2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Physics-Informed Neural Networks for Continuum Robots: Towards Fast Approximation of Static Cosserat Rod Theory

Martin Bensch, Tim-David Job, Tim-Lukas Habich, Thomas Seel and Moritz Schappler

***Abstract*— Sophisticated models can accurately describe deformations of continuum robots while being computationally demanding, which limits their application. Especially when considering sampling-based path planning, the model has to be evaluated frequently, which can lead to substantially increased computation times. We present a new approach to compute the entire shape of a tendon-driven continuum robot by a physics-informed neural network (PINN). The underlying physics is modelled with the Cosserat rod theory and incorporated into the PINN's loss function. The boundary values for the training are obtained from a reference model, solved by the shooting method. Our approach allows for a computation of the learned Cosserat rod model multiple orders of magnitude faster than a publicly available reference model. The median position deviation from the reference model lies below $1\,\mathrm{mm}$ (0.5% of the simulated robot length) for each of the robot's 20 disks.**

I. Introduction

Due to their slender and compliant design, continuum robots (CRs) are well-suited for inspection tasks in confined spaces [1]. Therefore, path planning is needed, and since the robot has to perform relatively slow motion, quasi-static models are sufficient to describe the robot's shape. The most general approach for modelling slender tendon-driven continuum robots (TDCRs) is the Cosserat rod theory [2]. Rucker and Webster proposed such a model for statics and dynamics, assuming friction-less tendon-routing [3]. Unfortunately, these models are computationally demanding and, to the best of our knowledge, *path-planning algorithms* in continuum robotics therefore do not make use of them. Instead, only simple kinematic models are used [4], [5], [6], since they are fast to compute. Especially in a *sampling-based approach* (e.g., RRT [4]), the model has to be computed frequently and might become a bottleneck. The authors of [6] based their path-planning on the robot's Jacobian, which can also be computationally demanding if it has to be obtained numerically due to a non-existing analytical formulation. One disadvantage of using purely kinematic models for path planning in confined spaces is that the planner may output a path that is not feasible for the robot in a real scenario. For instance, due to the robot's self-weight, it might differ in its shape significantly from the constant-curvature (CC) assumption [7], leading to collisions, or the necessary actuation might exceed the allowed maximum forces for following the CC path. Hence, *computationally efficient models* that can account for the non-linear and non-CC shapes are needed.

All authors are with the Leibniz University Hannover, Institute of Mechatronic Systems, 30823 Garbsen, Germany. Contact: martin.bensch@imes.uni-hannover.de.
The work received funding from the programme zukunft.niedersachsen by the Lower Saxonian Ministry of Science and Culture.

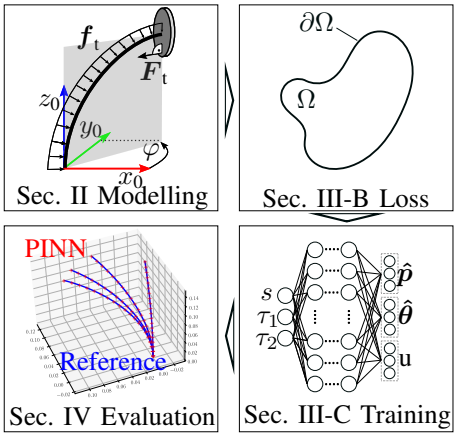

Fig. 1. Scheme of the proposed PINN approach and evaluation

Although *data-driven models* can make accurate predictions and are fast to compute, they lack the ability to represent the full shape so far, which is needed for collision detection. *Physics-Informed Neural Networks* [8], on the other hand, offer the ability of incorporating physics into the training, which allows for predicting the *entire robot's shape*, given the actuation, instead of just learning actuator to end-effector mappings. Data-driven methods can account for effects that cannot be modelled properly by classical methods given suitable measurement data of the considered system [9]. PINNs can learn from two sources: prior knowledge of the underlying physics and measurement data (e.g., unmodeled effects like hysteresis).

A. Related Work

Till et al. [10] proposed a fast static and dynamic Cosserat rod formulation, which allows real-time control, but does not offer the possibility to incorporate data to account for unmodeled effects. Another approach was proposed by [11] using a non-linear observer as a solver which shows a 7-fold speed-up compared to common solvers utilized to solve the Cosserat rod dynamics. However, models with the lowest possible computation time are necessary for sampling-based path planning. Otherwise, the model might become the bottleneck during the planning since it has to be evaluated frequently. Approximations to the Cosserat rod statics by employing classical model-order reduction methods where proposed by [12] using the Galerkin-Ritz method and [13] using the collocation method. Although they can reduce computation time, they must trade accuracy for computation speed.

In addition to classical methods, multiple approaches were proposed for *learning CR's forward and inverse kinematics*. Schemes for learning the inverse kinematic mapping for a three-segment CR were proposed by [14] and [9]. They sampled from a Cosserat rod model that did not account for the influence of tendon forces on the backbone. The learned forward and inverse kinematics of a concentric tube CR from measurement data in [15] already achieved sufficient accuracy for position and orientation. In [16] an artificial neural network (ANN) learned the in-plane forward kinematics of a TDCR. Their ANN only predicted discrete positions in one bending plane, which were connected by cubic spline interpolation after the prediction. Two outputs for each predicted position were necessary. Hence, its output size scaled with the amount of predicted points per shape. Recurrent neural networks (RNNs) were used in [17] for in-plane tip force estimation, while in [18] discrete points along the robot shape were predicted for given measurement input from multiple distributed bending sensors. Both works are unsuitable for the path planning problem, because one would need already known points along the shape as inputs. The inverse CC model was learned for a TDCR in [19] and [20], where for the latter, the approximator was combined with an ANN control strategy and state observer to account for external forces despite the utilised CC model. In contrast to the aforementioned works, in [21] the dynamics model of a CR was employed by an RNN-based parallel predictive scheme, where the CR was discretised into a *chain* of RNN models that had to be trained for each trajectory individually with the dynamics model.

None of the previously mentioned authors focused on *learning the continuous shape* of a CR for given actuation forces, nor were the physics of the considered system itself introduced into the learning scheme. Models were used for sampling purposes only. A novel approach was published by [22], wherein Gaussian process regression was used to estimate the full robot's shape given some measurement data of the robot's tip pose. However, they, too, relied on a simple constant-curvature model. A more sophisticated approach was taken by [23] where they modelled the TDCR using Cosserat rod theory and formulated the Gaussian process by exploiting Lie theory. Although these methods are able to predict the robot's shape continuously and incorporate the underlying kinematics/physics, they are not meant to be used in a sampling scheme but for estimation only, given (some) external measurements along the robot's shape.

To target the approximation of hard-to-compute physics, Raissi et al. recently introduced PINNs [24], [8], which incorporate the considered physics into the learning process. PINNs were successfully applied to physical systems defined by ordinary and partial differential equations, and since they have been proposed, much progress has been made [25], [26]. For instance, [27] utilised PINNs in order to learn the dynamics of a rigid manipulator with two degrees of freedom, which was then used for model-predictive control. To the best of our knowledge, the only work employing physics-informed learning in the field of continuum robotics is [28], where the piecewise-constant-curvature (PCC) dynamics of a two-segment TDCR was learned. To do so, they applied Hamiltonian and Lagrangian neural networks, but PCC dynamics model the real physics less accurate than the Cosserat rod theory [2].

B. Contribution

The few existing works that use purely data-driven methods for the modelling of continuum robots only consider actuator-to-tip mappings or discretise the manipulator [16], [15], [14], [9], but *cannot efficiently predict the complete shape of a continuum robot with high accuracy*. Therefore, these approaches cannot be used for path planning since an efficient shape representation is needed to perform *collision detection* when operating in confined spaces. Furthermore, they only used measurement data, kinematics-only models, or more sophisticated static models solely to generate training data. Hence, they did not incorporate prior knowledge of the underlying physics into the learning process.

Our work contributes to this field with the main goal of realising *fast-to-evaluate CR models with high accuracy*: For the first time, we **(1)** apply physics-informed neural networks to learn the Cosserat rod statics of a continuum robot. To show the advantage of our method, **(2)** we compare our PINN against a benchmark reference model provided by [2] and **(3)** achieve an improvement of computation time by a factor of 641 at comparable accuracy. In addition, **(4)** we publish our implementation open source[1] increasing the accessibility of the method and reproducibility of the shown results. Unlike all prior works, the proposed hybrid model predicts the complete shape of the TDCR, making it a promising approach for employing accurate and computationally efficient models for sampling-based path planning. The methodology is not limited to tendon actuation but can be applied to other types of CRs such as pneumatic soft robots.

C. Outline

The remainder of this publication is structured as follows. In Section II the Cosserat rod model and PINNs are introduced. Section III presents the definition of our learning method and the training. In Section IV we describe our evaluation and results. Section V concludes the paper and outlines future work.

II. Fundamentals

The utilised Cosserat rod statics and physics-informed neural networks are presented in the following subsections.

A. Cosserat Rod Statics

We adopt the model and notation from [3] except for the $\hat{\diamond}$ ("hat") operator, where we relate to the cross-product mapping by the bracket notation $\boldsymbol{x} \times \boldsymbol{y} = [\boldsymbol{x}]_{\times}\boldsymbol{y}$. Vectors are denoted by bold symbols and matrices are written in non-bold capital letters. Fig. 2 shows the free-body diagram on which the following formulation is based. Considering

[1] https://github.com/Martin-Bensch/tdcr-pinn.git

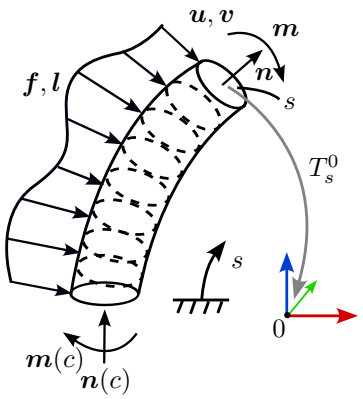

Fig. 2. Free-body diagram of one section of a rod to illustrate the Cosserat rod statics. The vectors $\boldsymbol{m}$ and $\boldsymbol{n}$ represent inner couple and inner force in the base coordinate system, c indicates a constant location and T_s^0 represents the coordinate system at s w.r.t. the base coordinate system.

a rod of length l that is subject to a force distribution $\boldsymbol{f}$ per unit of $s \in [0, l]$, where s is considered to be the reference parameter. Each cross-section is characterised by its position vector $\boldsymbol{p}(s) \in \mathbb{R}^3$ and orientation $R(s) \in \mathrm{SO}(3)$. The linear and angular rates of change are expressed by the kinematic variables $\boldsymbol{v}(s)$ and $\boldsymbol{u}(s)$ defined in the local frame. They indicate the change in orientation and position through the relation $\dot{R}(s) = R(s)[\boldsymbol{u}(s)]_\times$ and $\dot{\boldsymbol{p}}(s) = R(s)\boldsymbol{v}(s)$, where the *dot* symbol indicates a derivative w.r.t. s. From a Lie-group perspective, the kinematic variable $[\boldsymbol{u}(s)]_\times$ is an element of the tangent space $T_{R(s)}\mathrm{SO}(3)$ at $R(s)$ and the adjoint $R(s)$ maps it on the Lie algebra $\mathfrak{so}(3)$ of $\mathrm{SO}(3)$. The isomorphism $[\cdot]_\times$ acts as a mapping from $\mathbb{R}^3$ to the generators of the Lie algebra $\mathfrak{so}(3)$ [29]. Mechanical equilibrium is present for a rod with force and moment distributions $\boldsymbol{f}(s), \boldsymbol{l}(s)$ and internal force and moment vectors $\boldsymbol{n}(s)$, $\boldsymbol{m}(s)$, if the resultant forces and moments acting on each material segment $[c, s]$ are zero, where c indicates a constant location on the rod. This leads to

$$\boldsymbol{n}(s) - \boldsymbol{n}(c) + \int_c^s \boldsymbol{f}(\bar{s})\,\mathrm{d}\bar{s} = \boldsymbol{0} \tag{1}$$

$$\boldsymbol{m}(s) - \boldsymbol{m}(c) + \boldsymbol{p}(s) \times \boldsymbol{n}(s) - \boldsymbol{p}(c) \times \boldsymbol{n}(c) + \int_c^s [\boldsymbol{p}(\bar{s}) \times \boldsymbol{f}(\bar{s}) + \boldsymbol{l}(\bar{s})]\,\mathrm{d}\bar{s} = \boldsymbol{0}, \tag{2}$$

where moments and forces at the ends must be incorporated by initial and/or boundary values. After differentiating the system (1)–(2) w.r.t. s one obtains the differential equations for the special Cosserat rod [30]:

$$\dot{\boldsymbol{n}}(s) + \boldsymbol{f}(s) = \boldsymbol{0}, \tag{3}$$

$$\dot{\boldsymbol{m}}(s) + \dot{\boldsymbol{p}}(s) \times \boldsymbol{n}(s) + \boldsymbol{l}(s) = \boldsymbol{0}. \tag{4}$$

Material properties are introduced by the shear-extension $K_{\mathrm{se}}(s)$ and bending-torsion $K_{\mathrm{bt}}(s)$ stiffness matrices by

$$\boldsymbol{n}(s) = R(s)K_{\mathrm{se}}(s)(\boldsymbol{v}(s) - \boldsymbol{v}^*(s)) \tag{5}$$

$$\boldsymbol{m}(s) = R(s)K_{\mathrm{bt}}(s)(\boldsymbol{u}(s) - \boldsymbol{u}^*(s)), \tag{6}$$

where $\diamond^*$ denotes the undeformed reference configuration. The matrices $K_{\mathrm{se}}(s)$ and $K_{\mathrm{bt}}(s)$ are defined by

$$K_{\mathrm{se}}(s) = \mathrm{diag}\,(GA(s), \quad GA(s), \quad EA(s)),$$
$$K_{\mathrm{bt}}(s) = \mathrm{diag}\,(EI_{xx}(s), \quad EI_{yy}(s), \quad G(I_{xx}(s) + I_{yy}(s))),$$

with $A(s)$ denoting the area of cross-section, E Young's modulus, G shear modulus, and $I_{xx}(s)$ and $I_{yy}(s)$ the second moments of area of the cross-section about the principal axis. We consider a TDCR, with fully constrained tendon routing. Hence, the i-th tendon terminating at a disk at $\boldsymbol{p}_i(s = s_1)$ applies a tendon force $\boldsymbol{F}_i$, with tension τ_i, and associated moment $\boldsymbol{M}_i$ on the CR with

$$\boldsymbol{F}_i(s_1) = -\tau_i \frac{\dot{\boldsymbol{p}}_i(s_1)}{\|\dot{\boldsymbol{p}}_i(s_1)\|_2} \tag{7}$$

$$\boldsymbol{M}_i(s_1) = (\boldsymbol{p}_i(s_1) - \boldsymbol{p}(s_1)) \times \boldsymbol{F}_i(s_1). \tag{8}$$

The resulting tendon force $\boldsymbol{F}_{\mathrm{t}}(s_1)$ and moment $\boldsymbol{M}_{\mathrm{t}}(s_1)$ at a location s_1 are calculated by the summation of the n individual tendon forces/moments. For a better readability, we drop the dependency on s, whenever its apparent. In addition, the tendons lead to distributed forces $\boldsymbol{f}_i$ on the CR's backbone, which can be calculated by

$$\boldsymbol{f}_i = \tau_i \frac{[\dot{\boldsymbol{p}}_i]_\times^2}{\|\dot{\boldsymbol{p}}_i\|_2^3}\ddot{\boldsymbol{p}}_i \quad \text{and} \quad \boldsymbol{f}_{\mathrm{t}} = \sum_{i=1}^{n} \boldsymbol{f}_i, \tag{9}$$

where $\boldsymbol{f}_{\mathrm{t}}$ is the collective distributed tendon load due to tendon tension. It follows from (3) for the assumptions of frictionless contact (between tendon and disks) and the tendon being an inextensible, ideal string (that can only support tension) [3]. After several substitutions and rearrangements, [3] presented the complete model of a TDCR with general tendon routing and external loading, summarized by

$$\dot{\boldsymbol{p}} = R\boldsymbol{v}, \quad \dot{R} = R[\boldsymbol{u}]_\times,$$
$$\begin{pmatrix} \dot{\boldsymbol{v}} \\ \dot{\boldsymbol{u}} \end{pmatrix} = \begin{pmatrix} K_{\mathrm{se}} + C & G \\ B & K_{\mathrm{bt}} + H \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{c} \end{pmatrix}. \tag{10}$$

The intermediate matrix and vector quantities are defined as

$$\boldsymbol{a} = \sum_{i=1}^{n} C_i([\boldsymbol{u}]_\times R^{\mathrm{T}}\dot{\boldsymbol{p}}_i), \qquad C_i = \tau_i \frac{[R^{\mathrm{T}}\dot{\boldsymbol{p}}_i]_\times^2}{\|\dot{\boldsymbol{p}}_i\|_2^3}$$

$$\boldsymbol{b} = \sum_{i=1}^{n} [\boldsymbol{r}_i]_\times C_i([\boldsymbol{u}]_\times R^{\mathrm{T}}\dot{\boldsymbol{p}}_i), \qquad A = \sum_{i=1}^{n} C_i,$$

$$H = -\sum_{i=1}^{n} [\boldsymbol{r}_i]_\times C_i [\boldsymbol{r}_i]_\times, \qquad B = \sum_{i=1}^{n} [\boldsymbol{r}_i]_\times C_i,$$

$$\boldsymbol{c} = K_{\mathrm{bt}}\dot{\boldsymbol{u}}^* - [\boldsymbol{u}]_\times K_{\mathrm{bt}}(\boldsymbol{u} - \boldsymbol{u}^*) - [\boldsymbol{v}]_\times K_{\mathrm{se}}(\boldsymbol{v} - \boldsymbol{v}^*) - R^{\mathrm{T}}\boldsymbol{l}_{\mathrm{e}} - \boldsymbol{b}, \qquad G = -\sum_{i=1}^{n} C_i [\boldsymbol{r}_i]_\times$$

$$\boldsymbol{d} = K_{\mathrm{se}}\dot{\boldsymbol{v}}^* - [\boldsymbol{u}]_\times K_{\mathrm{se}}(\boldsymbol{v} - \boldsymbol{v}^*) - R^{\mathrm{T}}\boldsymbol{f}_{\mathrm{e}} - \boldsymbol{a}$$

with $\boldsymbol{f}_{\mathrm{e}}$ and $\boldsymbol{l}_{\mathrm{e}}$ being external distributed loads, such that $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{t}} + \boldsymbol{f}_{\mathrm{e}}$ and $\boldsymbol{l} = \boldsymbol{l}_{\mathrm{t}} + \boldsymbol{l}_{\mathrm{e}}$, while $\boldsymbol{r}_i$ specifies the tendon routing on a disk in its body frame, and we already accounted for parallel tendon routing along the rod.

B. Physics-Informed Neural Networks

PINNs were proposed by [8] in order to solve non-linear partial differential equations. However, they can also be used for solving (non-linear) ordinary differential equations. They include physics into the training, which guides the training by

penalising inconsistencies with the governing equations [31]. PINNs can be used to learn physics expressed like

$$\begin{aligned} \mathcal{N}(\boldsymbol{y}(\boldsymbol{x});\boldsymbol{\gamma}) &= \boldsymbol{f}(\boldsymbol{x}), \quad \text{with} \quad \boldsymbol{x} \in \Omega, \\ \mathcal{B}(\boldsymbol{y}(\boldsymbol{x})) &= \boldsymbol{g}(\boldsymbol{x}), \quad \text{with} \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{11}$$

where $\mathcal{N}$ is a non-linear differential operator, $\boldsymbol{y}$ the unknown solution, $\boldsymbol{\gamma}$ physics parameters and $\boldsymbol{f}$ a function identifying the data of the problem. The underlying initial or boundary conditions of the partial/ordinary differential equation are specified by $\mathcal{B}$ while $\boldsymbol{g}$ is the boundary function [25]. The goal is to learn an approximation $\hat{\boldsymbol{y}}$ to the solution $\boldsymbol{y}$ of the system (11) (the $\hat{\diamond}$ indicates the approximation by the PINN). In addition, labelled data can also be incorporated into the total loss function if available, resulting in the optimisation problem, w.r.t. the weights $\boldsymbol{\omega}$ of the ANN,

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}}{\operatorname{argmin}}(\lambda_{\mathcal{N}}\mathcal{L}_{\mathcal{N}} + \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}}). \tag{12}$$

The first two terms in (12) penalise violations of the governing equation and their initial/boundary conditions, while the last term accounts for additional data (e.g., measurement data from a test bench), which leads to the loss function

$$\begin{aligned} \mathcal{L}_{\text{total}} &= \lambda_{\mathcal{N}}\mathcal{L}_{\mathcal{N}} + \lambda_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + \lambda_{\text{data}}\mathcal{L}_{\text{data}} \\ &= \lambda_{\mathcal{N}}\frac{1}{N_{\mathcal{N}}}\sum_{i=1}^{N_{\mathcal{N}}}\|\mathcal{N}(\hat{\boldsymbol{y}}(\boldsymbol{x}_i);\boldsymbol{\gamma}) - \boldsymbol{f}(\boldsymbol{x}_i)\|^2 \\ &+ \lambda_{\mathcal{B}}\frac{1}{N_{\mathcal{B}}}\sum_{i=1}^{N_{\mathcal{B}}}\|\mathcal{B}(\hat{\boldsymbol{y}}(\boldsymbol{x}_i)) - \boldsymbol{g}(\boldsymbol{x}_i)\|^2 \\ &+ \lambda_{\text{data}}\frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}\|\hat{\boldsymbol{y}}(\boldsymbol{x}_i) - \boldsymbol{y}_i\|^2. \end{aligned} \tag{13}$$

The derivatives of the ANN which are needed to form the physics loss $\mathcal{L}_{\mathcal{N}}$ and – depending on the problem at hand – $\mathcal{L}_{\mathcal{B}}$, are obtained by automatic differentiation (AD) by traversing through its computational graph. Since all inverse operations in the computational graph are analytic, derivatives are computed by the chain rule of differentiation, instead of difference quotients.

III. Methods

Systems defined by (11) normally describe dynamic systems, which among other factors, depend on time (e.g., the time evolution of a pendulum, flow in fluid mechanics). In the present work, however, we consider the *statics* of a TDCR, which are defined by (10). Since this presents a system of ordinary differential equations depending on the reference variable s, we treat s equivalently to time in a *dynamic* system (similar to the approach proposed by [23]). Fig. 1 shows the proposed scheme. Starting with a physics model, we formulate the loss function and train the PINN.

A. Model and Network Architecture

In the present work, we employ the open-source TDCR model from [2]. The model is implemented in MATLAB and C++. We prepared binding code using [32] to incorporate the provided C++ implementation of the TDCR model into our ANN based on the Python interface of the PyTorch library. Although the model provided by [2] assumes $\boldsymbol{u}$ and $\boldsymbol{v}$ both to be dependent on s, we assume $\boldsymbol{v}$ (associated to stretch and shear) to be constant w.r.t. s, which is a valid assumption for TDCRs with steel backbones. Furthermore, for the sake of simplicity, we don't consider external forces acting on the robot's tip to maintain the simplicity of the example. A fully-connected ANN with three hidden linear layers and a width of 100 neurons was employed. We used the SiLU activation function for the input layer and otherwise hyperbolic tangent. The inputs to the ANN are the reference parameter s and two tendon tensions $\boldsymbol{\tau} = (\tau_1, \tau_2)^{\mathrm{T}}$. Because of restrictions we set regarding the ANN's predictions of the bending plane angle φ on the interval $[0°, 60°]$, we were able to reduce training time. One can always transform an actuation for a bending angle $\varphi \notin [0°, 60°]$ into this interval, predict the shape and afterwards rotate it accordingly. We did not consider pretension due to the numerical instabilities of the reference model. Therefore, we chose to limit the input to two tendon tensions. All inputs to the ANN were normalised w.r.t. their maximum values.

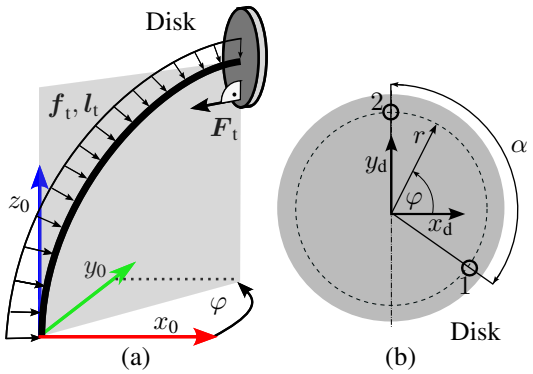

Fig. 3. (a) Shape of the robot with one disk at its tip and distributed tendon load $\boldsymbol{f}_t$, due to collective tendon force $\boldsymbol{F}_t$. (b) Disk at the tip of the robot viewed in negative z_0 direction, when the robot is not bent. Tendons are attached to the holes indicated by 1 and 2. The bending plane is rotated around base axis z_0 by φ.

B. Boundary and Loss Formulation

Modeling the CR represents a boundary value problem (BVP), where in general $\boldsymbol{p}(0)$, $R(0)$, $\boldsymbol{n}(l)$ and $\boldsymbol{m}(l)$ are known and $\boldsymbol{u}(0)$ and $\boldsymbol{v}(0)$ are optimised for. In our scenario, we are interested in learning position $\boldsymbol{p}(s, \boldsymbol{\tau})$, isomorphism of Lie algebra $\boldsymbol{\theta}(s, \boldsymbol{\tau})$, and angular rate of change $\boldsymbol{u}(s, \boldsymbol{\tau})$. In contrast to the standard CR model, our boundary becomes a set (instead of points $\boldsymbol{p}(0)$, $R(0)$, $\boldsymbol{n}(l)$ and $\boldsymbol{m}(l)$). The PINN should learn not only one solution (robot's shape) to a CR BVP but a range of solutions for tendon tensions satisfying $\|\boldsymbol{M}_t(l, \boldsymbol{\tau})\|_2 \leq r \cdot \tau_{\max}$. The term $r \cdot \tau_{\max}$ denotes an upper boundary for the magnitude of the applied tendon couple, where $\tau_{\max}$ is the maximum tendon tension in case only one tendon is actuated, and r is the radius on which the tendon attachment points are located (Fig. 3 b). In addition, the bending plane angle range $\varphi \in [0°, 60°]$ leads to an increase of solution domain as well, which is a direct consequence

of the two-dimensional actuation vector $\boldsymbol{\tau}$. Therefore, we defined the set of (possible) inputs as

$$\mathcal{D} = \big\{(s,\boldsymbol{\tau}) \in \mathbb{R}\times\mathbb{R}^2 | \; s\in[0,l], \boldsymbol{\tau}\in\mathbb{R}^2 | \; \|\boldsymbol{M}_{\mathrm{t}}(l,\boldsymbol{\tau})\|_2 \leq r\cdot\tau_{\max}\big\} \quad (14)$$

to define the initial value (IV) and BV sets. However, for the training, we only used a discrete subset $\mathcal{D}_{\mathrm{sub}} \subset \mathcal{D}$. The index "Ref" in the following BV definition indicates that the elements are obtained from the reference model $\mathcal{M}$, which gives the solutions to (10). The IV/BV definitions are:

$$\partial\mathcal{I}_{0,\mathrm{Ref}} = \Big\{(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) \in \mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^3 | (\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) = \underline{\mathcal{M}(0,\boldsymbol{\tau})} | \; \underline{(0,\boldsymbol{\tau})} \in D_{\mathrm{sub}} \mid \|\boldsymbol{M}_{\mathrm{t}}(l,\boldsymbol{\tau})\|_2 < r\cdot\tau_{\max}\Big\} \quad (15)$$

$$\partial\mathcal{B}_{l,\mathrm{Ref}} = \Big\{(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) \in \mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^3 | (\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) = \underline{\mathcal{M}(l,\boldsymbol{\tau})} | \; \underline{(l,\boldsymbol{\tau})} \in D_{\mathrm{sub}} \mid \|\boldsymbol{M}_{\mathrm{t}}(l,\boldsymbol{\tau})\|_2 < r\cdot\tau_{\max}\Big\} \quad (16)$$

$$\partial\mathcal{B}_{\mathrm{Interior,Ref}} = \big\{(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) \in \mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^3 | \; (\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) = \underline{\mathcal{M}(s,\boldsymbol{\tau})} \mid (s,\boldsymbol{\tau}) \in D_{\mathrm{sub}} | \; s\in(0,l),\; \|\boldsymbol{M}_{\mathrm{t}}(l,\boldsymbol{\tau})\|_2 < r\cdot\tau_{\max}\big\} \quad (17)$$

$$\partial\mathcal{B}_{\tau,\mathrm{Ref}} = \big\{(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) \in \mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^3 | \; (\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u}) = \mathcal{M}(s,\boldsymbol{\tau}) \mid (s,\boldsymbol{\tau}) \in D_{\mathrm{sub}} | \; \underline{\|\boldsymbol{M}_{\mathrm{t}}(l,\boldsymbol{\tau})\|_2 = r\cdot\tau_{\max}}\Big\} \quad (18)$$

$\partial\mathcal{I}_{0,\mathrm{Ref}}$ and $\partial\mathcal{B}_{l,\mathrm{Ref}}$ describe reference values for varying $\boldsymbol{\tau}$ at $s=0$ and $s=l$, respectively. The computation of these sets lead also to data associated to the interval of $s\in(0,l)$, meaning at the disks between base and tip, which we also included into the training denoted by $\partial\mathcal{B}_{\mathrm{Interior,Ref}}$. The last set of BVs generated by the reference model is the set of maximal actuation $\partial\mathcal{B}_{\tau,\mathrm{Ref}}$. For each maximal actuation, we obtained 21 elements $(\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{u})$, corresponding to 20 disks plus the robot's base. In addition to the sets (15)–(18), which are made up of sampled data from the reference model, we can also formulate IVs, where $\mathbb{1}_3$ is the identity matrix,

$$\begin{aligned} \boldsymbol{p}(s=0,\boldsymbol{\tau}) &= \boldsymbol{0} \\ R(s=0,\boldsymbol{\tau}) &= \mathbb{1}_3 \qquad \text{with } (s,\boldsymbol{\tau}) \in \mathcal{C} \end{aligned} \quad (19)$$

for all collocation points $\mathcal{C}$, which are also sampled from $\mathcal{D}$.

Based on the specified sets of IV and BV and (19) we define several loss functions. For $\boldsymbol{p}$ and $\boldsymbol{u}$ we use the mean squared error (MSE) between prediction and reference per dimension and afterwards the mean of the obtained MSE vector. A different metric is needed for $\boldsymbol{\theta}$ and the mean is subsequently evaluated for all samples. Since $\boldsymbol{\theta}$ represents an orientation, and the reference data was given as rotation matrices we use

$$d(\hat{R}, R_{\mathrm{Ref}}) = \|\log(\hat{R}R_{\mathrm{Ref}}^{\mathrm{T}})\| \quad (20)$$

where $d(\cdot,\cdot)$ denotes distance function and $\log(\cdot)$ is the matrix logarithm [33]. Hence, (20) is the magnitude of the angle axis rotation that is needed to align the PINNs prediction $\hat{R}$ (after employing the Rodrigues formula to $\hat{\boldsymbol{\theta}}$) with the reference orientation R_{Ref}. The loss functions regarding the IVs and BVs (15)–(18), obtained by the reference model, do not deviate from standard learning approaches using labelled data. What distinguishes PINNs from the standard approach, however, is the inclusion of the physics equations into the loss function. The system at hand is given by (10), so we defined the *physics loss* to be made of the differences

$$\Delta\dot{\boldsymbol{p}} = \dot{\hat{p}} - \hat{R}\boldsymbol{v}, \qquad \Delta\dot{R} = \dot{\hat{R}} - \hat{R}[\hat{\boldsymbol{u}}]_{\times}, \quad \text{and}$$
$$\begin{pmatrix}\Delta\dot{\boldsymbol{v}}\\ \Delta\dot{\boldsymbol{u}}\end{pmatrix} = \begin{pmatrix}\boldsymbol{0}\\ \dot{\hat{u}}\end{pmatrix} - \begin{pmatrix}K_{\mathrm{se}}+\hat{A} & \hat{G}\\ \hat{B} & K_{\mathrm{bt}}+\hat{H}\end{pmatrix}^{-1}\begin{pmatrix}\hat{\boldsymbol{d}}\\ \hat{\boldsymbol{c}}\end{pmatrix}. \quad (21)$$

With a slight abuse of notation, we indicate the matrix elements as dependencies on the PINN's outputs. The derivatives of $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{p}}$ of the PINN's output were directly computed by PyTorch's AD engine w.r.t. the input s. $\dot{\hat{R}}$ on the other hand was obtained by first applying the Rodrigues formula on $\hat{\boldsymbol{\theta}}$ and then taking the derivative using AD. The $\boldsymbol{0}$ in the difference of $\Delta\dot{\boldsymbol{v}}$ originates from our assumption of $\boldsymbol{v}(s) = (0 \quad 0 \quad 1)^{\mathrm{T}}$ being constant. Therefore, the physics loss was made of the MSE per dimension of (21) and taking the mean per entity. We do not employ the metric for rotations for $\Delta\dot{R}$, since it is apparent that $\dot{R}$ is not an element of $\mathrm{SO}(3)$ but of $\mathfrak{so}(3)$. Therefore, first we square the entries of $\Delta\dot{R}$, then take the mean while keeping its dimension, and afterwards take the mean of the sum of all entries. The remaining loss functions are formed by

$$\Delta\boldsymbol{m}(l) = \hat{R}(l)K_{\mathrm{bt}}\hat{\boldsymbol{u}}(l) + \hat{R}(l)\sum_{i=1}^{2}\boldsymbol{r}_i\times\tau_i\boldsymbol{v}$$
$$\Delta\boldsymbol{u}(0) = \hat{\boldsymbol{u}}(0) + K_{\mathrm{bt}}^{-1}\hat{R}(l)\sum_{i=1}^{2}(\boldsymbol{r}_i\times\tau_i\boldsymbol{v})$$

Finally, we compute the arc length for each collocation point (each actuation), which had to be equal to the robot's length l. All aforementioned loss functions were heuristically weighted and summed and formed the term $\mathcal{L}_{\mathcal{N}}$ in (13).

C. Training

The sampling space is parameterised by the bending plane angle φ (Fig 3, a) and an equivalent force $\boldsymbol{F}_r$ in a distance of r (Fig. 3, b) to the the disk's origin and parallel to the z_{d}-axis. Fig. 3 (b) shows a disk with two attachment points on a circle with radius r and angle α apart, where tendon tensions τ_1 and τ_2 are being applied. The position vector $\boldsymbol{r}'(\varphi)$, with $\|\boldsymbol{r}'(\varphi)\|_2 = r$ indicates the location where an equivalent force $\boldsymbol{F}_r$ has to act for the robot to bend in a bending plane rotated by φ. The tendon tensions τ_1 and τ_2 can be calculated by considering the equilibrium

$$\boldsymbol{r}'(\varphi)\times\boldsymbol{F}_r = \boldsymbol{r}_1\times\boldsymbol{F}_1 + \boldsymbol{r}_2\times\boldsymbol{F}_2 \quad (22)$$

of the tendon forces $\boldsymbol{F}_1, \boldsymbol{F}_2$ at locations 1 and 2 with position vectors $\boldsymbol{r}_1, \boldsymbol{r}_2$ w.r.t. the disk's frame. The magnitudes of $\boldsymbol{F}_1, \boldsymbol{F}_2$ were obtained by

$$F_2 = \frac{\sin(\varphi)}{\sin(\alpha)}F_r \quad \text{and} \quad F_1 = \left(\cos\varphi - \frac{\sin(\varphi)}{\tan(\alpha)}\right)F_r. \quad (23)$$

The number of samples is summarised in Table I. Reference data and collocation points were uniformly distributed in the ranges $\varphi \in [0°, 60°]$ and $F_r \in [0, \tau_{\max}]$. However, with a probability of 10% collocation points were randomly sampled after each epoch depending on the outcome of that epoch.

Therefore, the ten worst actuations regarding the maximal deviations to the reference model were considered. For these, multiple actuations were sampled according to a two-dimensional Gaussian distribution, centred on these ten worst actuations with a variance of $\sigma_\tau^2 = 0.01 \cdot \tau_{\max}$. For each drawn location ten locations s_j were sampled along the rod with variance $\sigma_s^2 = 0.01 \cdot l$. The exact amount of samples was calculated to lie in the same order of magnitude as the collocation points obtained in the deterministic case (grid). PyTorch's "AdamW" optimiser was utilised for the training, together with a one-cycle learning rate scheduler [34] to reach the learning rate $1.5 \cdot 10^{-4}$, starting at a 100th of it. The final learning rate was reached after 150 epochs, then switched to a "reduce on plateau scheme". It was reduced by 10% with regard to the current learning rate if the mean of the ten last epoch's losses did not decrease over the last 20 epochs. However, after the learning rate was reduced, at least ten epochs were trained with the new learning rate, before another reduction could occur.

TABLE I

AMOUNT OF TRAINING DATA FOR EACH SOURCE.

Source	Number	Source	Number	Source	Number
$\partial\mathcal{I}_{0,\mathrm{Ref}}$	120	$\partial\mathcal{B}_{L,\mathrm{Ref}}$	120	$\mathcal{C}$	65,536
$\partial\mathcal{B}_{\mathrm{Interior},\mathrm{Ref}}$	2,286	$\partial\mathcal{B}_{\tau,\mathrm{Ref}}$	1,344		

IV. EVALUATION

The PINN and reference model were compared concerning their computation-time. Therefore, we ran the reference model in different configurations for (fixed) step sizes of the Runge-Kutta-45 integration and the first-order optimality (FOO) criterion of the Levenberg-Marquardt algorithm, part of the GNU Scientific Library. All parameters of the FOO were set to the same value for one configuration. Fig. 4 shows the results of our tests. Each model (PINN and reference with shooting method) was run three times for 108 actuations each, and the time was measured for calling the PINN's prediction function. For the reference model, the time was measured starting with the call to the bound forward statics function, which couldn't prevent including the overhead of passing from Python to C++. To reduce this overhead, the actuation vector, which consisted of all 108 actuations, was passed with a single call for each run of the 108 actuations. A MacbookPro 2021 with M1-Chip and 32 GB of RAM was employed for all experiments. The actuations used for the evaluation were not present in the PINN's training set. If both entries of a generated tension vector were closer than $0.01\,\mathrm{N}$ to any tension vector in the training set, it was removed from the test set. Table II shows the values per configuration, from left to right w.r.t. blue markers in Fig. 4. The accuracy for each model is calculated w.r.t. the reference model configured with an FOO of 10^{-11} and step size

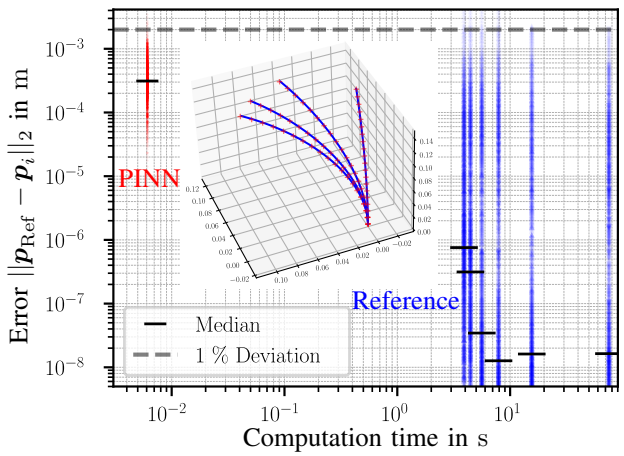

Fig. 4. Accuracy and computation time for the PINN (left, red) and different configurations of the reference model (right, blue). Deviation of 1% w.r.t. the robot's length of 20 cm. Red/blue markers show the accuracy per run for each configuration compared to the baseline configuration (FOO of 10^{-11} and $h = 10^{-4}$). The PINN computes the test configurations 641 times faster at comparable accuracy.

TABLE II

CONFIGURATIONS FOR THE COMPUTATION-TIME (CT) COMPARISON.

Step size	10^{-3}	0.005	0.01	0.01	0.01	0.01
FOO	10^{-4}	10^{-4}	10^{-4}	10^{-3}	0.005	0.01
$\frac{\text{CT reference}}{\text{CT PINN}}$	12,352	2,556	1,297	921	731	641

$h = 10^{-4}$. The position error was measured by the Euclidean distance for all disks ($10\,\mathrm{mm}$ apart from each other) along the TDCR. It is apparent that the PINN's median stays below the 1% accuracy boundary, besides some outliers. The shooting method led to more accurate computations but also showed outliers above the 1% boundary, while its median lies three to five orders of magnitude below the PINN. This raises the question of whether this higher accuracy in simulation also translates to higher accuracy in a real-world scenario, as the CR model itself is limited by the assumptions it is based on. The PINN, however, can potentially represent effects not modelled by the CR by incorporating data from a test-bench. Regarding computation time, the PINN is *two orders of magnitude faster* than the fastest configuration of the CR model. An additional speedup of the PINN would be possible by PyTorch's `compile(.)` function.

V. CONCLUSION

We propose the application of physics-informed neural networks for modelling tendon-driven continuum robots based on the Cosserat rod model with parallel tendon routing. Therefore, the definition of the boundary value problem and the corresponding loss functions are presented in order to learn a *range* of solutions to the Cosserat rod model. Our evaluation shows that a PINN can predict the robot's shape with a 1% error relative to the total arc length compared to the reference model, with a notable 641-fold improvement in computation time. This work demonstrates the feasibility of the utilisation of PINNs in the field of continuum robotics. Future research will focus on increasing the number of segments, considering external forces, gravity, and incorporating measurement data from a test-bench.

REFERENCES

[1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015.

[2] P. Rao, Q. Peyron, S. Lilge, and J. Burgner-Kahrs, "How to model tendon-driven continuum robots and benchmark modelling performance," *Frontiers in Robotics and AI*, vol. 7, Feb. 2021.

[3] D. C. Rucker and R. J. I. Webster, "Statics and dynamics of continuum robots with general tendon routing and external loading," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1033–1044, Dec. 2011.

[4] G. Niu, Y. Zhang, and W. Li, "Path planning of continuum robot based on path fitting," *Journal of Control Science and Engineering*, vol. 2020, pp. 1–11, Dec. 2020.

[5] K. Shahzad, S. Iqbal, and P. Bloodsworth, "Points-based safe path planning of continuum robots," *International Journal of Advanced Robotic Systems*, vol. 12, no. 7, p. 107, July 2015.

[6] I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Path planning for multisection continuum arms," in *2012 IEEE International Conference on Mechatronics and Automation*, Aug. 2012, pp. 1208–1213.

[7] H. P. Sanders and M. D. Killpack, "Dynamically feasible trajectory generation for soft robots," in *2023 IEEE International Conference on Soft Robotics (RoboSoft)*, Apr. 2023, pp. 1–8.

[8] 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, pp. 686–707, Feb. 2019.

[9] T. George Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi, "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments," *Soft Robotics*, vol. 4, no. 3, pp. 285–296, Sept. 2017.

[10] J. Till, V. Aloi, and C. Rucker, "Real-time dynamics of soft and continuum robots based on Cosserat rod models," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 723–746, May 2019.

[11] B. Thamo, K. Dhaliwal, and M. Khadem, "Rapid solution of Cosserat rod equations via a nonlinear partial observer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 9433–9438.

[12] S. M. H. Sadati, S. E. Naghibi, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Control space reduction and real-time accurate modeling of continuum manipulators using Ritz and Ritz-Galerkin methods," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 328–335, Jan. 2018.

[13] A. L. Orekhov and N. Simaan, "Solving Cosserat rod models via collocation and the magnus expansion," no. arXiv:2008.01054, Aug. 2020.

[14] T. G. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi, "Learning global inverse statics solution for a redundant soft robot," in *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, Lisbon, Portugal, July 2016, pp. 303–310.

[15] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the forward and inverse kinematics of a 6-DOF concentric tube continuum robot in SE(3)," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 5125–5132.

[16] X. Li, J. Zhang, J. Zhao, G. Zhang, and C. Shi, "A model-free method-based shape reconstruction for cable-driven continuum manipulator using artificial neural network," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2019, pp. 1424–1429.

[17] F. Feng, W. Hong, and L. Xie, "A learning-based tip contact force estimation method for tendon-driven continuum manipulator," *Scientific Reports*, vol. 11, no. 1, p. 17482, Sept. 2021.

[18] K. Tanaka, Y. Minami, Y. Tokudome, K. Inoue, Y. Kuniyoshi, and K. Nakajima, "Continuum-body-pose estimation from partial sensor information using recurrent neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 244–11 251, Oct. 2022.

[19] A. Ghoul, K. Kara, S. Djeffal, M. Benrabah, and M. L. Hadjili, "Inverse kinematic model of continuum robots using artificial neural network," in *2022 19th International Multi-Conference on Systems, Signals & Devices (SSD)*, May 2022, pp. 1893–1898.

[20] Z. Wang, T. Wang, B. Zhao, Y. He, Y. Hu, B. Li, P. Zhang, and M. Q.-H. Meng, "Hybrid adaptive control strategy for continuum surgical robot under external load," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1407–1414, Apr. 2021.

[21] A. Tariverdi, V. K. Venkiteswaran, M. Richter, O. J. Elle, J. Tørresen, K. Mathiassen, S. Misra, and Ø. G. Martinsen, "A recurrent neural-network-based real-time dynamic model for soft continuum manipulators," *Frontiers in Robotics and AI*, vol. 8, p. 631303, Mar. 2021.

[22] W. Shen, G. Yang, T. Zheng, Y. Wang, K. Yang, and Z. Fang, "An accuracy enhancement method for a cable-driven continuum robot with a flexible backbone," *IEEE Access*, vol. 8, pp. 37 474–37 481, 2020.

[23] S. Lilge, T. D. Barfoot, and J. Burgner-Kahrs, "Continuum robot state estimation using Gaussian process regression on SE(3)," *The International Journal of Robotics Research*, vol. 41, no. 13-14, pp. 1099–1120, Nov. 2022.

[24] M. Raissi, "Deep hidden physics models: Deep learning of nonlinear partial differential equations," no. arXiv:1801.06637, Jan. 2018.

[25] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics-informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, sep 2022.

[26] E. Kharazmi, Z. Zhang, and G. E. M. Karniadakis, "Hp-VPINNs: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, Feb. 2021.

[27] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed neural networks-based model predictive control for multi-link manipulators," *IFAC-PapersOnLine*, vol. 55, no. 20, pp. 331–336, 2022.

[28] J. Liu, P. Borja, and C. Della Santina, "Physics-informed neural networks to model and control robots: A theoretical and experimental investigation," *Advanced Intelligent Systems*, p. 2300385, Feb. 2024.

[29] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," 2018. [Online]. Available: https://arxiv.org/abs/1812.01537

[30] S. S. Antman, *Nonlinear Problems of Elasticity*, 2nd ed., ser. Applied Mathematical Sciences. New York: Springer, 2005, no. v. 107.

[31] C. Legaard, T. Schranz, G. Schweiger, J. Drgoňa, B. Falay, C. Gomes, A. Iosifidis, M. Abkar, and P. Larsen, "Constructing Neural Network Based Models for Simulating Dynamical Systems," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1–34, Nov. 2023.

[32] W. Jakob, J. Rhinelander, and D. Moldovan, "pybind11 – seamless operability between C++11 and Python," 2017, https://github.com/pybind/pybind11.

[33] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[34] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," no. arXiv:1708.07120. arXiv, May 2018.