Abstract—Controlling soft robots with precision is a challenge due in large part to the difficulty of constructing models that are amenable to model-based control design techniques. Koopman operator theory offers a way to construct explicit linear dynamical models of soft robots and to control them using established model-based linear control methods. This method is data-driven, yet unlike other data-driven models such as neural networks, it yields an explicit control-oriented linear model rather than just a “black-box” input-output mapping. This work describes this Koopman-based system identification method and its application to model predictive controller design. A model and MPC controller of a pneumatic soft robot arm is constructed via the method, and its performance is evaluated over several trajectory following tasks in the real-world. On all of the tasks, the Koopman-based MPC controller outperforms a benchmark MPC controller based on a linear state-space model of the same system.

I. INTRODUCTION

Soft robots have bodies made out of intrinsically soft and/or compliant materials. This inherent softness enables them to safely interact with delicate objects, and to passively adapt their shape to unstructured environments [26]. Such traits are desirable for robotic applications that demand safe human-robot interaction such as wearable robots, in-home assistive robots, and medical robots. Unfortunately, the soft bodies of these robots also impose modeling and control challenges, which have restricted their functionality to date. While many novel soft devices such as soft grippers [13], crawlers [32], and swimmers [16] exploit the flexibility of their bodies to achieve coarse behaviors such as grasping and locomotion, they do not exhibit precise control capabilities.

The challenge in constructing such precise control techniques is due in large part to the difficulty of devising models of soft robots that are amenable to model-based control design techniques. Consider for instance a rigid-bodied robotic system that is made up of rigid links connected together by discrete joints. Since joint displacements can be used to fully describe the configuration of a rigid-bodied system, joint displacements and their derivatives make a natural choice for the state variables for rigid-bodied robots [27]. One can use this choice of state variables to describe the dynamics of the rigid-bodied robot. This, as a result, makes the application of model-based control design techniques such as feedback linearization design for the lifted linear system (top-right). This paper develops a data-driven method to construct such a lifted model representation for soft robotic systems in the presence of outliers and a to construct a convex, model-based control design technique for such systems.

Fig. 1. A nonlinear dynamical system (bottom-left) has a linear representation in the lifted space made up of all real-valued functions (top-left). While a model predictive controller (MPC) designed for the nonlinear system in state-space requires solving a non-convex optimization problem to choose inputs at each time-step (bottom-right), this problem is convex for an MPC controller designed for the lifted linear system (top-right). This paper develops a data-driven method to construct such a lifted model representation for soft robotic systems in the presence of outliers and a to construct a convex, model-based control design technique for such systems.

Soft robots, in contrast, do not exhibit localized deformation at discrete joints, but instead deform continuously along their bodies and have infinite degrees-of-freedom. In the absence of joints, there does not yet exist a canonical choice of state variables to describe the geometry of a soft robot. As a result, existing representations are typically only rich enough to describe the system under restrictive simplifying assumptions. For example, the popular piecewise constant curvature model [35] provides a low-dimensional description of the shape of continuum robots, but only under the assumption that bending occurs in sections of constant curvature. Other simplified models such as pseudo-rigid-body [12] and quasi-static [5, 30, 9, 34] have proven useful, but they are only able to describe behavior in the subset of conditions over which the simplifying assumptions hold. This can make applying model-
based control design techniques impractical.

Alternatively, data-driven methods such as traditional machine learning and deep learning can be applied to construct models for soft robots without making structural simplifying assumptions. Such models provide a “black-box” mapping from inputs to outputs and have been shown to predict behavior well across various configurations of soft robots [8, 30]. However, since no explicit model is constructed, these methods are also not amenable to existing model-based control design techniques.

Koopman operator theory offers an approach that can overcome the challenges of modeling and controlling soft robots. The approach leverages the linear structure of the Koopman operator to construct linear models of nonlinear controlled dynamical systems from input-output data [6, 18], and to control them using established linear control methods [1, 14]. In theory, this approach involves lifting the state-space to an infinite-dimensional space of scalar functions (referred to as observables), where the flow of such observables along trajectories of the nonlinear dynamical system is described by the linear Koopman operator. In practice, however, it is not feasible to compute an infinite-dimensional operator, so a modified version of the Extended Dynamic Mode Decomposition (EDMD) is employed to compute a finite-dimensional projection of the Koopman operator onto a finite-dimensional subspace of all observables (scalar functions). This approximation of the Koopman operator describes the evolution of the output variables themselves, provided that they lie within the finite subspace of observables upon which the operator is projected. Hence, this approach makes it possible to control the output of a nonlinear dynamical system using a controller designed for its linear Koopman representation.

The Koopman approach to modeling and control is well suited for soft robots for several reasons. Soft robots pose less of a physical threat to themselves or their surroundings when subjected to random control inputs than conventional rigid-bodied robots. This makes it possible to safely collect input-output data over a wide range of operating conditions, and to do so in an automated fashion. Furthermore, since the Koopman procedure is entirely data-driven, it inherently captures input-output behavior and avoids the ambiguity involved in choosing a discrete set of states for a structure with infinite degrees of freedom.

The work presented here can be considered an extension of the work on Koopman-based modeling and control of Mauroy and Goncalves [18] and Korda and Mezić [14]. The novel contributions of this work, as depicted in Fig. 1 are:

1) An extension to the Koopman system identification procedure described in [18] to make the resulting Koopman operator both more sparse and less sensitive to outliers and noise in the training data.
2) The application of this identified Koopman model for model predictive control of a physical soft robotic system.

The rest of this paper is organized as follows: In Section II we formally introduce the Koopman operator and describe how it is used to construct linear models of nonlinear dynamical systems. In Section III we describe how the Koopman model can be used to construct a linear model predictive controller (MPC). In Section IV we describe the soft robot and the set of experiments used to evaluate the performance of a Koopman-based MPC controller. In Section V concluding remarks and perspectives are provided.

II. LINEAR SYSTEM IDENTIFICATION

Any finite-dimensional, Lipschitz continuous nonlinear dynamical system has an equivalent infinite-dimensional linear representation in the space of all real-valued functions of the system’s state [15, Definition 3.3.1]. This linear representation, which is called the Koopman operator, describes the flow of functions along trajectories of the system. While it is not possible to numerically represent the infinite-dimensional Koopman operator, it is possible to represent its projection onto a finite-dimensional subspace as a matrix. This section shows that for a given choice of basis functions, a lifted linear dynamical system model can be extracted directly from the matrix approximation of the Koopman operator. The remainder of this sections outlines the approach for constructing the Koopman operator approximation and the linear system representation from data. Section III illustrates how this model can be incorporated into a model predictive control algorithm.

A. Koopman Representation of a Dynamical System

Consider a dynamical system

\[
\dot{x}(t) = F(x(t))
\]  

where \( x(t) \in X \subset \mathbb{R}^n \) is the state of the system at time \( t \geq 0 \), \( X \) is a compact subset, and \( F \) is a continuously differentiable function. Denote by \( \phi(t, x_0) \) the solution to (1) at time \( t \) when beginning with the initial condition \( x_0 \) at time 0. For simplicity, we denote this map, which is referred to as the flow map, by \( \phi(t, x_0) \) instead of \( \phi(t, x_0) \).

The system can be lifted to an infinite dimensional function space \( \mathcal{F} \) composed of all square-integrable real-valued functions with compact domain \( X \subset \mathbb{R}^n \). Elements of \( \mathcal{F} \) are called observables. In \( \mathcal{F} \), the flow of the system is characterized by the set of Koopman operators \( U_t : \mathcal{F} \to \mathcal{F} \), for each \( t \geq 0 \), which describes the evolution of the observables \( f \in \mathcal{F} \) along the trajectories of the system according to the following definition:

\[
U_t f = f \circ \phi_t,
\]  

where \( \circ \) indicates function composition. As desired, \( U_t \) is a linear operator even if the system (1) is nonlinear, since for \( f_1, f_2 \in \mathcal{F} \) and \( \lambda_1, \lambda_2 \in \mathbb{R} \)

\[
U_t(\lambda_1 f_1 + \lambda_2 f_2) = \lambda_1 f_1 \circ \phi_t + \lambda_2 f_2 \circ \phi_t = \lambda_1 U_t f_1 + \lambda_2 U_t f_2.
\]  

Thus, the Koopman operator provides a linear representation of the flow of a nonlinear system in the infinite-dimensional space of observables (see Fig. 1) [7]. Contrast this representation with the one generated by the (nonlinear) flow map that
for each $t \geq 0$ describes how the initial condition evolves according to the dynamics of the system. In particular if one wants to understand the evolution of an initial condition $x_0$ at time $t$ according to (1), then one could solve the nonlinear differential equation to generate the flow map. On the other hand, one could apply $U_t$ (a linear operator) to the indicator function centered at $x_0$ (i.e. the function that is 1 at $x_0$ and zero everywhere else) to generate an indicator function centered at the point $\phi_t(x_0)$.

### B. Identification of Koopman Operator

Since the Koopman operator is an infinite-dimensional object, it cannot be represented by a finite-dimensional matrix. Therefore, we settle for the projection of the Koopman operator onto a finite-dimensional subspace. Using a modified version of the Extended Dynamic Mode Decomposition (EDMD) algorithm [36] originally presented in [18, 19], we identify a finite-dimensional approximation of the Koopman operator via linear regression applied to observed data.

Define $\bar{F} \subset F$ to be the subspace of $F$ spanned by $N > n$ linearly independent basis functions $\{\psi_i : \mathbb{R}^n \rightarrow \mathbb{R}\}_{i=1}^N$. We denote the image of $\psi_i$ as $\mathcal{R}_i$ which is equal to $\{w \in \mathbb{R} | \exists x \in \mathbb{R}^n \text{ such that } \psi_i(x) = w\}$. For convenience, we assume that the first $n$ basis functions are defined as

$$
\psi_i(x) = x_i
$$

where $x_i$ denotes the $i^{th}$ element of $x$. Any observable $\bar{f} \in \bar{F}$ can be expressed as a linear combination of elements of these basis functions

$$
\bar{f} = \theta_1 \psi_1 + \cdots + \theta_N \psi_N
$$

where $\theta_i \in \mathbb{R}$. To aid in presentation, we introduce the vector of coefficients $\theta = [\theta_1 \cdots \theta_N]^\top$ and the lifting function $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^N$ defined as:

$$
\psi(x) := \begin{bmatrix} x_1 & \cdots & x_n & \psi_{n+1}(x) & \cdots & \psi_N(x) \end{bmatrix}^\top.
$$

By (5) and (6), $\bar{f}$ evaluated at a point $x$ in the state space is given by

$$
\bar{f}(x) = \theta^\top \psi(x)
$$

We therefore refer to $\psi(x)$ as the lifted state, and $\theta$ as the vector representation of $\bar{f}$.

Given this vector representation for observables, a linear operator $L : \bar{F} \rightarrow \bar{F}$ can be represented as an $N \times N$ matrix. We denote by $\bar{U}_t \in \mathbb{R}^{N \times N}$ the approximation of the Koopman operator in $\bar{F}$, which operates on observables via matrix multiplication:

$$
\bar{U}_t \theta = \theta'
$$

where $\theta, \theta'$ are each vector representations of observables in $\bar{F}$.

Our goal is to find a $\bar{U}_t$ that describes the action of the infinite dimensional Koopman operator $U_t$ as accurately as possible in the $L^2$-norm sense on the finite dimensional subspace $\bar{F}$ of all observables.

To perfectly mimic the action of $U_t$ on an observable $\bar{f} \in \bar{F} \subset F$, according to (2) the following should be true for all $x \in X$

$$
\bar{U}_t \bar{f}(x) = \bar{f} \circ \phi_t(x)
$$

(9)

$$
(\bar{U}_t \theta)^\top \psi(x) = \theta^\top \psi \circ \phi_t(x)
$$

(10)

$$
\bar{U}_t^\top \psi(x) = \psi \circ \phi_t(x),
$$

(11)

where the second equation follows by substituting (5) and the last equation follows by cancelling $\theta^\top$. Since this is a linear equation, it follows that for a given $x \in X$, solving (11) for $\bar{U}_t$ yields the best approximation of $U_t$ on $\bar{F}$ in the $L^2$-norm sense [22]:

$$
\bar{U}_t = (\psi^\top(x))^{\dagger} (\psi \circ \phi_t(x))^\top
$$

(12)

where superscript $\dagger$ denotes the least-squares pseudoinverse.

To approximate the Koopman operator from a set of experimental data, we take $K$ discrete state measurements in the form of so-called “snapshot pairs” $(a[k], b[k])$ for each $k \in \{1, \ldots, K\}$ where

$$
a[k] = x[k]
$$

(13)

$$
b[k] = \phi_{T_s}(x[k]) + \sigma[k],
$$

(14)

where $\sigma[k]$ denotes measurement noise, $T_s$ is the sampling period which is assumed to be identical for all snapshot pairs, and $x[k]$ denotes the measured state corresponding to the $k^{th}$ measurement. Note that consecutive snapshot pairs do not have to be generated by consecutive state measurements. We then lift all of the snapshot pairs according to (6) and compile them into the following $K \times N$ matrices:

$$
\Psi_a := \begin{bmatrix} \psi(a[1])^\top \\ \vdots \\ \psi(a[K])^\top \end{bmatrix}
$$

$$
\Psi_b := \begin{bmatrix} \psi(b[1])^\top \\ \vdots \\ \psi(b[K])^\top \end{bmatrix}
$$

(15)

$\bar{U}_{T_s}$ is chosen so that it yields the least-squares best fit to all of the observed data, which, following from (12), is given by

$$
\bar{U}_{T_s} := \Psi_a^{\dagger} \Psi_b.
$$

(16)

Sometimes a more accurate model can be attained by incorporating delays into the set of snapshot pairs. To incorporate these delays, we define the snapshot pairs as

$$
a[k] = [x[k]^\top, x[k-1]^\top, \ldots, x[k-d]^\top]^\top
$$

(17)

$$
b[k] = [(\phi_{T_s}(x[k]) + \sigma[k])^\top, x[k]^\top, \ldots, x[k-d+1]^\top]^\top
$$

(18)

where $d$ is the number of delays. We then modify the domain of the lifting function such that $\psi : \mathbb{R}^{n+d} \rightarrow \mathbb{R}^N$ to accommodate the larger dimension of the snapshot pairs. Once these snapshot pairs have been assembled, the model identification procedure is identical to the case without delays.
C. Building Linear System from Koopman Operator

For dynamical systems with inputs, we are interested in using the Koopman operator to construct discrete linear models of the following form

\[
\begin{aligned}
    z[j + 1] &= A z[j] + B u[j] \\
    x[j] &= C z[j]
\end{aligned}
\]  

(19)

for each \( j \in \mathbb{N} \), where \( x[0] \) is the initial condition in state space, \( z[0] = \psi(x[0]) \) is the initial lifted state, \( u[j] \in \mathbb{R}^m \) is the input at the \( j \)th step, and \( C \) acts as a projection operator from the lifted space onto the state-space. Specifically, we desire a representation in which (non-lifted) inputs appear linearly in the resulting model. With these pairs, we define the following \( K \times (N + m) \) matrices:

\[
\begin{aligned}
    \Gamma_\alpha &= \begin{bmatrix}
        \alpha[1]^T \\
        \vdots \\
        \alpha[K]^T
    \end{bmatrix} \\
    \Gamma_\beta &= \begin{bmatrix}
        \beta[1]^T \\
        \vdots \\
        \beta[K]^T
    \end{bmatrix}
\end{aligned}
\]  

(21)

and solve for the corresponding Koopman operator according to (12)

\[
\hat{U}_T := \Gamma_\alpha \Gamma_\beta^{-1}.
\]  

(22)

Note that by (11) and (22) the transpose of this Koopman matrix is the best approximation of a transition matrix between the elements of snapshot pairs in the \( L^2 \)-norm sense

\[
\hat{U}_T^T \begin{bmatrix}
    \psi(a[k]) \\
    u[k]
\end{bmatrix} \approx \begin{bmatrix}
    \psi(b[k]) \\
    u[k]
\end{bmatrix},
\]  

(23)

and we desire the best \( A, B \) matrices such that

\[
A \psi(a[k]) + Bu[k] \approx \psi(b[k])
\]  

(24)

Therefore, the best \( A \) and \( B \) matrices of (19) are embedded in \( \hat{U}_T^T \) and can be isolated by partitioning it as follows:

\[
\hat{U}_T^T = \begin{bmatrix}
    A_{N\times N} & B_{N\times m} \\
    O_{m\times N} & I_{m\times m}
\end{bmatrix}
\]  

(25)

where \( I \) denotes an identity matrix, \( O \) denotes a zero matrix, and the subscripts denote the dimensions of each matrix. The \( C \) matrix is defined

\[
C = \begin{bmatrix}
    I_{n\times n} & O_{n\times (N-n)}
\end{bmatrix}
\]  

(26)

since by (4), \( x = [\psi_1(x), \ldots, \psi_n(x)] \). Note we can also incorporate input delays into the model by appending them to the snapshot pairs as we did in (17) and (18).
Algorithm 1: Koopman Linear System Identification

Input: $\lambda$, $\{a[k], b[k]\}$ and $u[k]$ for $k = 1, \ldots, K$

Step 1: Lift data via (6)
Step 2: Combine lifted data and inputs via (20)
Step 3: Approximate Koopman operator $\hat{U}_T$, via (27)
Step 4: Extract model matrices $A, B$ via (25)
Step 5: Identify projection operator $P$ via (30)

Output: $\hat{A} := PA$, $\hat{B} := PB$

accuracy in prediction. As illustrated in Fig. 2, the lifting function $\psi$ maps from $\mathbb{R}^n$ to $\mathcal{M}$, but at some time step $j$, $A\psi(a[j]) + B\psi(a[j])$ may not map onto $\mathcal{M}$. When this happens and we try to simulate our linear model from an initial condition, it may leave the space of legitimate “lifted states” rapidly and fail to predict behavior accurately. We therefore desire the sparsest model that minimizes the distance from $\mathcal{M}$ at each iteration.

This can be accomplished by applying a projection operator at each time step. For each snapshot pair, the ideal projection operator $P$ should satisfy the following for all $k$:

$$ P(A\psi(a[k]) + Bu[k]) = \psi(b[k]). $$

(28)

To build an approximation to this operator, we construct the following $K \times N$ matrix,

$$ \Omega_a := \begin{bmatrix} (A\psi(a[1]) + Bu[1])^\top \\
\vdots \\
(A\psi(a[K]) + Bu[K])^\top \end{bmatrix}. $$

(29)

Then the best projection operator in the $L^2$-norm sense based on our data is given by

$$ P := (\Omega_a^\top \Omega_a)^{-1} \Omega_a^\top. $$

(30)

Composing $P$ with the $A$ and $B$ matrices in (19) yields a modified linear model that significantly reduces the distance from $\mathcal{M}$ at each iteration.

$$ z[j+1] = \hat{A}z[j] + \hat{B}u[j] $$

(31)

where $\hat{A} := PA$ and $\hat{B} := PB$. Algorithm 1 summarizes the proposed model construction process.

III. MODEL PREDICTIVE CONTROL

A system model enables the design of model-based controllers that leverage model predictions to choose suitable control inputs for a given task. In particular, model-based controllers can anticipate future events, allowing them to optimally choose control inputs over a finite time horizon. The most popular model-based control design technique is model predictive control (MPC), wherein one optimizes the control input over a finite time horizon, applies that input for a single timestep, and then optimizes again, repeatedly [24]. For linear systems, MPC consists of iteratively solving a convex quadratic program.

Algorithm 2: Koopman-Based MPC

Input: Prediction horizon: $N_h$

Cost matrices: $G_i, H_i, g_i, h_i$ for $i = 0, \ldots, N_h$

Constraint matrices: $E_i, F_i, b_i$ for $i = 0, \ldots, N_h$

Model matrices: $A, B$

for $k = 0, 1, 2, \ldots$ do

Step 1: Set $z[0] = \psi(x[k])$

Step 2: Solve (32) to find optimal input $(u[i])_{i=0}^{N_h}$

Step 3: Set $u[k] = u[0]^*$

Step 4: Apply $u[k]$ to the system

end

Importantly, this is also the case for Koopman-based MPC control, wherein one solves the following program at each time instance $k$ of the closed-loop operation:

$$ \min_{u[i], z[i]} z[N_h]^TG_N z[N_h] + g_N^T z[N_h] + \sum_{i=0}^{N_h-1} z[i]^T G_i z[i] + u[i]^T H_i u[i] + g_i^T z[i] + h_i^T u[i] $$

s.t.

$$ z[i+1] = \hat{A}z[i] + \hat{B}u[i], \quad \forall i \in \{0, \ldots, N_h - 1\} $$

$$ E_i z[i] + F_i u[i] \leq b_i, \quad \forall i \in \{0, \ldots, N_h - 1\} $$

$$ z[0] = \psi(x[k]) $$

(32)

where $N_h \in \mathbb{N}$ is the prediction horizon, $G_i \in \mathbb{R}^{N \times N}$ and $H_i \in \mathbb{R}^{m \times m}$ are positive semidefinite matrices, and where each time the program is called, the predictions are initialized from the current lifted state $\psi(x[k])$. The matrices $E_i \in \mathbb{R}^{c \times N}$ and $F_i \in \mathbb{R}^{c \times m}$ and the vector $b_i \in \mathbb{R}^{c}$ define state and input polyhedral constraints where $c$ denotes the number of imposed constraints. While the size of the cost and constraint matrices in (32) depend on the dimension of the lifted state $N$, Korda and Mezić [14] show these can be rendered independent of $N$ by transforming the problem into its so-called "dense-form." Algorithm 2 summarizes the closed-loop operation of this Koopman based MPC controller.

Since this optimization problem is convex, it has a unique globally optimal solution that can efficiently be constructed without initialization [4] for models with thousands of states and inputs [21, 28]. This contrasts sharply with the MPC formulation for nonlinear systems (referred to as nonlinear model predictive control or NMPC [2]). NMPC requires solving an optimization problem with nonlinear constraints and a (potentially) nonlinear cost function. As a result, algorithms to solve such problems typically require initialization and can struggle to find globally optimal solutions [23]. Though techniques have been proposed to improve the speed of algorithms to solve NMPC problems [20, 11] or even globally solve such problems without requiring initialization [37], these formulations still take several seconds per iteration, which can make them too slow to be applied during real-time control.
IV. EXPERIMENTS

This section describes the robot and the set of experiments used to demonstrate the efficacy of the modeling and control methods from Sections II and III. Video of the soft robot performing several tasks from the final experiment is included in a supplementary video file.

A. Robot Description: Soft Arm with Laser Pointer

The robot used for the experiments is a suspended soft arm with a laser pointer attached to the end effector (see Fig. 3). The laser dot is projected onto a 50 cm × 50 cm flat board which sits 34 cm beneath the tip of the laser pointer when the robot is in its relaxed position (i.e. hanging straight down). The position of the laser dot is measured by a digital webcam overlooking the board.

The arm itself consists of two sections that are each composed of three pneumatic artificial muscles or PAMs (also known as McKibben actuators [33]) adhered to a central foam spine by latex rubber bands (see Fig. 3). The PAMs in the upper and lower sections are internally connected so that only three input pressure lines are required, and they are arranged such that for any bending of the upper section, bending in the opposite direction occurs in the bottom section. This ensures that the laser pointer mounted to the end effector points approximately vertically downward so that the laser light strikes the board at all times. The pressures inside the actuators are regulated by three Enfield TR-010-g10-s pneumatic pressure regulators, which accept 0 – 10V command signals corresponding to pressures of ≈ 0 – 140 kPa. In the experiments there is a three-dimensional input corresponding to the voltages into the three pressure regulators and a two-dimensional state corresponding to the position of the laser dot with respect to the center of the board.

B. Characterization of Stochastic Behavior

Most mechanical systems demonstrate stochastic behavior (i.e. when an identical input and state produces a different output) to some extent. Stochastic behavior is characteristic of electronic pressure regulators, which can limit the precision of pneumatically driven soft robotic systems and undermine the predictive capability of models.

We quantified the stochastic behavior of our soft robot system by observing the variations in output from period-to-period under sinusoidal inputs to the three actuators of the form

\[
    u[k] = \begin{bmatrix}
        6 \sin\left(\frac{2\pi}{T_s} k T_s\right) + 3 \\
        6 \sin\left(\frac{2\pi}{T_s} k T_s - \frac{\pi}{3}\right) + 3 \\
        6 \sin\left(\frac{2\pi}{T_s} k T_s - \frac{\pi}{2}\right) + 3
    \end{bmatrix}
\]

for periods of \( T = 6, 7, 8, 9, 10, 11, 12 \) seconds and a sampling time of \( T_s = 0.1 \) seconds with a zero-order-hold between samples. Under these inputs, the laser dot traces out a circle with some variability in the trajectory over each period. In

\[1\]https://youtu.be/c35o2OPsQHs

C. Data Collection and Model Identification

To construct a model, we ran the system through 16 trials each lasting approximately 20 minutes. A randomized input was applied during each trial to generate a representative sampling of the system’s behavior over its entire operating range. To ensure randomization, a matrix \( \Upsilon \in [0, 10]^{3 \times 1000} \) of uniformly distributed random numbers between zero and ten was generated to be used as an input lookup table. Each control input was smoothly varied between elements in consecutive columns of the table over a transition period \( T_u \), with a time
TABLE I
AVERAGE PREDICTION ERROR OVER 2.5 SECOND HORIZON (CM)

<table>
<thead>
<tr>
<th>Period of Sinusoidal Inputs (seconds)</th>
<th>Model</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Koopman</td>
<td>2.21</td>
<td>2.78</td>
<td>1.35</td>
<td>1.53</td>
<td>1.21</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>Linear S.S.</td>
<td>4.64</td>
<td>4.54</td>
<td>3.94</td>
<td>3.56</td>
<td>3.15</td>
<td>2.72</td>
</tr>
</tbody>
</table>

offset of $T_u/3$ between each of the three control signals

$$u_i(t) = \left(\frac{\gamma_{i,k+1} - \gamma_{i,k}}{T_u}\right) \left(\frac{t + (i-1)T_u}{3}\right) + \gamma_{i,k} \quad (34)$$

where $k = \text{floor}(t/T_u)$ is the current index into the lookup table at time $t$. The transition period $T_u$ varied from 5 seconds to 10 seconds between trials. After collection, the data was uniformly sampled with period $T_u = 0.1$ seconds.

Two models were fit from the data: a Koopman model, and a linear state space model. The linear state-space model provides a baseline for comparison and was identified from the same data as the Koopman model using the MATLAB System Identification Toolbox [17]. This model is a four dimensional linear state-space model expressed in observer canonical form. The Koopman model was identified via the method described in Section II on a set of 191,000 snapshot pairs $\{a[k], b[k]\}$ that incorporate a single delay $d = 1$:

$$a[k] = [x[k] \quad x[k-1] \quad u[k-1]]^\top \quad (35)$$

$$b[k] = [(\phi T_s(x[k]) + \sigma[k])^\top \quad x[k]^\top \quad u[k]^\top]^\top, \quad (36)$$

and using an $N = 330$ dimensional set of basis functions consisting of all monomials of maximum degree 4. To find the sparsest acceptable matrix representation of the Koopman operator, equation (27) was solved for $\lambda = 0, 1, 2, \ldots, 50$. Predictions from the resulting models were evaluated against a subset of the training data, with the error quantified as the average Euclidean distance between the prediction and actual trajectory at each point, normalized by dividing by the average Euclidean distance between the actual trajectory and the origin. Fig. 5 shows that as $\lambda$ increases so does this error, but the density of the $A$ matrix of the lifted linear model decreases. The model chosen is the one that minimizes prediction error, which results in an $A$ matrix with 70% of its entries equal to zero.

D. Experiment 1: Model Prediction Comparison

The accuracy of the predictions generated by each of the two models was evaluated by comparing them to the actual behavior of the system under the sinusoidal inputs defined in (33). The model responses were simulated over a time horizon of 2.5 seconds given the same initial condition and input as the real system. The results of this comparison are summarized by Fig. 6 and Table I. They illustrate that the Koopman model predictions are more accurate over the time horizon.

E. Experiment 2: Model-Based Control Comparison

The two identified models were each used to build a model predictive controller which solves an optimization problem in the form of (32) at each time step using the Gurobi Optimization software [10]. We refer to the two controllers by the abbreviations K-MPC for the one based on the Koopman model, and L-MPC for the one based on the linear state-space model. Both model predictive controllers run in closed-loop at 10 Hz, feature an MPC horizon of 2.5 seconds ($N_h = 25$), and a cost function that penalizes deviations from a reference trajectory $r[k]$ over the horizon with both a running and terminal cost:

$$\text{Cost} = 100 \left(y[N_h] - r[N_h]\right)^2 + \sum_{i=0}^{N_h-1} 0.1 \left(y[i] - r[i]\right)^2 \quad (37)$$

In the K-MPC case, $y[i] = Cz[i]$, where $C$ is defined as in (26). In the L-MPC case, $y[i] = C_L x_L[i]$ where $x_L$ is the four dimensional system state and $C_L$ is the projection matrix that
isolates the states describing the current laser dot coordinates.

The performance of the controllers was assessed with respect to a set of three trajectory following tasks. Each task was to follow a reference trajectory as it traced out one of the following shapes over a certain amount of time:
1) Pacman (90 seconds)
2) Star (180 seconds)
3) Block letter M (300 seconds)

The error for each trial was quantified as the Euclidean distance from the reference trajectory at each time step over the length of the trial.

The performances of the K-MPC and L-MPC controllers at Tasks 1, 2, and 3 are shown visually in Fig. 7, and the error is quantified in Table II. In both tasks the K-MPC controller achieved better performance, exhibiting an average tracking error of 1.26 cm compared to the L-MPC controller’s average error of 2.45 cm. This amounts to an average error roughly 25% larger than the maximum magnitude of observed noise (see Fig. 4).

### V. Conclusion

In this work, a data-driven modeling and control method based on Koopman operator theory was successfully applied to a soft robot. The Koopman-based MPC controller was shown to be capable of commanding a soft robot to follow a reference trajectory better than an MPC controller based on another linear data-driven model. By making explicit control-oriented models of soft robots easier to construct, this method enables the rapid development of new control strategies and applications.

While these preliminary results are promising, further work is needed to make such methods feasible for higher dimensional robotic systems. Toward that end, this work introduced a method for promoting sparsity in matrix representations of the Koopman model. Additional work will explore strategies for further promoting sparsity, choosing the most effective basis of observables, and building models that can account for external loading and contact forces.

### Table II

<table>
<thead>
<tr>
<th>Controller</th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Avg.</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-MPC</td>
<td>1.25</td>
<td>1.19</td>
<td>1.34</td>
<td>1.26</td>
<td>0.07</td>
</tr>
<tr>
<td>L-MPC</td>
<td>2.21</td>
<td>2.34</td>
<td>2.73</td>
<td>2.45</td>
<td>0.31</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

This material is supported by the Toyota Research Institute, and is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1256260 DGE. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES


