Learning physics-informed simulation models for soft robotic manipulation: A case study with dielectric elastomer actuators

Citation for published version:
Lahariya, M, Innes, C, Develder, C & Ramamoorthy, S 2022, Learning physics-informed simulation models for soft robotic manipulation: A case study with dielectric elastomer actuators. in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2022.* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Institute of Electrical and Electronics Engineers (IEEE), pp. 11031-11038, The 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, Kyoto, Japan, 23/10/22. https://doi.org/10.1109/IROS47612.2022.9981373

Digital Object Identifier (DOI):
10.1109/IROS47612.2022.9981373

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of the International Conference on Intelligent Robots and Systems (IROS) 2022

Learning physics-informed simulation models for soft robotic manipulation: A case study with dielectric elastomer actuators

Manu Lahariya[1]*, Craig Innes[2], Chris Develder[1] and Subramanian Ramamoorthy[2]

***Abstract*— Soft actuators offer a safe and adaptable approach to robotic tasks like gentle grasping and dexterous movement. Creating accurate models to control such systems however is challenging due to the complex physics of deformable materials. Accurate Finite Element Method (FEM) models incur prohibitive computational complexity for closed-loop use. Using a differentiable simulator is an attractive alternative, but their applicability to soft actuators and deformable materials remains under-explored. This paper presents a framework that combines the advantages of both. We learn a differentiable model consisting of a material properties neural network and an analytical dynamics model of the remainder of the manipulation task. This physics-informed model is trained using data generated from FEM, and can be used for closed-loop control and inference. We evaluate our framework on a dielectric elastomer actuator (DEA) coin-pulling task. We simulate DEA coin pulling in FEM, and design experiments to evaluate the physics-informed model for simulation, control, and inference. Our model attains $\leq 5\%$ simulation error compared to FEM, and we use it as the basis for an MPC controller that outperforms (i.e., requires fewer iterations to converge) a model-free actor-critic policy, a heuristic policy, and a PD controller.**

***Index Terms*— Dielectric elastomer actuators, Differentiable simulator, Finite element methods, Model predictive control, Neural Networks, Physics based machine learning, Soft Actor-Critic, Soft robotics**

I. INTRODUCTION

Soft robotic actuators provide a safe, adaptive, low-cost solution for manipulation tasks such as gripping and motion [1]. Precision manipulation using soft actuators however is a major challenge, as it requires modeling the deformable actuator behaviour within the context of the manipulation task [2]. Such models are then used to learn accurate control strategies via simulation [3]. Recently *differentiable simulators* have been used to learn controllers in closed-loop scenarios by allowing the use of gradient-based optimization methods (e.g., Model Predictive Control, MPC) [4]. They have also been used for inference and data generation tasks.

Simulating deformable robots and contact rich manipulation is expensive [3]. Traditional methods model such dynamics by decomposing their geometry. For example, *Position Based Dynamics* approximates multi-body physics by deconstructing the system into particles [5]. However, these methods fail to define the underlying physics, making it difficult to meaningfully interpret or constrain the particles. The continuum mechanics and contact dynamics of deformable materials are difficult to model with such approximate methods, leading to physically unrealistic results. Physically accurate simulation of soft robotic manipulation requires modeling the underlying equations, defined by complex Ordinary/Partial Differential Equations (ODEs/PDEs).

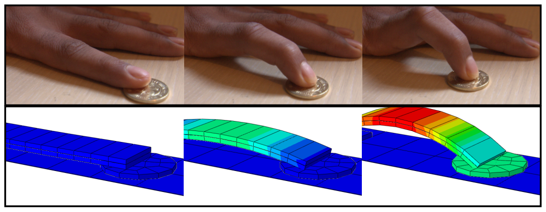
Fig. 1. A coin is pulled with a dielectric elastomer actuator (DEA) — a soft actuator that deforms under electric actuation. Our proposed control framework learns accurate physics-informed differentiable simulators and model-based control for this kind of soft robot manipulation.

Finite Element Methods (FEMs) provide a numerical method for solving such equations. Yet despite the ability of FEMs to accurately model such phenomena, integrating FEM simulation with closed-loop control is challenging due to their computationally expensive meshing: unless the meshes are dense and cover the domain, fidelity is poor.

This paper's key idea is to generate data from an accurate (but slow) FEM model to learn an approximate (but fast) physics-informed model f for soft robotic manipulation. Our framework uses f as a differentiable simulator for simultaneous closed-loop control and inference. Our model f is composed of two parts: a *material network* m — a neural network approximating deformable material behaviour (e.g., hyperelastic) — and the *dynamics* d — equations representing the physical context of manipulation task (e.g., motion).

We apply our framework to a soft robotic pulling task using Dielectric Elastomer Actuators (DEAs). DEAs are soft actuators made using electroactive polymers that convert electrical work to mechanical work via expanding or bending motion. In our task, the goal is to pull a stationary coin by deforming the free end of the DEA (Fig. 1). We learn the physics-informed model for this pulling motion f, and evaluate its accuracy as a simulator against the FEM simulations. For control, we use the differentiability of f to learn a model-based control policy (using the MPC solver defined in [6]) and infer the parameters of the system's dynamics.[1]

Our main contributions are: (i) a closed-loop control

[1]Authors are with IDLab, Ghent University – imec, Technologiepark-Zwijnaarde 126, 9052 Ghent, Belgium,
[2]Authors are with the School of Informatics, University of Edinburgh, 10 Crichton St, EH8 9AB, United Kingdom,
*Corresponding Author: e-mail: manu.lahariya@ugent.be.

[1]For the case study of DEA pulling, the mass of coin m_c and kinetic friction coefficient μ_b are inferred. For details, refer to Section III-B.

framework for soft robotic manipulation, that uses a differentiable physics-informed model f trained using FEM (Section II), (ii) the design of an exemplary DEA pulling task (Section III), that is simulated in FEM (Section IV), and (iii) performance evaluation of model f and its use in closed-loop control. For the latter, we compare simulation accuracy of f with both a FEM model and a baseline neural network. Additionally, we compare the model-based control policy (MPC with f), with (i) a model-free control policy (learnt using the Soft Actor-Critic SAC algorithm [7]), (ii) a PD control policy (evaluated previously for DEA control [8]), and (iii) a heuristic control policy[2] (inspired by typical soft-robotic control policies [3]). We design experiments (Section V-B) across 8 DEA pulling setups to evaluate our framework and answer the following questions:

(Q1) How to define f using the physical laws of the system? What is the simulation accuracy of f in a system with new *unknown* parameters (e.g., frictional coefficient)?

(Q2) What is the performance of model-based control policy (based on f), compared to other control policies?

(Q3) What is the accuracy of the inferred model parameters?

Our results (Section VI) show that f provides $\leq 5\%$ simulation error compared to FEM. Further, in closed-loop control, an MPC using f outperforms all other policies, while we simultaneously infer system properties (mass of the coin) with $\leq 10\%$ inference error.

A. Related Work

Soft robots are inspired by biological systems, where animals use muscles to achieve safe actuation and control [1]. Engineers use soft actuators to develop similar safe, quick, adaptable, and precise robotic manipulation [2]. These soft actuators generate mechanical work under a specific actuation, e.g., shape memory alloys respond to thermal actuation, hydraulic actuators respond to pressure, etc. Learning control for soft actuators requires accurate simulation models that are used inside the control loop [4]. Designing simulator models for soft actuators is a challenging task, traditionally using particle based models (e.g., liquids [9]). In recent years, researchers are using FEM modeling that allows highly accurate modeling of deformable materials (e.g., fabric [10], composite materials [11]).

Dielectric elastomers (DE) are electroactive polymers that produce deformation under the influence of an external electric field. DEA are soft actuators that use thin layers of DE materials to achieve actuation under the stimulus of electric activation. DEAs provide fast and large deformation, are lightweight, and have a high energy density, which makes them promising candidates for soft robotic applications [12]. Hence, DEA has been explored to design soft robotic grippers [13], underwater robots [14], crawling robots [15], etc.

There have been several previous approaches to modelling DE behaviour. A simplified finite element analysis of a dielectric bending actuator is performed in [13]. FEM models

[2]The heuristic policy linearly ramps up actuation voltage until the 'episodic' task terminates. For details, refer to Section V-B.

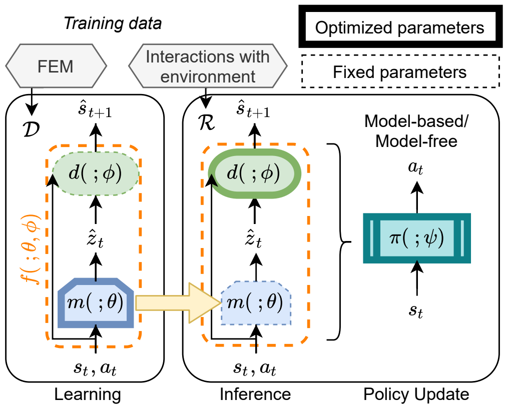

Fig. 2. Training the model f, that represents simulator for the manipulation task. Material network m is optimized during learning, dynamics d are inferred during closed-loop environment interactions.

for DE material have been explored, where deformation in unimoph DEA with inhomogeneous geometry modeled in [16] uses piezoelectric elements. The FEM model for a gripping actuator presented in [17] is based on a custom user defined material. All the above methods focus on modeling the DEA behaviour in isolation. They thus lack an understanding of the task context in which the DEA manipulator is being used. In contrast, our method simulates the complete manipulation task using FEM. This allows us to learn the task specific context as well as the behaviour of DEA.

An accurate simulator of the manipulation task can assist in learning a controller. For example, a simulator based on position based dynamics (defined using particle interactions) is used in [9] to develop control strategies for pouring liquid. A FEM based differentiable simulator proposed in [18] is used to learn control strategies for cutting. The above methods are designed to control one specific manipulation task (e.g., in [19], the model is explicitly engineered for cutting). In contrast, the control framework we propose can be used for any robotic manipulation task, as long as it can be simulated in FEM. Furthermore, we show that our trained physics-informed model f can infer properties of new *unknown* setups, thus it can adapt during the closed-loop control.

II. Control Framework

The objective of our control framework is to learn a control policy π for a manipulation task. Figure 2 shows the closed-loop control design using the trained physics-informed model f of the manipulation task along with policy π.

A model-based control approach utilizes a forward model of the system: $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, where $\mathcal{S}$ is the state space, and $\mathcal{A}$ is the control action space. For each timestep t, the state is $s_t \in \mathcal{S}$, and the control action is $a_t \in \mathcal{A}$. For MPC, the optimal control actions at each timestep is estimated by solving the optimization problem defined in Eq. (1), where a_{init} is the initial action. We particularly choose to use a physics-informed f, which is differentiable, and allows us to use gradient-based methods to solve this optimization problem (such as finite-horizon iterative Linear Quadratic Regulator, iLQR [6]). The objective of the control (e.g., get to a target location) is used to define the cost function $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (e.g., distance from the target location). For example, in DEA pulling, cost function $\mathcal{C}$ is defined for the objective of achieving target state (i.e., the target location of the coin) with penalty on control actions to minimize actuation voltage of the DEA.[3]

$$\begin{aligned} &\underset{s_{1:T}\in\mathcal{S}, a_{1:T}\in\mathcal{A}}{\arg\min} \sum_{t=1}^{T} \mathcal{C}(s_t,\ a_t) \\ &\text{s.t.} \quad s_{t+1} = f(s_t, a_t) \quad \text{and} \quad a_1 = a_{\text{init}} \end{aligned} \tag{1}$$

The physics of the robotic manipulation task includes, (i) the physical laws of the deformable material behaviour, e.g., electromechanical/hyperelastic behaviour, characterized by high order ODEs/PDEs that are computationally complex, and, (ii) the physical laws built according to the context of the manipulation task, e.g., sliding motion laws, or gravity. In modeling these physics, interaction variables z are introduced to describe the contact properties (e.g., force, stress, pressure) between the deformable material and its surroundings. In particular, for DEA pulling, z are the forces exerted by the DEA actuator on the contact surface.

We simulate the manipulation task using a FEM model to numerically solve the associated physics equations. In this model, state s_{t+1} and interaction variables z_t are simulated, given s_t and a_t. The FEM model for DEA pulling is described in Section IV. How the simulated data is then used to train a physics-informed model f is described below.

A. Physics-informed model (f)

The physics-informed model f has two parts:

(i) The material network (m): a function approximator (i.e., neural network) with weights θ that estimates the interaction variables $\hat{z}$ (Eq. (4)). These interaction variables (z) characterize deformable material behaviour in the manipulation task (e.g., forces by DEA on contact surface).

(ii) The dynamics (d): the physical laws characterizing the motion/dynamics of the system in form of mathematical equations (e.g., a system of linear equations or ODEs/PDEs representing sliding or gripping). The dynamics d estimate the next state using interaction variables z, state s, and action a (Eq. (3)). Parameters ϕ describe the system's physical properties, e.g., the mass of coin.

Thus, we can write the model f as in Eq. (2).

$$\hat{s}_{t+1} = f(s_t,\ a_t;\ \theta,\ \phi) \tag{2}$$

$$f(s_t,\ a_t;\ \theta,\ \phi) = d(\hat{z}_t,\ s_t,\ a_t;\ \phi) \tag{3}$$

$$\hat{z}_t = m(s_t,\ a_t;\ \theta) \tag{4}$$

The material model m of a deformable material can describe an actuator (e.g., DEA), or the manipulated object (e.g., cloth) depending on the manipulation task. For example, in DEA pulling, m describes the actuator behaviour of a unimorph DEA (Section IV).

[3]The cost function defined in [6]. For details, refer to Section V-A.

Algorithm 1 Learning
```
Output: Material network $m$;
 1: Randomly initialize weights $\theta$, and fix parameters $\phi$;
 2: while not stopping condition do
 3:    $a_t \leftarrow$ select action at $t$ using a fixed policy;
 4:    $s_{t+1}, z_t = FEM(s_t,\ a_t)$;
 5:    Dataset $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, z_t, s_{t+1})$;
 6: end while
 7: Using all data from $\sim \mathcal{D}$; // Say $|\mathcal{D}| = N$ data samples
 8: $\hat{z}_t = m(s_t,\ a_t;\ \theta)$;
 9: $\hat{s}_{t+1} = f(s_t,\ a_t;\ \theta,\ \phi)$;
10: $\theta \leftarrow \theta - \alpha_\theta \frac{d\mathcal{L}_l(\theta,\ \phi)}{d\theta}$; // Update $\theta$ using $\hat{z}$ and $\hat{s}$, Eq. (5)
11: return $m$
```

Algorithm 2 Control
```
Input: Trained weights $\theta$
 1: Randomly initialize parameters $\phi$, and $\psi$, and fix $\theta$;
 2: $s_1 \leftarrow env.reset()$;
 3: while not stoping condition do
 4:    $a_t = \pi(s_t;\ \psi)$;
 5:    $s_{t+1}, r_t \leftarrow env.step(a_t)$;
 6:    Replay buffer $\mathcal{R} \leftarrow \mathcal{R} \cup (s_t, a_t, r_t, s_{t+1})$;
 7:    if it's time to update then
 8:       Randomly sample $B$ transitions from $\sim \mathcal{R}$;
 9:       // Inference
10:       $\hat{s}_{t+1} = f(s_t,\ a_t;\ \theta,\ \phi)$;
11:       $\phi \leftarrow \phi - \alpha_\phi \frac{d\mathcal{L}_i(\theta,\ \phi)}{d\theta}$; // Update $\phi$ using $\hat{s}$, Eq. (6)
12:       // Policy Update
13:       Update $\psi$ by policy defined updates, e.g., SAC [7];
14:    end if
15: end while
```

This section answers part of Q1 (how to define f using physical laws). The model f is differentiable and captures the physics of the manipulation task in the form of dynamics d. Thus, in addition to its usefulness as a simulator for data generation, it can also be used for inference, and for learning gradient-based optimization control policies.

B. Training and policy synthesis

The physics-informed model f and policy π are learnt in two steps. First, a *Learning* step optimizes the weights θ of the m, using data generated by the FEM model of the task. Second, a *Control* step, where the policy π is learnt via interactions with the environment. These interactions are used to infer parameters ϕ (e.g., coin mass) of dynamics d, which informs f. We then use f to learn π.

a) Learning: The weights θ are optimized by minimizing the loss function based on the error in estimating material model, i.e., m: $(z_t - \hat{z}_t)$, and the error in enforcing dynamics, i.e., d: $(s_t - \hat{s}_t)$. Incorporating the loss encountered in $\hat{s}$ ensures that our model adheres to dynamics d (as shown

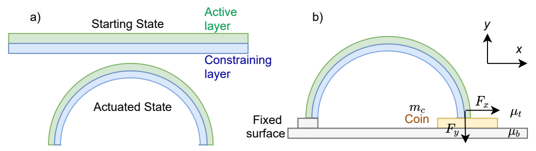

Fig. 3. Unimorph DEA. (a) Unimorph DEA. The active layer expands under influence of electric voltage, causing the bending actuation. (b) Problem setup: DEA is used to move a stationary coin on the surface.

by optimization of physics informed neural networks [20]). Algorithm 1 shows how θ is optimized by fixing parameters ϕ and minimizing the learning loss $\mathcal{L}_l$ (Eq. (5)). A fixed policy is used to select actions a_t (e.g., a random or uniform policy). The learning rate for weights θ is α_θ and number of data samples is N.

$$\mathcal{L}_l(\theta, \phi) = \frac{1}{N}\sum_{t=1}^{N}(z_t - \hat{z}_t)^2 + \frac{1}{N}\sum_{t=1}^{N}(s_t - \hat{s}_t)^2 \quad (5)$$

b) Control: Algorithm 2 provides the pseudocode for closed-loop control of the manipulation task. First, during inference, the parameters ϕ are optimized by minimizing the loss function based on the error in estimating dynamics, i.e., d: $(s_t - \hat{s}_t)$. The inference loss function is provided in Eq. (6). Learning rate for ϕ is γ_ϕ and batch size is B.

$$\mathcal{L}_i(\theta, \phi) = \frac{1}{B}\sum_{t=1}^{B}(s_t - \hat{s}_t)^2 \quad (6)$$

Secondly, we learn the policy π with weights ψ. The updates in ψ are defined using the underlying policy π and its training objective. In case of model-free policy trained using the SAC algorithm, ψ will be updated based on the loss function defined in [7], and the weights ψ represent the weights of the neural network. In case of model-based MPC, we use the differentiable trained model f and gradient-based optimization to estimate the optimal control action. Thus, in case of MPC, we do not need to update the parameters ψ (line 13 of Algorithm 2).

III. Soft Robotic DEA pulling

We design the manipulation task of coin pulling using a unimorph Dielectric Elastomers Actuators (DEAs) to evaluate the framework proposed in Section II. The deformable DEA actuator is made of Dielectric Elastomers (DEs), which are a type of electroactive polymers that produce mechanical strain under the influence of electric voltage. Thus, a DE membrane expands its area when a voltage is applied across its thickness [21].

Figure 3(a) shows a unimorph DEA, with one active and one constraining layer. The active layer expands under externally applied voltage causing the bending motion. The DEA is fixed at one end, and the other end rests freely on a circular coin c. On actuation, the DEA acts as a soft robotic finger, pulling the coin. A controller policy π can be learnt to achieve a certain displacement in the coin. Figure 3(b) shows the 2D view of the setup, where the mass of the coin is m_c, the kinetic friction coefficient between the coin and DEA is μ_t, and the kinetic friction coefficient between the coin and bottom surface is μ_b. The displacement of the coin depends on such parameters of the system. A pulling coin setup C_c is characterized by fixed values of $\{m_c, \mu_t, \mu_b\}$. Setups $C_1, C_2, \ldots$ represent pulling different coins, based on different parameter values.[4]

TABLE I
State and action definition for timestep t

s_t	x_t	Location of the coin along x-axis at time t
	u_t	Velocity of the coin along x-axis at time t
a_t	V_t	Voltage applied on the DEA
	Δt	time difference between t and t+1
z_t	$F_{x,t}$	Force along x-axis by DEA on coin c
	$F_{y,t}$	Force along y-axis by DEA on coin c

The physics-informed model f of the system is defined by the variables shown in Table I. The state of the system at time-step t is characterized by the location x_t and velocity u_t of the coin along the x-axis. The action comprises the voltage (V_t) applied on the DEA and Δt,[5] and the hidden variables are the forces (F_x and F_y) applied by the DEA on the top surface of the coin.

A. Material network (m)

Modeling non-linear properties of DEs require modeling the effects of hyperelasticity and Maxwell stress [21]. On application of voltage V, maxwell stress causes the bending actuation in DEA. The actuated DEA exerts forces F_x and F_y on the top surface of the coin, which results in its motion. The material network used to estimate these forces is defined in Eq. (7). We simulate DEA pulling using FEM, to generate data and optimize weights θ (Section IV).

$$\hat{F}_{x,t}, \hat{F}_{y,t} = m(x_t,\ u_t,\ V_t,\ \Delta t;\ \theta) \quad (7)$$

B. Dynamics (d)

Physical laws of the pulling setup define the system dynamics d (Section II-A). There are two stages during pulling: *static friction* (where forces are applied but there is no motion), and *kinetic friction* (where the applied forces cause motion in coin). A threshold voltage V^T should be used during actuation to achieve motion in the coin (i.e., to get to the stage of kinetic friction).[6] The acceleration A_t in the coin is due to the net force in the direction of the x-axis, given by Eq. (8), where F_μ is the opposing frictional force. We can calculate F_μ using Eq. (9), assuming a linear growth in frictional force during the stage of static friction, and a

[4] We consider the coins to be of fixed dimensions (i.e., fixed volume), and thus change the mass m_c by changing the density ρ of the coin material. For further details, please refer to Section V.

[5] Note that in FEM, the time between successive simulation datapoints may vary.

[6] Voltage required to achieve a minimum displacement in the coin.

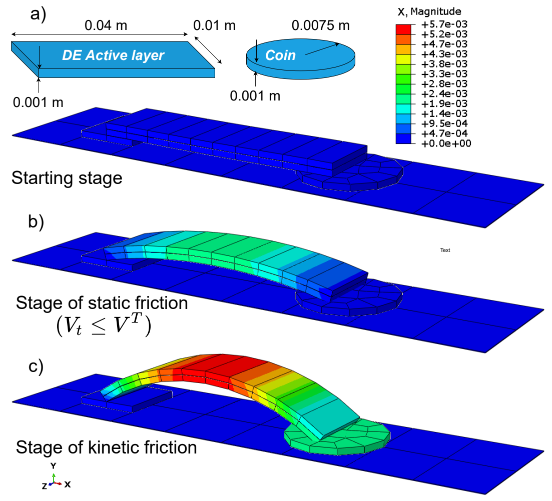

Fig. 4. Finite element model of the DEA pulling, (a) inactive DEA with dimensions, (b) active DEA during the stage of static friction, and (c) motion occurs during the stage of kinetic friction.

no-slip condition on the top surface.

$$\begin{aligned} \hat{F}^{Net}_{x,t} &= m_c A_t \\ &= \hat{F}_{x,t} - F_{\mu,t} \end{aligned} \tag{8}$$

$$F_{\mu,t} = \begin{cases} \mu_b \left(\hat{F}_{y,t} + m_c g\right) & \text{if } V \geq V^T; \\ \dfrac{\mu_b V_t}{V_T} \left(\hat{F}_{y,t} + m_c g\right) & \text{otherwise} \end{cases} \tag{9}$$

where g is the gravitational acceleration ($9.8\,\text{m/s}^2$). We assume a frictional decay in velocity for a moving coin if the DEA actuation is stopped (i.e., $V_t = 0$). These dynamics do not consider non-linear motion in coin (with high DEA actuation voltages, where DEA loses contact with the coin surface). The dynamics d of this pulling setup is characterized by parameters ϕ (Section II), which are (i) mass of the coin (m_c), and (ii) frictional coefficient (μ_b),. While training the material network m, these values are fixed. We will infer these parameters during closed-loop interactions with the environment.

The next location and velocity of the coin, i.e., $\hat{x}_{t+1}$ and $\hat{u}_{t+1}$, are calculated using A_t, x_t, u_t, and Δt and equations of motion. Thus, the dynamics d is a set of linear equations based on the laws of motion.

IV. FEM of DEA pulling

FEM is a numerical method for solving differential equations and mathematical modeling of a physical system. The physical system of DEA pulling consists of a unimorph DEA on the fixed surface and a solid coin (Section III). We use commercially available software ABAQUS [22] to build a 3D model of DEA pulling. Figure 4 shows the simulated setup, during the starting stage and the stages of static and kinetic friction. In the stage of static friction, we clearly see no motion in the coin even when actuating the DEA.

a) DE Material: To model DE material in FEM, we approximate their behaviour using piezoelectric materials elements [23], as there are no commercial FEM packages that provide DE elements out of the box. We modify the piezoelectric finite elements material properties to model the Maxwell stress effect observed in dielectric materials. The Maxwell stress p on the DE membrane is given by Eq. (10) [21]. Similar to DE, piezoelectric materials exhibit strain when in the presence of electric fields. The piezoelectric stress is given by Eq. (11).

$$p = e_0 e_r \left(\frac{V}{z}\right)^2 \tag{10}$$

$$\sigma_{ij} = D^E_{ijkl} \epsilon_{kl} - e_{mij} E_m \tag{11}$$

where V is the applied voltage across thickness z of the DE membrane, the relative permittivity is e_r, and the permittivity of free space is e_0. The piezoelectric elastic stiffness matrix is D^E_{ijkl}, the strain tensor is ϵ_{kl}, the stress coefficient is e_{mij} and the electric potential gradient is E_m. As detailed in [16] the piezoelectric stress becomes approximately equal to the Maxwell stress for a thin membrane, such that the strain e_{zz} in the direction of thickness (z-axis) is given by:

$$e_{zz} = e_r e_0 E_z \tag{12}$$

These piezoelectric elements assume linear elasticity, which is not a limiting factor as we can assume such behaviour for our case of thin DE membranes [16].

A. FEM simulation settings

Figure 4(a) shows the dimensions and assembly of the FEM setup, where the DEA is inactive (i.e., no voltage applied). Mesh is created using an 8-node linear brick element (ABAQUS element type C3D8E), such that each DE membrane has 10 elements.[7] For meshing the coin, ABAQUS's internal meshing strategy is used to generate 20 elements.

For the DE material, the Poisson's ratio is 0.5 and Young's modulus is 0.56 MPa [24]. We use Eq. (12) to calculate e_{zz} = 3.68 and set all other piezoelectric coefficient values to zero. This DE material is used for both active and constraining layers. Elastic behaviour is assumed for the coin. The bottom surface and the fixed end of the DEA are constrained using encastre boundary condition. The top surface of the fixed end assumes no-slip condition.

The coin rests on the frictional surface with a frictional coefficient $\mu_b \in \{0.2, 0.25\}$, which is defined as tangential behaviour in the contact interactions in ABAQUS. Similarly, the free end of the DEA rests on top of the coin with a frictional coefficient $\mu_t \in \{0.5, 0.55\}$. To simulate a real scenario, we include gravitational load in the model (g = $9.8\,\text{m/s}^2$). We do not assume a no-slip condition between the top of the coin and DEA, in contrast to the dynamics described in Section III-B, to keep FEM simulation realistic. The mass of the coin (m_c) is calculated using the volume of

[7]We use a limited number of elements in our mesh due to software limitations.

the coin and density $\rho \in \{7.7,\ 7.8\}\,\text{g/cm}^3$. Thus, the value of m_c rounded up to 2 decimal points is $\in \{1.36,\ 1.38\}\,\text{g}$. The total time simulated in FEM is 1 s, with Δt between consecutive points determined by the internal solver.

An experimental coin setup C is described using the mass of the coin and frictional coefficients (Section III). A total of 8 setups are defined based on the values of $\mu_t,\ \mu_b,$ and m_c denoted by $\{C_1, C_2, \ldots, C_8\}$. FEM models are developed for each of setup. A linearly increasing electric potential load is applied on top surface of the active DE layer (i.e., $V_t \in \{0.0,\ 400.0\}\,\text{V}$ to collect the dataset for each model. This dataset contains the values for all variables described in Table I for each timestep t. Each dataset has approximately 1000-2000 data points. For all setups, the initial location x_t is 0 m and the initial velocity u_t is 0 m/s.

V. Experimental Design

In this section, we detail the experiments designed to evaluate the proposed framework. Using these experiments, we report the high simulation and inference accuracy of f, and develop an effective closed-loop soft robotic controller. For the case of DEA pulling, we evaluate, (i) the accuracy of the f as a simulator, and (ii) the accuracy of f in inference, and (iii) the closed-loop MPC controller that utilizes f.

A. Parameters setting

Each setup C is characterized by three parameters: mass m_c and frictional coefficients μ_b, μ_t (Section IV). For example, in setup C_1, m_c = 1.36 g, μ_b = 0.2, and μ_t = 0.5. During learning, we initialize parameters ϕ (= (m_c, μ_b), see Section III-B) of the dynamics d using the actual values used in the FEM model. We note that μ_t is not used as a parameter in the dynamics of f, however, it is needed for the FEM modeling. For each setup, we set the threshold voltage (V^T) achieve a displacement of -10^{-5} m. Further, we assume the coin loses contact with the DEA for $V_t \geq 300\,\text{V}$.

Physics-informed model f is developed using Pytorch [25] and contains the material network m and the dynamics d (Section II). The material network m is a fully connected neural network with four input nodes, two output nodes, three hidden layers with 64 neurons each, and rectified linear (ReLU) activation functions. The weights are optimized by minimizing Eq. (5) and Eq. (6) using ADAM optimization for 1,000 iterations and a learning rate of 0.001. An early stopping criterion based on validation loss with 0.0 minimum change is included during optimization.

The target state for the controller is $x^T = -1.0\,\text{mm}$, i.e., goal is to achieve a 1 mm displacement. The manipulation task 'episode' terminates when the coin is $\leq$ 0.01 mm from x^T (i.e., $\|x_t - x^T\| <$ 0.01 mm). We average the results for 10 'episodes' for all controllers. Batch size (Algorithm 2) is 256, and Δt is fixed to 0.001 s. For inference, m_c is initialized to 0.001 g and μ_b is initialized to 0.2.

The model-based control policy model f and an MPC solver [6]. For MPC, the number of timesteps is set to 20, LQR iterations to 20, and the penalty of actions is set to

TABLE II
6 Simulation Experiment Sets

	1	2	3	4	5	6
C_{train}	C_1	C_2	C_3	C_4	C_5	C_6
C_{val}	C_2	C_3	C_4	C_5	C_6	C_7
C_{test}	C_3	C_4	C_5	C_6	C_7	C_8

0.001. The model-free policy is trained using Soft Actor-Critic (SAC) [7]. For SAC, fully connected neural networks are used for actor and critic with two hidden layers of 256 neurons each. The value of τ (soft updates) is set to 0.005, and networks are optimized using MSE loss and the ADAM optimizer. For the PD controller, the value of K_p is set to -0.5 and K_d is set to 5.

B. Experiments

*a) **Experiment 1** Simulation:* In this experiment, we simulate data using f to answer Q1: to show that f can simulate data for new parameter settings, we train and simulate on different setups (defined in Section III). Note that different setups represent different coins, with different mass m_c and frictional coefficients μ_b and μ_t.

A simulation experiment set has coin setups given by $\{C_{\text{train}}, C_{\text{val}}, C_{\text{test}}\}$. FEM data from $\{C_{\text{train}},\ C_{\text{val}}\}$ is used to optimize weights θ (material network). Data is simulated recursively (for T = 1 s) in a test setup C_{test}, using simulator f. This f consists of (i) a material network with previously optimized θ, and (ii) the dynamics with parameters ϕ_{test} (based on C_{test}). Results are averaged for 6 simulation experiment sets presented in Table II.

We evaluate the absolute errors encountered in simulating $\hat{s}_{t+1}$ (x_t and u_t) at each timestep t. For example, error in x_t is given by $e^x_t = |\hat{x}_t - x_t|$, where $\hat{x}_t$ is the location simulated using f, and x_t is the true value (from the FEM dataset). Absolute errors in data simulated using a black-box baseline Neural Network (NN) trained using data from C_{train} and C_{val} are also included (i.e., a NN that simply approximates $\hat{s}_{t+1} = NN(s_t,\ a_t; w)$).

*b) **Experiment 2** Control and inference:* In this experiment, we evaluate f to answer Q2 (performance of model-based MPC compared to other control policies?) and Q3 (what is the accuracy of the inferred parameters?). In the control step (Algorithm 2), we learn closed-loop control and infer m_c and μ_b for test setups C_1 and C_2. Prior to this, in the learning step (Algorithm 1), we train the model f_a using the FEM data from setups $\{C_3, \ldots, C_8\}$. We do not use the data from C_1 and C_2 during learning to avoid information leakage. For C_1 and C_2, we learn the following policies:

(i) *MPC policy*: A model-based control policy defined using differentiable model f and an MPC solver [6]
(ii) *SAC policy*: A model-free Actor-Critic policy learnt using the Soft Actor-Critic algorithm [7],
(iii) *PD policy*: A feedback based control policy (previously tested for DEA control [8]),
(iv) *Heur policy*: A heuristic control policy that linearly ramps up actuation voltage (i.e., voltage increases by 0.5 V after each iteration until terminal state).

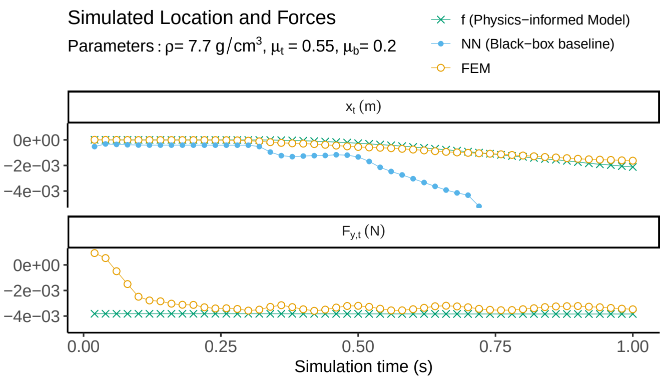

Fig. 5. Data for setup C_3 ($\rho = 7.7g/cm^3, \mu_b = 0.2, \mu_t = 0.55$). Data is simulated given the action sequences in the test setup.

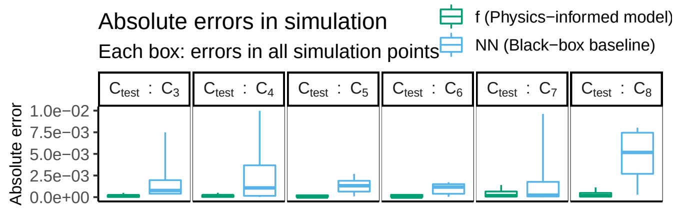

Fig. 6. Absolute error in simulated location x_t. Each box point has data for all simulation points.

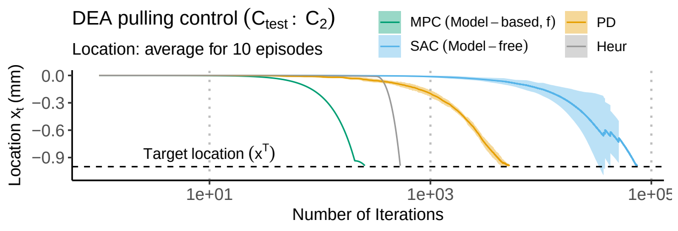

Fig. 7. DEA pulling control for C_2 . Terminal state: coin is $\leq$ 0.01 mm from x^T (Solid line: average for 10 'episodes', shaded region: 25-75%)

The *Heur policy* is a simple policy inspired by typical soft-robotic control policies [3]. The environment is simulated by trained physics-informed models $f_{s,1}$ and $f_{s,2}$. This is due to the lack of a real-world DEA setup, however is not limiting, as we show that our physics-informed models are accurate simulators (Section VI-A).

VI. Results and Discussions

We now present results for Experiment 1, and Experiment 2 to answer Q1-Q3. We show that the physics-informed model learnt in our framework is an accurate simulator and can assist in fast closed-loop model-based control.

A. Simulation Results (Q1)

To evaluate our physics-informed model as a simulator for new setups (defined by parameters of system dynamics, Q1), we study the absolute errors in simulating the data across test setups. Figure 5 shows the location (x_t) and forces (F_y) simulated using physics-informed model f, a baseline Neural Network (NN), and the FEM model. In the first half of the simulation time during static friction (Section III-B), we see negligible displacements and increasing F_y. In the latter half, the kinetic frictional force becomes stable, and we see a change in coin location (x_t). Our model f accurately estimates x in both stages. During kinetic friction, f outperforms the baseline NN, where the location simulated by NN increases exponentially.

Figure 6 shows the absolute error in x for all test setups for f and baseline NN. In all cases, f outperforms the baseline NN. Additionally, the average absolute error in x simulated using f is less than 0.05 times the magnitude of the actual values, i.e., we note approximately $\leq 5\%$ error compared the FEM simulation. We see a similar accuracy for f compared to the FEM and NN in simulating velocity u_x for the coin. We notice similar results in simulated velocity u_t.

The absolute error in forces F_x and F_y in the region of static friction is higher compared to the region of kinetic friction. This happens because we optimize material network m using a physics informed loss function, i.e., a loss function that is based on the error in the next state s and the error in the interaction variables z (Eq. (5)). Optimizing m using this loss function assists in learning the overall manipulation task behaviour, as opposed to only learning the outputs of m (F_x and F_y). This behaviour is non-restrictive, as the objective of our model is to learn the next state of the motion, which is simulated accurately.

Once trained, model f can simulate data for new setups by changing dynamics parameters (like mass m_c). This helps in answering Q1 (simulation accuracy of f for system with different parameter setting?), as we can conclude from Fig. 6 that our physics-informed model has $\leq 5\%$ error compared to FEM and outperforms a baseline NN.

B. Control and Inference Results (Q2-Q3)

This section presents results for Experiment 2, which aims to evaluate closed-loop control (Q2) and inference (Q3) defined in the proposed framework. In DEA pulling, using Algorithm 2, we learn a model-based control policy by utilizing the differentiable f and an MPC solver. Figure 7 shows the average coin location x_t during closed-loop control of test setup C_2. Average is calculated across 10 episodes to compare the *MPC policy* (model-based), with *SAC policy* (model-free), a *PD policy* and a *Heur policy* (defined in Experiment 2). We notice similar results for both test setups C_1 and C_2.

The coin reaches the target in ≤ 200 iterations (0.2 s) under the *MPC policy*. In contrast, the *SAC policy* takes approximately 10,000 iterations (100 s) to reach the target. We further note that the *MPC policy* outperforms the *PD policy* and *Heur policy*, which take approximately 1,500 and 500 iterations respectively.

During control using *Heur policy* we notice sudden motion towards x^T after approximately 280 iterations. This represents the transition from stage of static friction to stage of kinetic friction. The *Heur policy* linearly ramps up actuation voltage every iteration, and thus, does not depend on location

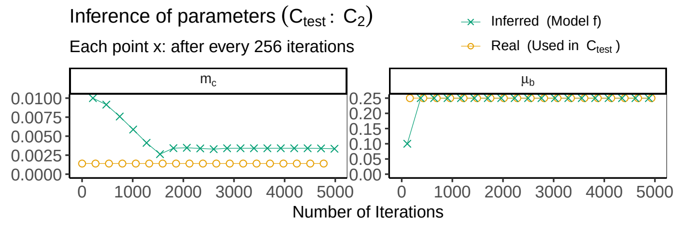

Fig. 8. Inference of mass of the coin and frictional coefficient by DEA pulling. (Reported after every 256 iterations to keep figure legible)

feedback. In contrast, both *PD policy* and *SAC policy* rely on observed location, and take longer to reach and manipulate the coin in the stage of kinetic friction.

Figure 8 shows the inferred m_c and μ_b by physics-informed model f_a and the real value, i.e., the value used in C_{test} during control with *MPC policy*. We note inferred m_c converges to $\leq 10\%$ error compared to the real value within 2,000 iterations. Similarly, μ_b, the frictional coefficient converges within $\leq$ 300 iterations. We notice similar results in both test setups and across all episodes.

We answered Q2 and Q3 in this section, and conclude that our framework learns accurate physics-informed model f that can be used as a simulator for inference and developing closed-loop model-based control policies. In control, a model-based *MPC policy* outperformed all other policies with an order of hundreds of iterations, and we note $\leq 10\%$ in parameter inference.

VII. Conclusions

This paper presents a framework to learn a differentiable simulator and develop control for soft robotic manipulation. We defined a physics-informed model f consisting of a material network m, and dynamics d. This model f can be used as a simulator for data-generation, inference, and control policy optimization. We designed a soft-robotics case study where a coin is pulled using unimorph DEA. FEM simulation of the DEA generated data to train f. Our experiments used multiple setups to evaluate the framework in learning f and model-based control.

From our analyses, we conclude that, (i) the physics-informed model f trained using the proposed framework can simulate new setups (characterized by parameters ϕ) with $\leq 5\%$ error compared to FEM (Fig. 6); (ii) a closed-loop *MPC policy* based on differentiable model f outperformed all other policies in orders of hundreds of iterations (Section VI-B); (iii) f can be used for accurate inference of the parameters ϕ: m_c and μ_b (Fig. 8). Open questions for future research include evaluating this framework for effective control of other soft robotic scenarios, and exploring alternative model-based policies which overcome the high computational requirements of MPC.

Acknowledgment

The authors would like to thank Mr. Mukul Sahu for his valuable comments on FEM modeling.

References

[1] S. Kim, C. Laschi, and B. Trimmer, “Soft robotics: a bioinspired evolution in robotics,” *Trends in Biotechnology*, vol. 31, 2013.

[2] D. Rus and M. Tolley, “Design, fabrication and control of soft robots,” *Nature*, vol. 521, 2015.

[3] H. Yin, A. Varava, and D. Kragic, “Modeling, learning, perception, and control methods for deformable object manipulation,” *Science Robotics*, vol. 6, 2021.

[4] N. El-Atab, R. Mishra, F. Al-modaf, L. Joharji, A. Alsharif, H. Alamoudi, M. Diaz, N. Qaiser, and M. Mustafa, “Soft actuators for soft robotic applications: A review,” *Advanced Intelligent Systems*, vol. 2, 2020.

[5] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026–5033.

[6] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable mpc for end-to-end planning and control,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*, 2018, p. 8299–8310.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018, pp. 1861–1870.

[8] T. Karner and J. Gotlih, “Position control of the dielectric elastomer actuator based on fractional derivatives in modelling and control,” *Actuators*, vol. 10, 2021.

[9] T. Lopez-Guevara, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, and K. Subr, “Adaptable pouring: Teaching robots not to spill using fast but approximate fluid simulation,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 77–86.

[10] E. Coevoet, A. Escande, and C. Duriez, “Soft robots locomotion and manipulation control using fem simulation and quadratic programming,” in *Proceedings of 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, 2019, pp. 739–745.

[11] S. David Müzel, E. Bonhin, N. Guimarães, and E. Guidi, “Application of the finite element method in the analysis of composite materials: A review,” *Polymers*, vol. 12, 2020.

[12] U. Gupta, L. Qin, Y. Wang, H. Godaba, and J. Zhu, “Soft robots based on dielectric elastomer actuators: a review,” *Smart Materials and Structures*, vol. 28, 2019.

[13] F. Zhou, X. Yang, Y. Xiao, Z. Zhu, T. Li, and Z. Xu, “Electromechanical analysis and simplified modeling of dielectric elastomer multilayer bending actuator,” *AIP Advances*, vol. 10, 2020.

[14] J. Shintake, H. Shea, and D. Floreano, “Biomimetic underwater robots based on dielectric elastomer actuators,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4957–4962.

[15] M. Duduta, F. Berlinger, R. Nagpal, D. R. Clarke, R. J. Wood, and F. Z. Temel, “Tunable multi-modal locomotion in soft dielectric elastomer robots,” *IEEE Robotics and Automation Letters*, vol. 5, 2020.

[16] O. Araromi and S. Burgess, “A finite element approach for modelling multilayer unimorph dielectric elastomer actuators with inhomogeneous layer geometry,” *Smart Materials and Structures*, vol. 21, 2012.

[17] X. Zhao and Z. Suo, “Method to analyze programmable deformation of dielectric elastomer layers,” *Applied Physics Letters*, vol. 93, 2008.

[18] P. Jamdagni and Y.-B. Jia, “Robotic cutting of solids based on fracture mechanics and fem,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 8252–8257.

[19] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos, “DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting,” in *Proceedings of Robotics: Science and Systems*, 2021.

[20] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *Journal of Computational Physics*, vol. 378, 2019.

[21] R. E. Pelrine, R. D. Kornbluh, and J. P. Joseph, “Electrostriction of polymer dielectrics with compliant electrodes as a means of actuation,” *Sensors and Actuators A: Physical*, vol. 64, 1998.

[22] M. Smith, *ABAQUS/Standard User’s Manual, Version 6.9*. United States: Dassault Systèmes Simulia Corp, 2009.

[23] V. Piefort, “Finite element modeling of piezoelectric structures,” 2000.

[24] N. Wang, C. Chaoyu, H. Guo, B. Chen, and X. Zhang, “Advances in dielectric elastomer actuation technology,” *Science China Technological Sciences*, vol. 61, 2017.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” 2017.