[10] N. Naughton, J. Sun, A. Tekinalp, T. Parthasarathy, G. Chowdhary, and M. Gazzola, "Elastica: A compliant mechanics environment for soft robotic control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3389–3396, 2021.

[11] O. Goury 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.

[12] G. Mengaldo, F. Renda, S. L. Brunton, M. Bächer, M. Calisti, C. Duriez, G. S. Chirikjian, and C. Laschi, "A concise guide to modelling the physics of embodied intelligence in soft robotics," *Nature Reviews Physics*, pp. 1–16, 2022.

[13] J. Sun and J. Zhao, "Modeling and simulation of soft robots driven by embedded artificial muscles: an example using twisted-and-coiled actuators," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 2911–2916.

[14] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.17.5.315

[15] 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, no. 3, pp. 948–961, 2020.

[16] 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)*. IEEE, 2020, pp. 1679–1685.

[17] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional koopman operators for model-based control," *arXiv preprint arXiv:1910.08264*, 2019.

[18] D. A. Haggerty, M. J. Banks, P. C. Curtis, I. Mezić, and E. W. Hawkes, "Modeling, reduction, and control of a helically actuated inertial soft robotic arm via the koopman operator," *arXiv preprint arXiv:2011.07939*, 2020.

[19] E. Kamenar, N. Ćrnjarić-Žic, D. Haggerty, S. Zelenika, E. W. Hawkes, and I. Mezić, "Prediction of the behavior of a pneumatic soft robot based on koopman operator theory," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 2020, pp. 1169–1173.

[20] J. Sun, B. Tighe, Y. Liu, and J. Zhao, "Twisted-and-coiled actuators with free strokes enable soft robots with programmable motions," *Soft robotics*, vol. 8, no. 2, pp. 213–225, 2021.

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

[22] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016.

[23] 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.

[24] C. Della Santina, A. Bicchi, and D. Rus, "On an improved state parametrization for soft robots with piecewise constant curvature and its use in model based control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1001–1008, 2020.

[25] 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," *arXiv preprint arXiv:2110.01358*, 2021.

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

[27] X. Zhang, F. Chan, T. Parthasarathy, and M. Gazzola, "Modeling and simulation of complex dynamic musculoskeletal architectures," *Nature Communications*, vol. 10, no. 1, pp. 1–12, 2019. [Online]. Available: https://doi.org/10.1038/s41467-019-12759-5

[28] V. Cibulka, T. Haniš, and M. Hromčík, "Data-driven identification of vehicle dynamics using koopman operator," in *2019 22nd International Conference on Process Control (PC19)*. IEEE, 2019, pp. 167–172.