Sim-to-Real of Soft Robots With Learned Residual Physics

Junpeng Gao, Mike Y. Michelis, Andrew Spielberg, and Robert K. Katzschmann, *Senior Member, IEEE*

***Abstract*—Accurately modeling soft robots in simulation is computationally expensive and commonly falls short of representing the real world. This well-known discrepancy, known as the sim-to-real gap, can have several causes, such as coarsely approximated geometry and material models, manufacturing defects, viscoelasticity and plasticity, and hysteresis effects. Residual physics networks learn from real-world data to augment a discrepant model and bring it closer to reality. Here, we present a residual physics method for modeling soft robots with large degrees of freedom. We train neural networks to learn a residual term — the modeling error between simulated and physical systems. Concretely, the residual term is a force applied on the whole simulated mesh, while real position data is collected with only sparse motion markers. The physical prior of the analytical simulation provides a starting point for the residual network, and the combined model is more informed than if physics were learned *tabula rasa*. We demonstrate our method on *1)* a silicone elastomeric beam and *2)* a soft pneumatic arm with hard-to-model, anisotropic fiber reinforcements. Our method outperforms traditional system identification up to 60%. We show that residual physics need not be limited to low degrees of freedom but can effectively bridge the sim-to-real gap for high dimensional systems.**

***Index Terms*—Deep learning methods, modeling, control, and learning for soft robots, dynamics, optimization and optimal control, simulation and animation.**

I. Introduction

We present a data-driven approach for reducing the sim-to-real gap in soft robotics. Despite soft robots' promise in solving tasks that are difficult for rigid robots to solve (e.g., delicate manipulation [1] and biomimicry [2]), modeling soft robots remains computationally expensive and physically inaccurate. This challenge hinders the application of computational methods for downstream tasks such as optimal control and design. By providing a generic means to improve simulation accuracy that is system agnostic, we can unlock applications across the diverse zoo of soft robotics.

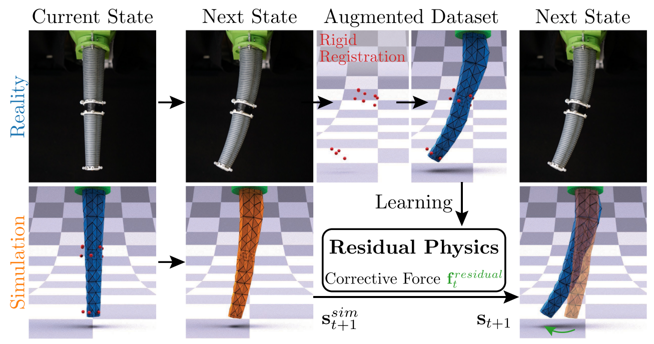

Fig. 1. Overview of the residual physics pipeline for high dimensional systems, demonstrated with a soft robotic arm. The learned residual force compensates for state-to-state prediction errors, such that sparse motion markers in simulation match those in reality.

Simulators that model soft robots commonly have limited options for fitting physical parameters. Once a material model is chosen, only a small set of parameters, such as material density, stiffness, compressibility, friction, and damping, can be tweaked to adjust the behavior of a soft body. In most practical applications, such parameter tweaking is sufficient to better match the simulated model with its real-world counterpart; this process is referred to as system identification (SysID) [3], [4], [5]. However, suppose the simulator's physics does not match the real world for reasons other than parameter mismatch, for example, incorrect material model, overly coarse discretization in time or space, or simulation artifacts (such as locking in finite element methods). In this case, the expressivity of the simulator may not suffice to cover the real-world dynamics.

We propose to combine deformable body simulators with data-driven auxiliary models, as a means of reducing the sim-to-real gap. We see this approach as a viable alternative to tediously modeling every possible aspect of every continuum-bodied robot, which would only grow with the creation of further soft robotic systems. Our *sparse residual physics learning framework* is a hybrid formulation taking advantage of a differentiable Finite Element Method (FEM) simulator and deep learning. Our framework learns a residual body force on the soft structure that captures the difference between the

Manuscript received 1 February 2024; accepted 24 July 2024. Date of publication 20 August 2024; date of current version 29 August 2024. This article was recommended for publication by Associate Editor C. Duriez and Editor C. Laschi upon evaluation of the reviewers' comments. This work at the Soft Robotics Lab was supported by Credit Suisse, Switzerland. The work of Mike Y. Michelis was supported by ETH AI Center through doctoral fellowship. The work of Andrew Spielberg was supported by an Amazon Research Award. *(Junpeng Gao and Mike Y. Michelis contributed equally to this work.) (Corresponding author: Robert K. Katzschmann.)*

Junpeng Gao is with Soft Robotics Lab, Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zürich, Switzerland (e-mail: jungao@ethz.ch).

Mike Y. Michelis and Robert K. Katzschmann are with Soft Robotics Lab and ETH AI Center, ETH Zurich, 8092 Zürich, Switzerland (e-mail: michelism@ethz.ch; rkk@ethz.ch).

Andrew Spielberg is with Harvard University, Cambridge, MA 02138 USA (e-mail: aespielberg@seas.harvard.edu).

Videos, code, and data are available on our website at: https://srl-ethz.github.io/website_residual_physics/.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3446287, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3446287

simulator and the real-world, *directly* minimizing the sim-to-real error. This formulation combines the qualitative priors of a coarse simulator, e.g., direction of bending and approximate magnitude of deformation, with the fine-tuning derived from real-world data. Unlike previous residual learning approaches (e.g. [6], [7]) we operate in a regime of sparse observation data, since continuum structures that cannot be fully sensorized are inherently partially observable. Our approach regularizes on system dynamics to create physically reasonable candidates for target learned motions and provides a means for a simulator to capture unmodeled robot dynamics and unseen settings.

We provide a full pipeline[1] from spatially sparse data to dense residual force estimation (visualized in Fig. 1) and apply our approach to both software and hardware experiments. Our approach is easy to apply in practice and can improve soft robotics engineering workflows through more reliable modeling. In summary, we contribute:

1) *Residual physics learning framework for soft robotics.* We design a hybrid learning framework that speeds up simulation while increasing simulation accuracy in both sim-to-sim and sim-to-real settings.
2) *Dense residuals from sparse observations.* We propose a data pre-processing method to build an augmented dataset from sparse real-world data (markers $\sim 10^1$) for learning residual physics on the discretized geometry of deformable bodies (degrees of freedom $\sim 10^3$).
3) *Overcoming shortcomings of system identification.* We benchmark our framework on dynamical high-dimensional systems, such as passive and actuated soft robots, and show that even optimal tuning of physical simulation parameters falls short in accuracy compared to our approach.

II. Related Work

Much effort has gone into developing more efficient and accurate simulators for applications of deformable systems over the years [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. The consistent trend, however, has been a trade-off between speed and accuracy. For the computer graphics community, a visually plausible simulation for soft bodies usually suffices for applications in animation and digital gaming. Fidelity is sacrificed for speed, enabling real-time simulation of deformable virtual characters [8], [9], [10]. Within the soft robotics community, much more emphasis is placed on the physical accuracy of the model to match real-world experiments [13], [14], [15], [16], [17]. Here, simulation is particularly important to test design performance without the labor of physical manufacturing and to derive optimal controllers on real-world systems.

Matching real-world experiments, or in other words, closing the sim-to-real gap, is traditionally done through SysID, where a set of simulation parameters is tuned. In soft robotics, these parameters often include material characteristics such as stiffness and density. Gradient-free approaches are often sample-inefficient and hence costly to run for larger sets of parameters [3], [18], [19]. For this type of inverse problem, gradient-based optimization through differentiable simulation frameworks have offered significant improvement in convergence time [3], [4], [18], [19], [20], [21], [22]. Differentiable simulations provide analytical gradients for any measurable quantity of a simulation with respect to any simulation parameter; such gradients are useful for inverse problems such as trajectory-matching and system identification. While most methods estimate the state of the system through sparse motion markers, recent methods have integrated differentiable rendering pipelines to allow direct parameter matching from video data [21], [22].

Since one notable limitation of SysID is the need to re-run the procedure when characteristics about the system change, one approach is to learn a generalizable mapping from one environment to another, tuning these SysID parameters iteratively in a fast and sample-efficient manner [23]. Yet an overarching challenge for bridging the sim-to-real gap remains the many real-world physical phenomena that are not explicitly modeled by the underlying simulations, ranging from electric actuator dynamics [24] to flying robot aerodynamics [25]. This need for a more generalizable model for matching simulation and reality has given rise to the field of residual learning [6], [7], [24], [25], [26], [27]. Instead of learning the full system dynamics, residual physics methods use deep neural networks to learn only an error correction between an analytical simulator and real-world physics. Previous work, however, has only worked with low-dimensional state spaces, and it has yet to be scaled up to the high degree-of-freedom meshes used in soft-body simulations [28].

A valid alternative to the previous hybrid residual physics simulations would be fully end-to-end learning-based simulators [29], [30], [31], [32]. Although computationally efficient at inference time, these data-driven methods lack generalizability and robustness [6], [26]. This fragility can be ameliorated by including constrained neural networks into hybrid simulations [33]; the analytical solvers within such hybrid methods guide the solution along a physically plausible prediction. However, such an approach is confined to a PDE's structure and cannot handle unmodeled phenomena.

III. Simulation Preliminaries

We simulate our soft robots using DiffPD [18], a differentiable FEM simulator based on projective dynamics. Each robot is discretized as having N nodes, where we denote the position and velocity of nodes at time step t with $\mathbf{q}_t \in \mathbb{R}^{N\times 3}$ and $\mathbf{v}_t \in \mathbb{R}^{N\times 3}$ respectively. Using implicit Euler, the simulation is integrated in time according to Newton's second law of motion over fixed time interval h. The resulting equations to the discretized dynamical system can be formulated as

$$\begin{aligned} \mathbf{q}_{t+1} &= \mathbf{q}_t + h\mathbf{v}_{t+1} \\ \mathbf{v}_{t+1} &= \mathbf{v}_t + h\mathbf{M}^{-1}\left[\mathbf{f}^{\text{int}}\left(\mathbf{q}_{t+1}\right) + \mathbf{f}_t^{\text{ext}}\right] \end{aligned} \tag{1}$$

where $\mathbf{M}$ is the mass-matrix, $\mathbf{f}^{\text{int}}$ accounts for the sum of internal forces and $\mathbf{f}_t^{\text{ext}}$ for the sum of external forces. This results in a system of equations solved as a sometimes numerically and

[1]All code and data used in this letter are available at https://github.com/srl-ethz/residual_physics_sim2real

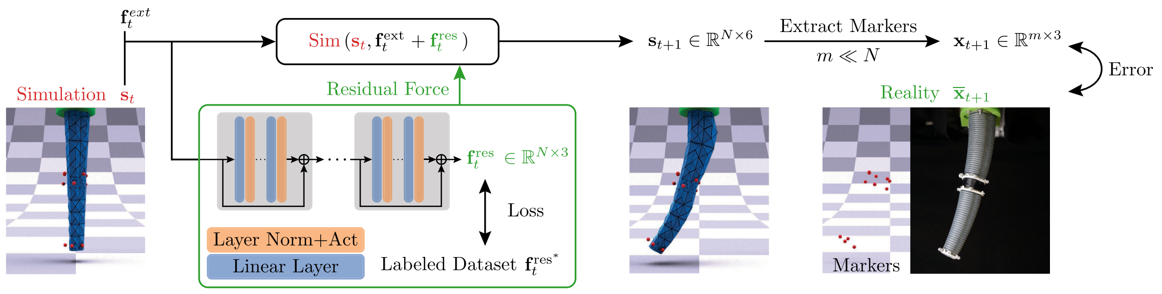
$\mathbf{f}_t^{ext}$
$\mathrm{Sim}\,(\mathbf{s}_t, \mathbf{f}_t^{\mathrm{ext}} + \mathbf{f}_t^{\mathrm{res}})$
$\mathbf{s}_{t+1} \in \mathbb{R}^{N\times 6}$
Extract Markers
$m \ll N$
$\mathbf{x}_{t+1} \in \mathbb{R}^{m\times 3}$
Error
Simulation $\mathbf{s}_t$
Residual Force
Reality $\overline{\mathbf{x}}_{t+1}$
$\mathbf{f}_t^{\mathrm{res}} \in \mathbb{R}^{N\times 3}$
Loss
Layer Norm+Act
Linear Layer
Labeled Dataset $\mathbf{f}_t^{\mathrm{res}^*}$
Markers

Fig. 2. Pipeline of how the residual physics forces $\mathbf{f}_t^{\mathrm{res}}$ compensate the erroneous simulated next state $\mathbf{s}_{t+1}$ to match the real observed marker state $\overline{\mathbf{x}}_{t+1}$. Our state $\mathbf{s}_t$ is defined by position $\mathbf{q}_t$ and velocity $\mathbf{v}_t$, from which we extract the motion markers $\mathbf{x}_t$ on the simulated mesh. The residual forces are predicted by a neural network given state and external force $\mathbf{f}_t^{\mathrm{ext}}$ information (such as pressure actuation) as input. This network is trained on a labeled augmented dataset of residual forces $\mathbf{f}_t^{\mathrm{res}^*}$, collected through gradient-based optimization in our differentiable simulation.

physically stiff optimization problem. For the sake of clarity, we simplify this forward solve as $\mathbf{q}_{t+1}, \mathbf{v}_{t+1} = \mathtt{Sim}(\mathbf{q}_t, \mathbf{v}_t, \mathbf{f}_t^{\mathrm{ext}})$.

The deformable structures used in this letter, a passive beam and a pneumatic arm, are both made from highly deformable silicone elastomers, namely Smooth-On Dragon Skin 10 with Shore Hardness 10A. As is common [18], we make a few modeling simplifications to aid elastic simulation stability. First, we use a corotational linear elastic material model. Second, silicone is typically assumed to be completely or nearly incompressible [4], i.e., Poisson's ratio $\nu = 0.5$ or 0.499. However, such Poisson's ratios are both unstable and time-stepping can take longer to converge when simulated; we set $\nu = 0.45$. We expect our residual physics (ResPhys) framework to be able to compensate for both the material model and compressibility assumptions. We roughly use manufacturer-provided values for material density and Young's modulus ($\rho = 1070\,\mathrm{kg\,m^{-3}}$ and $E = 215\,\mathrm{kPa}$), which have less impact on time-stepping convergence [5].

IV. Method

The objective of our framework is to learn a mapping from the state $\mathbf{s}_t$, consisting of positions $\mathbf{q}_t$ and velocity $\mathbf{v}_t$, and the action of the soft robot at time step t to an external residual body force $\mathbf{f}_t^{\mathrm{res}}$. The design of our framework is based on the assumptions that *1*) an external residual force in (1) can compensate for the residual dynamics of the simulated deformable objects, and *2*) if the simulator can predict the next state precisely, the residual force at each time step will follow similar distributions. This assumption will make the chosen neural network smaller and more efficient. In this section, we describe our procedure for leveraging data to overcome the sim-to-real gap in soft robot modeling, and illustrate the pipeline in Fig. 2. We begin by considering a simplified *sim-to-sim* setting, in which full state knowledge is available; we then relax this knowledge requirement to arrive at our method for *sim-to-real*, in which only partial information with measurement error is given about the physical world. Next, we describe the system identification we use as a baseline. Lastly, we describe the neural network used for the residual physics learning, how it is trained, and how we quantify the prediction performance.

A. Sim-to-Sim Setting

By removing the influence of potential fabrication and measurement errors, a *sim-to-sim* setting enables us to rigorously validate and refine our framework at a relatively low cost. Such a setting can provide essential insights and help pave the way for real-world experimentation.

We define a function $\mathtt{Sim}_n$ that takes as input state $\mathbf{s}_t$ and outputs position coordinates $\mathbf{q}_{t+1}$; the subscript n denotes a particular parameterization of DiffPD. We define the state $\mathbf{s}_t$ as a concatenation of $\mathbf{q}_t$ and $\mathbf{v}_t$ at time step t. We parameterize two differentiable simulators $\mathtt{Sim}_1, \mathtt{Sim}_2$ with different parameter configurations for sim-to-sim experiments. We aim to match the dynamics of $\mathtt{Sim}_2$ by injecting external residual forces generated from our framework into $\mathtt{Sim}_1$. Sim-to-sim scenarios provide a privileged, fully-known robot state at a low cost, as we can easily obtain the positions and velocity at each degree of freedom (DoF). We collect a series of ground truth motion data by running $\mathtt{Sim}_2$ offline.

We decompose the residual learning problem into two separate steps to ease the computational cost of rerunning each specific step. First, we leverage the differentiable property of the simulator to optimize for a dataset of external forces in $\mathtt{Sim}_1$ that makes the simulated motion trajectory best match the results of $\mathtt{Sim}_2$. Second, we perform supervised learning on this optimized dataset of external forces to create a neural network mapping between soft body state and residual forces. The network takes an input of the state $\mathbf{s}_t$ and, if the structure is actuated, the actuation forces, which are represented as external forces $\mathbf{f}_t^{\mathrm{ext}}$ (such as pneumatic pressure forces). The output is the residual force $\mathbf{f}_t^{\mathrm{res}}$ that helps the simulator correct the prediction. Note that gravity is applied separately in all simulations, but it is not considered as part of the external force input for the network.The first step can be formalized as follows:

$$\mathbf{f}_t^{\mathrm{res}^*} = \arg\min_{\mathbf{f}_t} \left\| \mathtt{Sim}_1(\mathbf{s}_t, \mathbf{f}_t^{\mathrm{ext}} + \mathbf{f}_t) - \overline{\mathbf{q}}_{t+1} \right\|_2^2 + \lambda \left\| \mathbf{f}_t \right\|_2^2 \quad (2)$$

where $\mathbf{f}_t$ is the residual force we aim to optimize at time step t, and $\overline{\mathbf{q}}_{t+1}$ is the ground truth full state from $\mathtt{Sim}_2$ at the next time step. We incorporate L_2 regularization with a weight λ so that the residual forces can help predict an accurate next state while having a small magnitude, aiding simulator stability. We leverage the differentiability from the simulator to solve the above optimization problem using an efficient L-BFGS-B minimization. In the first time step, we initialize a random residual force $\mathbf{f}_1 \sim \mathcal{N}(0, 10^{-4})$. In subsequent time steps, we use the preceding $\mathbf{f}_t$ as an initial guess for the ongoing step to solve the problem iteratively.

After we build a dataset of various trajectories over a fixed number of time steps each, we perform mini-batch training with batch-size M of the neural network $\mathcal{NN}$ with weights Θ based on the loss function:

$$\mathcal{L} := \sum_{i \in \text{batch}} \left\| \mathcal{NN}(\mathbf{s}_i, \mathbf{f}_i^{\text{ext}}; \Theta) - \mathbf{f}_i^{\text{res}^*} \right\|_2^2 + \lambda \left\| \Theta \right\|_2^2 \quad (3)$$

When we pass the state into the neural network, as a pre-processing step, we subtract the undeformed static state from the position $\mathbf{q}$. After each epoch, we run validation with the same loss function (3) on the validation set and save the model with the smallest validation error. This two-step training method can be thought of as a student-teacher formulation, in which residual forces are generated from privileged information, from which a residual network then learns with no privileged knowledge.

We test our trained model on R trajectories and evaluate the performance on T timesteps for each trajectory with:

$$\mathcal{E}_q := \frac{1}{R \cdot T \cdot N} \sum_{j=1}^{R} \sum_{t=1}^{T} \sum_{i=1}^{N} \left\| \mathbf{q}_{t,i}^{j} - \overline{\mathbf{q}}_{t,i}^{j} \right\|_2 \quad (4)$$

where $\mathbf{q}_{t,i}^{j}$ and $\overline{\mathbf{q}}_{t,i}^{j}$ are the simulated position coordinates at vertex i of $\texttt{Sim}_1$ and $\texttt{Sim}_2$ respectively, at the j-th trajectory and time step t.

B. Sim-to-Real Setting

Under a sim-to-real scenario, measuring full-state information is impractical, as it would require sensorizing every point of the target object throughout its motion. In the sim-to-real scenario, we record the motion of our robot with a marker-based motion capture system, which provides us with sparse partial information, but this data may lie in a different coordinate frame than our simulation environment. Therefore, we perform rigid registration to transform the frame of reference of the raw measurement data. We first use the undeformed state marker measurements of the object to define simulated marker locations. Subsequently, we estimate an optimal rotation matrix and translation vector between the measured markers and the recorded data at the corresponding undeformed state. Finally, we apply this transformation to the collected data and interpolate the transformed markers with

$$\mathbf{x}(\mathbf{q}) = \boldsymbol{\alpha}^T \mathbf{e}(\mathbf{q}) + s\mathbf{n} \quad (5)$$

where $\mathbf{n}$ is the unit normal vector to the closest surface mesh element, s is the distance between the transformed marker to the element, $\boldsymbol{\alpha}$ is a barycentric coordinate vector w.r.t. the neighboring surface element nodes $\mathbf{e}$. During forward passes of the simulation we can compute simulated marker positions by interpolating the corresponding surface mesh element with (5).

Next, we create our augmented dataset of reconstructed full-state information from the sparse partial information, similarly to (2):

$$\mathbf{f}_t^{\text{res}^*} = \arg\min_{\mathbf{f}_t} \left\| \mathbf{x}\left(\texttt{Sim}(\mathbf{s}_t, \mathbf{f}_t^{\text{ext}} + \mathbf{f}_t)\right) - \overline{\mathbf{x}}_{t+1} \right\|_2^2 + \lambda \left\| \mathbf{f}_t \right\|_2^2 \quad (6)$$

where $\mathbf{x}$ returns the simulated markers from the returned full state of the simulator, and $\overline{\mathbf{x}}_{t+1}$ is the transformed real markers at time step $t+1$.

With our residual force dataset now well-defined in the real-world scenario, we have reduced our sim-to-real problem to the same setting as that of the sim-to-sim problem. As such, we solve the simplified problem directly, and train a residual model with the loss as described in (3). As before, we assess the model's performance on the validation set after each training epoch and save the model with smallest validation error. We evaluate the saved model across R test trajectories and compute the mean rollout error between simulated markers and their real counterparts. We rollout the hybrid simulation with our trained network in an auto-regressive manner; starting from the same initial state as the ground truth motion, we feed in the previous position and velocity solutions from the simulation, and add the predicted residual physics to the next forward pass of the simulation. The error is computed based on m markers as follows:

$$\mathcal{E}_x := \frac{1}{R \cdot T \cdot m} \sum_{j=1}^{R} \sum_{t=1}^{T} \sum_{i=1}^{m} \left\| \mathbf{x}_{t,i}^{j} - \overline{\mathbf{x}}_{t,i}^{j} \right\|_2 \quad (7)$$

where $\mathbf{x}_{t,i}^{j}$ denotes the i-th marker position at the j-th trajectory and time step t.

C. Residual Physics Learning

We divide the network architecture into blocks that each contain Multilayer Perceptron (MLP), with skip connections added between blocks. A layer-normalization and Exponential Linear Unit (ELU) activation is applied to the output of each linear layer except the last, and the network is trained based on (3). The input layer takes in the positions, velocities, and actuation forces of the simulated system; note that for unactuated systems, the length of $\mathbf{f}^{\text{ext}}$ will be zero. An overview of the residual physics network is shown in Fig. 2. We standardize each spatial dimension (x, y, z) of our datasets for positions $\mathbf{q}$, velocities $\mathbf{v}$, and actuation forces $\mathbf{f}^{\text{ext}}$ to zero mean and unit standard deviation based on the training data, then apply the same standardization during validation and testing.

We run the sim-to-sim network on the datasets described in Table II and the sim-to-real network based on the datasets described in Table III for 1000 epochs. We set the input and output size of each block the same as our network output size. We optimize the hyperparameters of the network through a Bayesian optimization over the batch size, learning rate, scheduler gamma, number of hidden sizes, forward layers per block, and the number of blocks. The validation loss is used to determine which network architecture to keep during this hyperparameter optimization. Details on hyperparameter ranges can be found in the code repository.

D. System Identification as Baseline

In the sim-to-real setting, we run SysID as a baseline to our ResPhys approach. We optimize the Young's modulus E and Poisson's ratio ν to minimize the following objective function

TABLE I
PARAMETER CONFIGURATIONS FOR SIM-TO-SIM BEAMS

Parameters	Incorrect beam	Correct beam [4]
DoFs	528	528
Poisson ratio	0.45	0.499
Young's modulus	215 kPa	264 kPa

on the training dataset with R trajectories, based on the distance between transformed real markers $\overline{\mathbf{x}}_t$ and simulated markers $\mathbf{x}_t$:

$$\mathcal{L}_s := \frac{1}{2R}\sum_{j=1}^{R}\sum_{t=1}^{T}\left\|\mathbf{x}\left(\mathtt{Sim}(\mathbf{s}_t^j, \mathbf{f}_t^{\text{ext},j}; E, \nu)\right) - \overline{\mathbf{x}}_t^j\right\|_2^2 \quad (8)$$

E. Sim-Free Method as Baseline

We use the network architectures described before in residual physics, but instead of predicting corrective forces, this data-driven baseline predicts the next state directly; a mapping from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$. This method serves as a direct baseline to showcase the effectiveness of our model-based residual physics. We similarly optimize the sim-free hyperparameters, and test the networks with lowest validation error.

V. RESULTS

We discuss two continuum deformable structures in the sim-to-real setting: A clamped soft beam and a soft robotic arm called SoPrA [34]. The beam has measurements 10 cm × 3 cm × 3 cm, and the arm is 30 cm in length, with an outer tip diameter of 3 cm. SoPrA is made with the same silicone elastomer as the soft beam, and it has six fiber-reinforced chambers. Fiber reinforcement structures are useful in preventing excessive inflation of fluidic actuators but introduce composite material anisotropy, making it an interesting application domain for our residual physics framework.

All results were performed on a computer with a 32-Core AMD Ryzen Threadripper 3970X CPU and an RTX 3090 GPU. The GPU was used for training, while the CPU was used for all inference scenarios to minimize overhead since DiffPD is a pure CPU-based simulation framework.

A. Sim-to-Sim for Soft Beam

In our first experiment, we test to see if two soft clamped beams, one with "correct" and one with "incorrect" material parameters as shown in the Table I, can be translated between each other *via* residual physics. Since we employ a linear corotational model, the material is not particularly complex, and the problem should be tractable. However, this setting also allows us to analyze our algorithm with granularity.

We consider two motion patterns for the simulated clamped beam displayed in Fig. 3. In the first one, we apply a force to the tip of the beam, wait for it to reach a steady state, and release it to observe the oscillations. In the second pattern, we twist the beam at varying angles within the range of $[\frac{\pi}{6}, \pi]$, then release the beam. We apply the same tip forces for the oscillating beam as the weights we use in the real experiments in Section V-C.

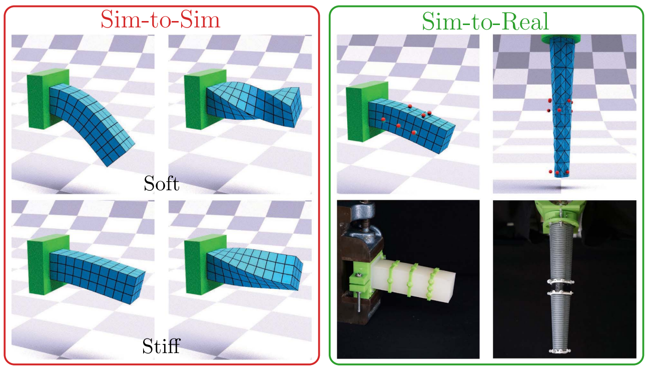

Fig. 3. Sim-to-sim experiments include oscillating and twisting beams, where we either apply a weight at the tip or twist the beam and release this constraint to observe a desired motion trajectory. The sim-to-real experiments show the same passive oscillating beam and a pneumatic soft arm as an actuated robot.

TABLE II
SIM-TO-SIM EXPERIMENTAL CONFIGURATIONS AND RESULTS

	Oscillating Beam	Twisting Beam
Step size (s)	0.01	0.01
Time Steps	150	100
Training Trajectories	9	10
Validation Trajectories	2	2
Testing Trajectories	5	8
Simulation Error (mm)	4.488 ± 1.468	4.362 ± 1.541
SimFree Error (mm)	0.035 ± 0.070	0.038 ± 0.147
ResPhys Error (mm)	$\mathbf{0.004 \pm 0.010}$	$\mathbf{0.004 \pm 0.011}$

We choose $\lambda = 10^{-4}$ in (2) and (6) when we optimize residual forces for all the beam experiments. We initialize our neural network following the discussion of Section III.

In Table II, we report the mean rollout error in (4) for those test trajectories. As expected, both purely data-driven and residual physics approaches can learn these simple dynamics effectively, though residuals are easier to learn and result in more accurate final trajectories.

B. Full State Reconstruction From Sparse Markers

As we collect sparse partial information from a marker-based system, we design an experiment in simulation to investigate how the number of markers influences the reconstruction of the full state information. The previous experiment in Section V-A was performed using full state information; in the following, we assume only to have access to a subset of surface vertices that artificially represent the motion markers we would use in the real world.

We choose a single trajectory with a tip force caused by a 50 g mass for the oscillating beam. We solve (6) for an increasing number of randomly sampled motion markers, starting from a single marker and ending at 128 markers, where the total number of surface vertices for this mesh is 140. We uniformly randomly sample each subset of markers 10 times and optimize (6) from scratch.

We observe in Fig. 4 that the median error drops below 0.1 mm starting from 4 markers, though the spread stays high until 10

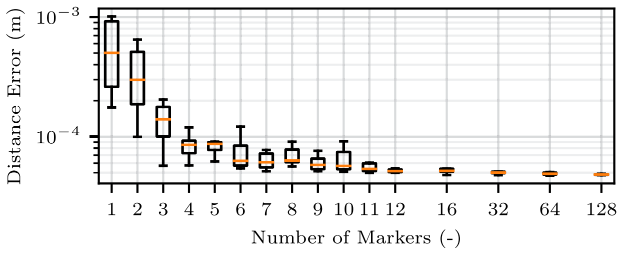

Fig. 4. Box plot of displacement error in (4), varying the number of markers that are available for the residual forces in (6). The orange line is the median of 10 samples, and the box extends from the lower to the upper quartile of the samples.

TABLE III
SIM-TO-REAL EXPERIMENTAL CONFIGURATIONS

Model	Beam	SoPrA
DoFs	528	1482
Poisson ratio	0.45	0.45
Young's modulus (kPa)	215	215
Step size (s)	0.01	0.01
Markers	10	12
Time Steps	140	1000
Training Trajectories	9	35
Validation Trajectories	2	5
Testing Trajectories	5	10

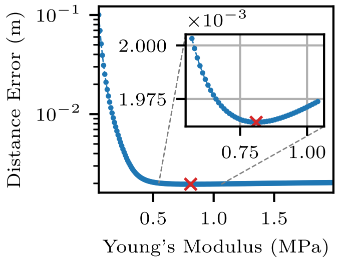

Fig. 5. System identification performed with grid search.

markers. This sim-to-sim experiment was designed to verify that our pipeline not only works for full-state information, but also when only sparse observations are available, such as in the sim-to-real scenarios that follow.

C. Sim-to-Real for Soft Beam

We set up a simulator for the beam following the parameters in Table III. Our objective is to match the position of simulated markers to the corresponding real markers. To collect motion data of the beam, we attach a set of 17 known weights ranging between 50210 to the tip of the beam. After the beam reaches a steady state under the weight, we release it and let it oscillate freely. To track the motion of the soft structures, we use a motion capture system (Miqus M3, Qualisys) that runs at 100 Hz.

We run SysID with a Youngs' modulus within 0.052 and Poisson's ratio within −0.9990.499. Our final optimized parameters converge to 810 kPa and 0.499, respectively. The Young's modulus value is much larger than the value reported by the manufacturers. Hence to validate and understand our findings, we run a system identification grid search at resolution 10 kPa

TABLE IV
MEAN ROLLOUT ERROR BETWEEN SIMULATED AND REAL MOTION MARKERS FOR PASSIVE AND ACTUATED SOFT STRUCTURES

Experiments	Beam Error (mm)	SoPrA Error (mm)
Original	5.46 ± 1.78	7.50 ± 3.69
SysID	1.32 ± 1.38	7.31 ± 3.54
SimFree	0.63 ± 0.49	8.35 ± 4.03
ResPhys	$\mathbf{0.52} \pm \mathbf{0.48}$	$\mathbf{5.77} \pm \mathbf{3.00}$

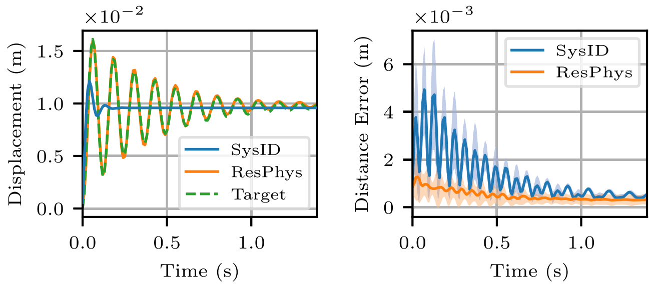

Fig. 6. Sim-to-real results on the soft beam. (Left) Single test trajectory showing displacement in axis of oscillation averaged over motion markers. (Right) Mean and standard deviation of errors on all test trajectories plotted over time.

as depicted in Fig. 5 on the left. The optimal Young's modulus value is obtained at 810 kPa, aligning with our gradient-based optimization result and highlighting the flat nature of the objective landscape. The optimal value is far from ground truth, highlighting the mismatch between the simulation and the physical world.

Building the augmented dataset described in Section IV-B requires us to know the initial state of the beam under weights, so for each weight, we start from an undeformed state $\mathbf{s}_1$ and optimize a series of virtual forces $\mathbf{f}_t^v$ such that after T_v steps we match the initial marker positions $\overline{\mathbf{x}}_{\text{init}}$:

$$\mathcal{L}_{\text{init}} := \sum_{t=1}^{T_v} \left\|\mathbf{x}\left(\texttt{Sim}(\mathbf{s}_t, \mathbf{f}_t^v)\right) - \overline{\mathbf{x}}_{\text{init}}\right\|_2^2 + \lambda \left\|\mathbf{f}_{1\ldots T_v}^v\right\|_2^2 \quad (9)$$

T_v should be chosen to reach a steady state, which we set to $T_v = 140$. This is the static analog to (6). The solution to the optimization problem $\mathbf{s}_{T_v}$ is our ground truth's initial position, from which we build the augmented dataset and train the network as described in Section IV-B.

A quantitative error evaluation between simulation and reality is presented in Table IV. We show a significant improvement of residual physics over system identification, not only via a lower average error (decreased by 60.3%) but also a consistently more robust performance through lower standard deviation in test trajectories. We visualize one test trajectory in Fig. 6 on the left. Our residual physics framework hereby helps to overcome the numerical damping problem and captures real-world dynamics better than SysID. We also plot the errors at each time step in Fig. 6 on the right. Our framework reduces the error at each step and has a smaller deviation than the simulation based on SysID.

Further, we find that our model can accelerate simulations for stiff systems. Our framework allows us to use suboptimal but non-stiff system parameters and correct the errors that these

TABLE V
TIME BENCHMARK OF SIMULATION WITH SUBOPTIMAL PARAMETERS (ORIGINAL), OPTIMAL PARAMETERS (SYSID), AND SUBOPTIMAL PARAMETERS WITH RESPHYS

Simulation	Time per Trajectory (s)
Original Simulation	0.711 ± 0.002
System Identification	2.218 ± 0.005
Residual Physics Simulation	1.393 ± 0.004
Residual Network Inference	0.112 ± 0.002

Statistics taken over 5 test trajectories.

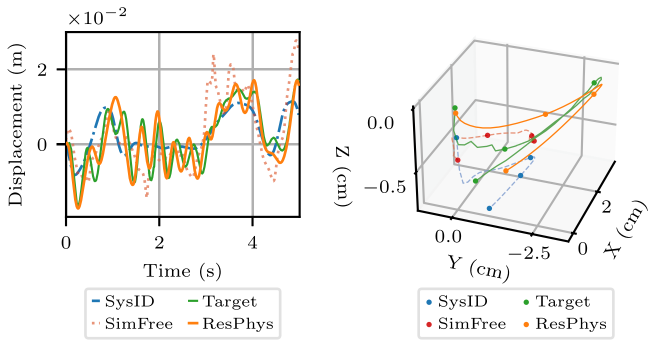

Fig. 7. Sim-to-real results on the soft arm. (Left) Single 5s-excerpt test trajectory showing displacement in y-axis, averaged over motion markers. (Right) 3D displacement plot of the same trajectory between 0 s and 0.46 s, with points along the trajectory spaced 0.15 s apart. For this particular segment of the trajectory, the average distance error for SysID is 1.34 mm, for DD 1.40 mm, and for ResPhys 0.73 mm.

parameters induce *via* the residual forces. From Table V, we observe that achieving an accurate simulation using SysID slows down the simulation by approximately $3.12\times$, yet for ResPhys, it is only $1.96\times$, while achieving a much higher accuracy than SysID. We note that the hybrid simulation has additional overhead compared to the base simulation, due to the injected forces adding numerical stiffening.

D. *Sim-to-Real for Pneumatically-Actuated Soft Arm*

We further test our framework on SoPrA [34], which presents a more challenging scenario due to its increase in mesh size, pneumatic actuation, hard-to-model anisotropic fiber reinforcements around the actuation chambers, and a higher likelihood of hardware fabrication errors. We actuate the arm using random pressure sequences generated from a multivariate normal distribution. We tune the covariance of the distribution such that the resulting pressure trajectories are smooth. Between the collection of each trajectory, we ensure that the system has returned to its unactuated steady state. We clip the commanded pressures to 20 kPa to avoid SoPrA inflating too much and prevent DiffPD from diverging without modeling fiber reinforcements.

Similar to the sim-to-real beam experiments, we first perform rigid registration to transform motion marker data into our simulation environment and then optimize the augmented dataset by choosing $\lambda = 10^{-5}$ in (6). However, different from the beam experiment, we now have non-zero pressure actuation forces $\mathbf{f}_t^{\text{ext}}$ in the network input. These forces are computed from

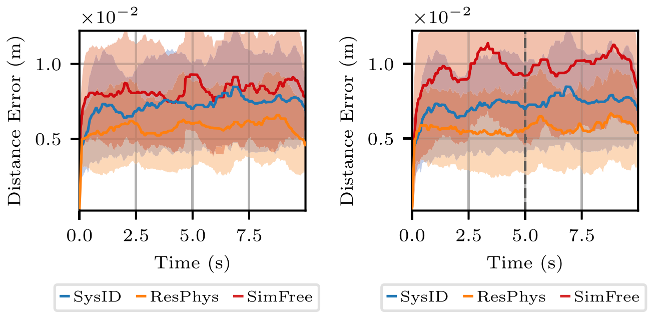

Fig. 8. Mean and standard deviation over all test trajectories on the soft arm. For readability, curves are smoothened using a median filter with a window size of 1 s. (Left) Train/test trajectories are both 10 s. (Right) Train on 5 s and test on extended 10 s.

TABLE VI
ROLLOUT ERROR OF MARKERS FOR 5 s (SAME LENGTH AS TRAINING TRAJECTORIES) AND EXTRAPOLATED TO 10 s (DOUBLE THE TRAINING LENGTH)

Experiments	SysID (mm)	Data (mm)	ResPhys (mm)
Test 5 s	7.51 ± 3.63	9.54 ± 5.60	5.79 ± 3.09
Test 10 s	7.57 ± 3.59	9.89 ± 5.02	6.10 ± 3.22

Mean and standard deviation taken over all test trajectories.

the pressure sequences and applied on the inner faces of the pneumatic chambers of the arm.

The final optimized SysID Young's modulus is 237630 and Poisson's ratio 0.4194, but we observe little improvement in the error on test trajectories in Table IV. Using residual physics, the error is reduced by 21.1%. The qualitative performance on an example trajectory is shown in Fig. 7 and the average errors over all test trajectories in Fig. 8 on the left. We note the poor performance of the purely data-driven approach in this actuated robotic setup, highlighting the benefit of the underlying simulator for the residual physics approach in complex dynamical scenarios.

Typically, machine learning models relying solely on data-driven approaches struggle to predict results in unseen domains accurately. We conduct an extrapolation experiment to test our framework's predictive capability on a longer time horizon. We keep all the training parameters and network architectures the same while only including each trajectory's first 500 time steps in the training set. Afterwards, we examine the model performance in predicting the last 500 time steps. Though prediction accuracy is decreased, our approach still outperforms both baselines as shown in Table VI and Fig. 8 on the right. Our method shows stable long-time horizon predictions but with slowly increasing errors.

VI. CONCLUSION AND FUTURE WORK

We demonstrated a hybrid residual physics framework for high-dimensional soft robots that combines numerical solvers with deep learning for residuals. By leveraging differentiable simulators and learned models, we eliminate the need for intricate domain-specific knowledge and full-state information about the robot. Instead, we shift our reliance to sparse marker data, simplifying the process for practitioners. Our framework helps reduce simulation errors under sim-to-sim and sim-to-real scenarios. We demonstrate its efficacy on passive and actuated soft structures such as a beam and pneumatic arm, showing that it consistently outperforms the system identification and data-driven baselines.

One drawback of our framework is the computationally expensive optimization procedure in the data pre-processing phase. This drawback, however, can be alleviated by more efficient simulators and parallel solving of independent problems. A second drawback is the generalization range of our method: if test data is sufficiently out of distribution, learned dynamics do not generalize. In additional experiments, we found that generalization suffered on systems with $1.75\times$ the internal actuation pressure. Future work should examine how to generalize beyond bounded training data and how to handle novel dynamical events, such as contact interactions.

Our results reveal some potential directions for future exploration. Since the network is provided a time sequence of motion data and we autoregressively call the network at inference time, we necessarily accumulate errors over longer trajectories. A data-driven model for long sequences would be suitable to address this problem, but due to the high DoFs of soft robots, it is hard to train the sequence model directly on the state of the soft robot. A powerful low-dimensional latent representation would provide a promising future avenue for investigation. Lastly, the question of generalizability should be addressed for varying shapes of soft robots. Until now, residual learning frameworks have been limited to single geometries. In future work, we hope to develop a single residual physics network applicable across various soft robot morphologies, proving the same level of applicability as numerical solvers while easing the need for laborious modeling.

References

[1] K. Junge, C. Pires, and J. Hughes, “Lab2Field transfer of a robotic raspberry harvester enabled by a soft sensorized physical twin,” *Commun. Eng.*, vol. 2, no. 1, 2023, Art. no. 40.

[2] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, “Exploration of underwater life with an acoustically controlled soft robotic fish,” *Sci. Robot.*, vol. 3, no. 16, 2018, Art. no. eaar 3449.

[3] J. Z. Zhang et al., “Sim2real for soft robotic fish via differentiable simulation,” in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 12598–12605.

[4] M. Dubied, M. Y. Michelis, A. Spielberg, and R. K. Katzschmann, “Sim-to-real for soft robots using differentiable fem: Recipes for meshing, damping, and actuation,” *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5015–5022, Apr. 2022.

[5] L. Ljung, *System Identification: Theory for the User*. London, U.K.: Pearson Educ., 1998.

[6] A. Ajay et al., “Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing,” in *Proc. 2018 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3066–3073.

[7] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, “TossingBot: Learning to throw arbitrary objects with residual physics,” *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1307–1319, Aug. 2020.

[8] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, “Position based dynamics,” *J. Vis. Commun. Image Representation*, vol. 18, no. 2, pp. 109–118, 2007.

[9] S. Bouaziz, S. Martin, T. Liu, L. Kavan, and M. Pauly, “Projective dynamics,” *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–11, 2014.

[10] M. Macklin and M. Muller, “A constraint-based formulation of stable neo-hookean materials,” in *Proc. Motion, Interaction Games. Virtual Event*, Switzerland, Nov. 2021, pp. 1–7, doi: 10.1145/3487983.3488289.

[11] T. Schneider, J. Dumas, X. Gao, M. Botsch, D. Panozzo, and D. Zorin, “Poly-spline finite-element method,” *ACM Trans. Graph.*, vol. 38, no. 3, Mar. 2019, doi: 10.1145/3313797.

[12] M. Li et al., “Incremental potential contact: Intersection- and inversion-free large deformation dynamics,” *ACM Trans. Graph.*, vol. 39, no. 4, 2020, Art. no. 49.

[13] M. Gazzola, L. Dudte, A. McCormick, and L. Mahadevan, “Forward and inverse problems in the mechanics of soft filaments,” *Roy. Soc. Open Sci.*, vol. 5, no. 6, 2018, Art. no. 171628, doi: 10.1098/rsos.171628.

[14] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, and M. Gross, “Computational design of actuated deformable characters,” *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–10, 2013.

[15] C. F. Graetzel, A. Sheehy, and D. P. Noonan, “Robotic bronchoscopy drive mode of the auris monarch platform,” in *Proc. 2019 Int. Conf. Robot. Automat.*, 2019, pp. 3895–3901.

[16] B. G. Cangan, S. E. Navarro, B. Yang, Y. Zhang, C. Duriez, and R. K. Katzschmann, “Model-based disturbance estimation for a fiber-reinforced soft manipulator using orientation sensing,” in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 9424–9430.

[17] R. K. Katzschmann et al., “Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer,” in *Proc. 2019 IEEE 2nd Int. Conf. Soft Robot.*, 2019, pp. 717–724.

[18] T. Du et al., “DiffPD: Differentiable projective dynamics,” *ACM Trans. Graph.*, vol. 41, no. 2, pp. 1–21, Nov. 2021, doi: 10.1145/3490168.

[19] Y. Hu et al., “ChainQueen: A real-time differentiable physical simulator for soft robotics,” in *Proc. 2019 IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6265–6271.

[20] T. Du, J. Hughes, S. Wah, W. Matusik, and D. Rus, “Underwater soft robot modeling and control with differentiable simulation,” *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4994–5001, Mar. 2021.

[21] K. M. Jatavallabhula et al., “GradSim: Differentiable simulation for system identification and visuomotor control,” in *Proc. Int. Conf. Learn. Representations*, 2021.

[22] P. Ma, T. Du, J. B. Tenenbaum, W. Matusik, and C. Gan, “Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation,” in *Proc. Int. Conf. Learn. Representations*, 2021.

[23] A. Allevato, E. S. Short, M. Pryor, and A. Thomaz, “TuneNet: One-shot residual tuning for system identification and sim-to-real robot task transfer,” in *Proc. Conf. Robot Learn.*, 2020, pp. 445–455.

[24] J. Hwangbo et al., “Learning agile and dynamic motor skills for legged robots,” *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau5872.

[25] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[26] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, “NeuralSim: Augmenting differentiable simulators with neural networks,” in *Proc. 2021 IEEE Int. Conf. Robot. Automat.*, 2021, pp. 9474–9481.

[27] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer, “Sim-to-real transfer with neural-augmented robot simulation,” in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 817–828. [Online]. Available: https://proceedings.mlr.press/v87/golemo18a.html

[28] K. Chin, T. Hellebrekers, and C. Majidi, “Machine learning for soft robotic sensing and control,” *Adv. Intell. Syst.*, vol. 2, no. 6, 2020, Art. no. 1900171.

[29] P. Battaglia et al., “Interaction networks for learning about objects, relations and physics,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 4509–4517.

[30] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, “Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids,” in *Proc. Int. Conf. Learn. Representations*, 2019.

[31] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, “Learning mesh-based simulation with graph networks,” in *Proc. Int. Conf. Learn. Representations*, 2021.

[32] C. Laschi, T. G. Thuruthel, F. Lida, R. Merzouki, and E. Falotico, “Learning-based control strategies for soft robots: Theory, achievements, and future challenges,” *IEEE Control Syst. Mag.*, vol. 43, no. 3, pp. 100–113, Mar. 2023.

[33] P. Ma et al., “Learning neural constitutive laws from motion observations for generalizable pde dynamics,” in *Proc. Int. Conf. Mach. Learn.*, 2023.

[34] Y. Toshimitsu, K. W. Wong, T. Buchner, and R. Katzschmann, “SoPrA: Fabrication & dynamical modeling of a scalable soft continuum robotic arm with integrated proprioceptive sensing,” in *2021 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 653–660, doi: 10.1109/IROS51168.2021.9636539.