Motion Dynamics Modeling and Fault Detection of a Soft Trunk Robot

1st Emadodin Jandaghi
Department of Mechanical Engineering
University of Rhode Island
Kingston, USA
emadjandaghi@uri.edu

2nd Xiaotian Chen
Department of Mechanical Engineering
University of Rhode Island
Kingston, USA
xiaotian_chen@uri.edu

6th Chengzhi Yuan
Department of Mechanical Engineering
University of Rhode Island
Kingston, USA
cyuan@uri.edu

***Abstract*—The field of soft robotics has been experiencing rapid growth, with researchers and engineers showing increasing interest due to the unique capabilities of these robots. Soft robots, characterized by their soft bodies and flexible structures, have demonstrated great potential in addressing real-world challenges across various domains, including medical applications. Effective modeling and control are vital for fully harnessing the potential of soft robots, particularly in applications involving human interaction. However, creating models for soft robots made of soft materials, diverse shapes, and actuators poses significant challenges. Moreover, accurate fault detection in soft robots necessitates precise modeling. This paper introduces a novel machine learning approach, termed deterministic learning, for training a soft robot model using a radial basis function neural network. The research explores the fault detection process by simulating four distinct faults that could impair system control performance, such as diminishing tracking accuracy or inducing instability. Furthermore, the paper examines the identification of fault occurrences during the operation of soft robots.**

***Index Terms*—soft robotics, radial basis function neural network, deterministic learning, fault detection.**

I. Introduction

Soft robots are generally constructed using soft materials like silicone and rubber, which provide them with unique characteristics. Their flexible bodies enable bending and twisting with high curvatures, allowing them to adapt to various environments and tasks. Due to their extensive applicability, ranging from underwater settings to the medical field, more research has been conducted on soft robots such as [1], [2], [3], [4], [5] and [6].

Although flexibility offers certain advantages for soft robots, it also complicates the modeling of their dynamics. Soft robots possess infinite-dimensional structures with unknown degrees of freedom, leading researchers to propose continuous mathematical models for describing their dynamics. Some of these models include the Constant-Curvature method [7], Cosserat-Rod-theory-based method[8], and Finite Element Method (FEM) [9]. However, due to linear approximations and simplifications, none of these methods can precisely represent the dynamics of soft robots. Therefore, they are not suitable for fault detection experiments which demand a more accurate approximation model for high nonlinear dynamics of soft robots. Developing a model-based fault detection method [10] that considers the robots' dynamic behavior for improved fault detection accuracy is also challenging due to the inherent physical properties of soft robots.

To prevent performance degradation and ensure safety, it is essential to identify abnormal behavior and faults during robot operation [10]. This necessity has spurred the development of fault diagnosis techniques specifically tailored for soft robots ([11] [12] [13]). However, these techniques may not be applicable when faults remain concealed due to the infinite-dimensional nature and highly nonlinear models of soft robots [14].

The most accurate model of soft robots' nonlinear dynamics can be derived by using neural networks (NN). For instance, [15] which trained a neural network to predict a soft robot's quasi-static physics, [16] which uses artificial NN to learn the input-output model of a soft robot, and [17] used NN for linear movement pattern. These methods have not used the high dimensional system states for simplicity and having less computational effort.

On the other hand, online data-driven model learning has proven to be more efficient in soft robot dynamics modeling when accounting for unknown nonlinearities [18]. This suggests that utilizing data-driven machine learning to capture the robots' dynamic motions can result in more accurate approximations of their models. Studies by [19] and [20] have demonstrated that machine learning can handle complex soft sensor information by modeling and predicting the external environment. [21] and [22] have used the data-driven approach with the continuum kinematics model to improve the modeling accuracy.

In this paper, we apply deterministic learning (DL) ([23] [24]) a novel machine learning method, to identify the motion dynamics of a recently developed soft robot using radial basis function neural networks. Deterministic learning theory can be utilized to find the system dynamics of general nonlinear systems [25] and [26]. A brand new Soft Trunk Robot (STR) which is designed and developed in our previous research [27] is investigated for unknown dynamic estimation using DL with radial basis function NN (RBFNN). After collecting the motion data of the STR following a predefined trajectory, its nonlinear dynamics model is obtained using DL. Then a fault detection procedure is studied involving four different faults that occurred in the STR to detect fault occurrences for each

fault.

The rest of the paper contains the following sections: section II delves into the DL theory in detail. Section III presents the formulation of RBFNN. Section IV introduces the STR design details and actuation. The average weight procedure is discussed in section V. The fault detection procedure and corresponding results are provided in section VI. Finally, section VII concludes the paper.

II. Preliminaries

The RBFNN can be described as equation $f_{nn}(Z) = \sum_{i=1}^{N} w_i s_i(Z) = W^T S(Z)$, where $\mathbf{Z} \in \Omega_{\mathbf{Z}} \subseteq \mathbb{R}^q$ described as input vector, and $W = w_1, ..., w_N^T \in \mathbb{R}^N$ as weight vector. N indicates the number of NN nodes, $S(Z) = [s_1(||\mathbf{Z} - \mu_i||), ..., s_N(||\mathbf{Z} - \mu_i||)]^T$ with $s_i(\cdot)$ is a radial basis function, and $\mu_i (i = 1, ..., N)$ is distinct points in the state space.

The Gaussian function $s_i(||\mathbf{Z} - \mu_i||) = \exp\left[-\frac{(\mathbf{Z}-\mu_i)^T(\mathbf{Z}-\mu_i)}{\eta_i^2}\right]$ is generally used for radial basis functions, where $\mu_i = [\mu_{i1}, \mu_{i2}, ..., \mu_{iN}]^T$ is the center and η_i is the width of the receptive field. The Gaussian function categorized by localized radial basis function s in the sense that $s_i(||\mathbf{Z} - \mu_i||) \to 0$ as $||Z|| \to \infty$.

It has been shown in [28] and [29] that for any continuous function $f(Z) : \Omega_Z \to R$ where $\Omega_Z \subset R^p$ is a compact set, and for the NN approximator, where the node number N is sufficiently large, there exists an ideal constant weight vector W^*, such that for each $\epsilon^* > 0$, $f(Z) = W^{*T} S(Z) + \epsilon(Z)$, $\forall Z \in \Omega_Z$, where $\epsilon(Z)$ is the approximation error. Moreover, for any bounded trajectory $Z_\zeta(t)$ within the compact set Ω_Z, $f(Z)$ can be approximated by using a limited number of neurons located in a local region along the trajectory: $f(Z) = W_\zeta^{*T} S_\zeta(Z) + \epsilon_\zeta$, where $S_\zeta(Z) = [S_{j1}(Z), ..., S_{j\zeta}(Z)]^T \in R^{N_\zeta}$, with $N_\zeta < N$, $|s_{ji}| > \iota$, $(ji = j1, ..., j\zeta)$, ι is a small positive constant, $W_\zeta^* = [w_{j1}^*, ..., w_{j\zeta}^*]^T$, and ϵ_ζ is the approximation error, with $\left||\epsilon_\zeta| - |\epsilon|\right|$ being small.

III. Neural Network Training Setup

Generally, obtaining accurate motion control of a robot requires developing its model first. For this purpose, a newly developed machine learning algorithm called the discrete-time Deterministic Learning (DL) algorithm will be employed for model learning.

Considering $f(x; p)$ as the unknown dynamics of the system, x as system state with $x(t_0) = x_0$ initial condition and p vector as the constant parameter of the system, the general form of a nonlinear dynamical system can be expressed as follows:

$$\dot{x} = f(x; p) \tag{1}$$

The discretized representation of this equation using Euler approximation can be written as follows:

$$x[k+1] = x[k] + T_s f(x[k]; p), \qquad x[0] = x_0. \tag{2}$$

$f(x[k]; p)$ is the unknown system dynamics to obtain from system states, and T_s is the time increment for each step.

Using discrete-time deterministic learning theory [24], the unknown dynamics of the system can be identified with the following algorithm:

$$\hat{x}[k+1] = x[k] + a(\hat{x}[k] - x[k]) + T_s \hat{W}^T[k+1] S(x[k]) \tag{3}$$

Where $\hat{x} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]$ and $0 < |a| < 1$ are the state vector and constant parameter respectively. $T_s = 0.12$ seconds is discretization time. $\hat{W}^T S(x[k])$ is the radial basis function neural network while $\hat{W}$ is the neural network weight which is updated using the learning law below:

$$\hat{W}[k+1] = \hat{W}[k] - \frac{\alpha P(e[k] - ae[k-1]) S(x[k-1])}{1 + \lambda_{max}(P) S^T(x[k-1]) S(x[k-1])} \tag{4}$$

Where $e[k] = \hat{x}[k] - x[k]$ the matrix $P = P^T > 0$ has the maximum eigenvalue λ and $\alpha \in (0, 2)$ is the learning gain for the design.

IV. The Soft Trunk Robot

A. Design and Fabrication

Our case study for this experiment is the Soft Trunk Robot (STR) (shown in Fig. 1) which is made of silicone rubber to enable a flexible motion in a 3-dimensional space.

The STR has consisted of 6 contiguous flattened spheres mounted from the base and tapering off to the tip point. Placing six 3D-printed retainers with the same pattern in between each sphere separately, not only stabilized the STR's movement but also holds the four strings all around the trunk to have precise motion. Each string is released and tied up by a stepper motor from the on-top base to actuate the STR. Therefore, the tip point can achieve a sphere-like motion range all around the STR.

The motors are coordinated in pairs to optimize the motion range and enable the tip point to reach the target with minimum oscillation. Each motor automatically either releases or pulls its corresponding string to reach the desired target optimally without unnecessary tension on the strings. This also improves the accuracy of the STR's positional states.

B. Actuation and State Definition

The STR is actuated by four stepper motors responsible for pulling or releasing strings attached to the tip. The STR tip point can be bent toward each direction by pulling the corresponding strings and releasing others. Thus, all motors work simultaneously in a heterogeneous environment to achieve precise trajectories.

Since the STR has a 3d motion plan, the x and y axis are defined across each pair of motors and the z-axis is defined toward the central axis from top to bottom. Stepper motors are fixed on the top base diagonally. Fig. 2 shows defined pair motors and their corresponding axis.

In order to make the STR motion more efficient and simple, each opposite motor works in pairs. Motors 1 and 3 from the first pair and motors 2 and 4 from the second pair, with each pair working independently. The first pair of motor bends the robot on the y-axis while the second pair bend on the x-axis,

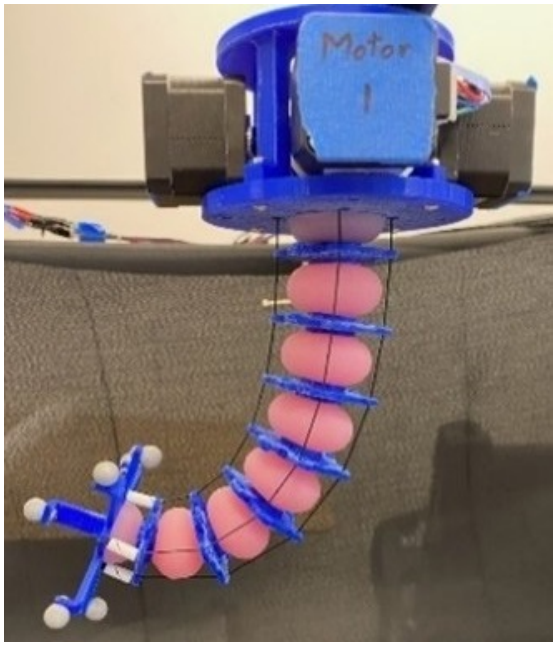

Fig. 1. Soft Trunk Robot.

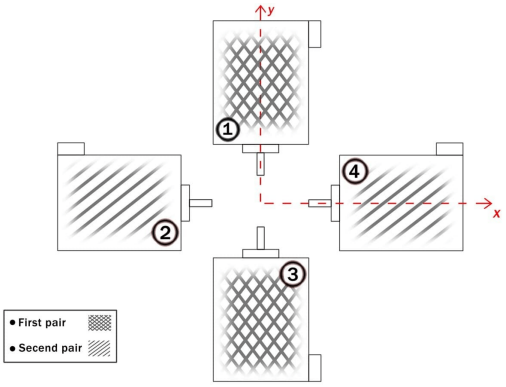

Fig. 2. Pair motors design overview and corresponding axis.

as shown in the picture. Correspondingly, the z-axis is defined from top to bottom of the robot.

The system inputs consist of two variables (u_1 and u_2) and each corresponds to one pair of motors. Positive and negative values indicate whether the first or the second motor of each pair is pulling or releasing the string respectively. For instance, if the first input value (u_1) is negative, it indicates that the second motor of the first pair is pulling its string, while the first motor is releasing its corresponding one. As the STR has a 3D motion plan, there are three positional data (x,y,z) to be collected as system positional states. Counting each pair of motors as one state as input and the three positional states, the overall state of the system includes u_1, u_2, x, y, and z.

V. Dynamic Estimation using the Average of NN Weights

To obtain the best estimation of the system dynamics, a well-trained NN weight of the model from an original trajectory is needed. The original trajectory for this experiment was generated by five continuous similar loops that start and end in a predefined position. Each loop consists of 120 steps, resulting in an overall trajectory of 600 steps. The NNs dimension is defined as 12, and with 5 system states, the number of NN nodes becomes 248,832. Also, in the training procedure, NN weights progressively converge to their optimal value Therefore, the latest learned NN weights are more reliable compared to the previous ones, and the average values of NN weights from the latest steps of the learning process (including all NN nodes) was derived to be used in the estimator. In fact, The average of NN weights, being a more balanced weight, helps the estimator achieve a more accurate prediction. The average of the latest 120 NN weights from the trained model was calculated as average weights:

$$\bar{W} = \frac{1}{k_b - k_a + 1} \sum_{k=k_a}^{k_b} \hat{W}[k] \tag{5}$$

Where $[k_a, ..., k_b]$ represents the range of steps (481-600) chosen to calculate the average weight $\bar{W}$, using the corresponding neural network weights $\hat{W}[k]$ from these steps.

According to [24], motion dynamics from the neural network can be accurately approximated by fully estimating dynamics information and average weights based on the following equation:

$$\bar{X}[k+1] = \bar{X}[k] + T_s B(\bar{X}[k] - X[k]) + T_s(\bar{W})^T S(X[k]) \tag{6}$$

With the real state of the trajectory as $X[k]$, $B = diag[0.5, 0.5, 0.5, 0.5, 0.5]$ as a diagonal matrix, the estimated states of the original trajectory is defined as $\bar{X}[k]$ in state estimator process.

Fig. 3 shows that the estimated dynamics align with the actual ones for all state values. The convergence of the corresponding error to a small neighborhood of zero in just a few steps demonstrates the estimator's precision in predicting the model dynamics. This outcome is achieved by implementing the adaptive learning algorithm with the average weights. Therefore, based on the accurate modeled dynamics of the robot, a fast FD can be achieved.

VI. Fault Detection procedure and Results

In the fault detection process, our objective is to identify when a fault happens. To achieve this, we use the fully estimated dynamics information associated with the original trajectory and implement the actual state of a faulty trajectory in the estimation process. When a significant error arises along the original trajectory, it indicates a faulty trajectory. The process diagram in Fig. 4 illustrates the learning process and FD in detail.

As shown in Fig. 4, by employing RBF NN, the unknown dynamics of the original trajectory is identified with the DL algorithm. The average of converged NN weights is then used in the estimator process along with the faulty trajectory motion dynamics. The fault detection procedure can be represented by calculating the norm error and determining a threshold.

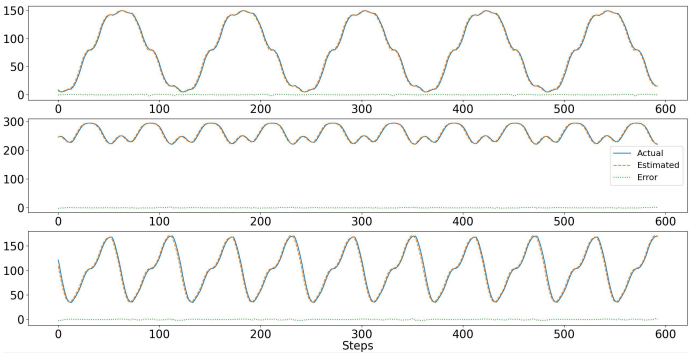

Fig. 3. Dynamics comparison between actual system states and predicted system model states for x, y, and z coordinates from top to bottom respectively).

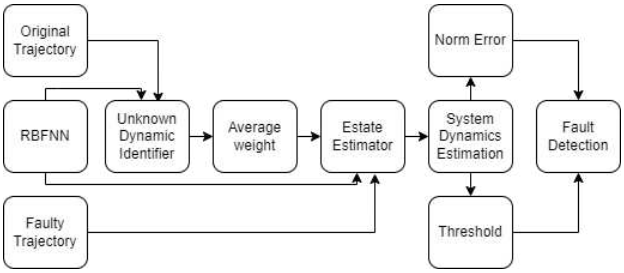

Fig. 4. Process diagram of the learning procedure.

A. *Fault Implementation*

Four different faults are defined to occur for this experiment:

1) Hanging an external weight from the tip point.
2) Holding the trunk near a fixed object by an elastic band.
3) Fastening a cable tie over the robot and tightening the strings to one sphere.
4) Motor shut down due to the power source issue or being disabled.

Fig. 5 shows how each fault is generated. Each of these four faults affects the actuation of the STR in different ways. Understanding how these faults affect the robot's performance is crucial for developing an effective fault detection procedure and ensuring the safe operation of the robot in real-world applications. The comparison between the actual system dynamics and approximated dynamics of faulty trajectories and their related errors are plotted in Fig. 6 and Fig. 7.

The estimation results before the fault occurrence of Fig. 6 and Fig. 7 demonstrate that the error converged to a small neighborhood of zero because the trajectory is identical to the original trajectory up to that point, and the dynamic estimation results are highly similar. However when a fault happens, the dynamics of the robot change, leading to a deviation in the state estimation results, and causing an increase in the corresponding error. In fact, The error seems to amplify noticeably following the fault event. This characteristic is utilized for fault detection procedures.

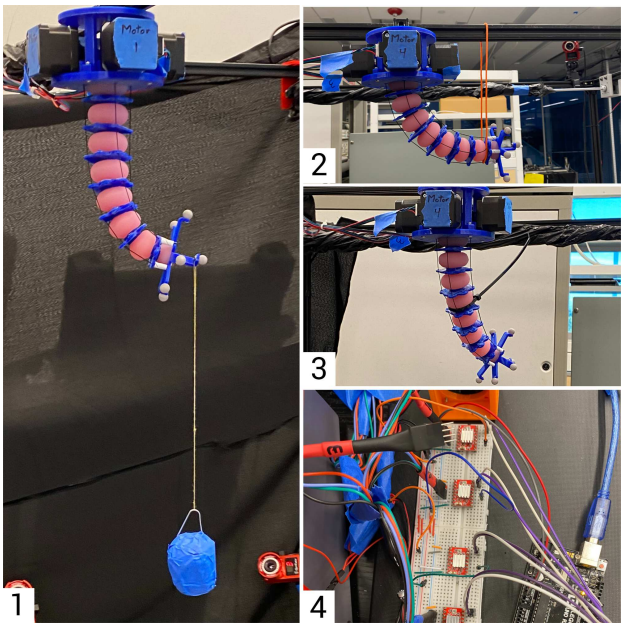

Fig. 5. Four defined faults: 1. Weight hanging. 2. Elastic band. 3. Cable tie. 4. Motor Shutdown(unplugged).

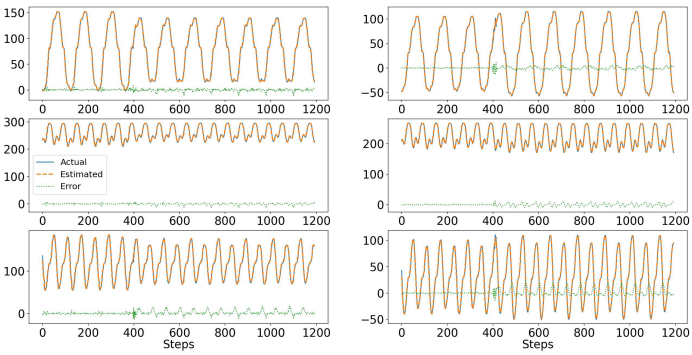

Fig. 6. Dynamic comparison between the actual system dynamics and approximated one of faulty trajectory and their corresponding error for x, y, and z position from top to bottom respectively. Fault 1 (left) and Fault 2 (right).

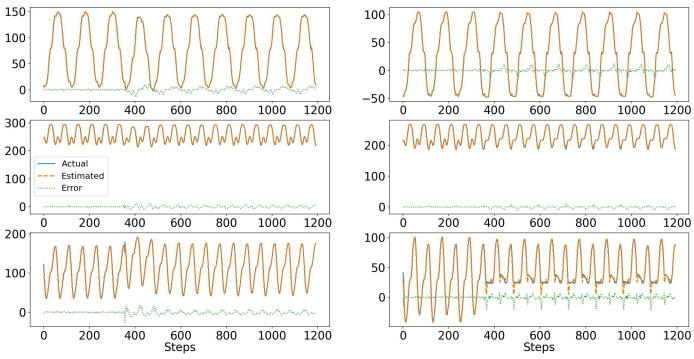

Fig. 7. Dynamic comparison between the actual system dynamics and approximated one of faulty trajectory and their corresponding error for x, y, and z position from top to bottom respectively. Fault 3 (left) and Fault 4 (right).

B. Fault Detection Procedure

The estimation error generated by the fault needs to be analyzed for the FD procedure. To detect a fault accurately, the estimation error has to exceed a predefined limit. To address what the exact limitation value is, we define the norm error of the estimation error and then a fault detection threshold can be implemented. The norm error is the absolute average of the estimation error for a maximum of 120 steps. the reason to consider the norm of the error is that the oscillation error would be neglected when using the norm of the estimation error.

C. FD Results

An intuitive approach to fault detection is defining a fixed threshold above the norm error of the estimation error. In this way, if the error happens, it can be detected as the norm error value starts surpassing the predefined threshold. By setting a threshold for the increased norm error resulting from a fault, fault occurrence can be easily detected. If the norm error does not exceed the threshold value, the trajectory is considered as healthy (remaining the same as the original one). On the other hand, if it remains larger than the upper threshold, it diagnoses as a faulty trajectory.

Three different trials of each fault were conducted on the robot. The procedure outlined in Fig. 4 was examined and their norm errors were plotted for each positional state. Specific threshold values were then defined for each trial separately. Fig. 8 and Fig. 9 display the norm error and the threshold corresponding to each fault for three different trials.

The norm error initially increased until it reaches the 120 steps in Fig. 8 and Fig. 9, remaining constant until the error occurs. It undergoes another constant growth (after the 360th step) when a fault happens. The threshold value can be determined for each positional state at the beginning of the constant growth of the norm error when a fault occurs. This value is defined after studying all three trials for each fault independently. Additionally, a fault detection time can be established for each fault. The fault detection time refers to the time interval between the occurrence of a fault and the moment when the norm error surpasses its designated threshold.

Table I presents the general threshold values of each fault for the x,y, and z direction. The threshold values indicate the limitation of norm error corresponding to each positional state of each fault. The fault detection time also reveals how long it takes for all norm errors corresponding to each positional state to reach their threshold values after fault occurrence. The maximum value among the trials is designated for FD time. For instance, when a fault happens due to the cable tie, it takes 0.96 seconds to detect the fault.

Fault detection duration from Table I demonstrates how quickly each fault can be detected. Since safety is one of the highest priorities for soft robot applications for human interactions, minimizing fault detection duration is crucial.

In this experiment by defining corresponding threshold values and using the norm error for each positional state, fault detection time is minimized. This accomplishment is also because of the precise prediction of dynamic estimation using deterministic learning with an average of constant RBFNN weights. The fault detection process further ensures that different norm errors can be defined for different possible faults with FD time.

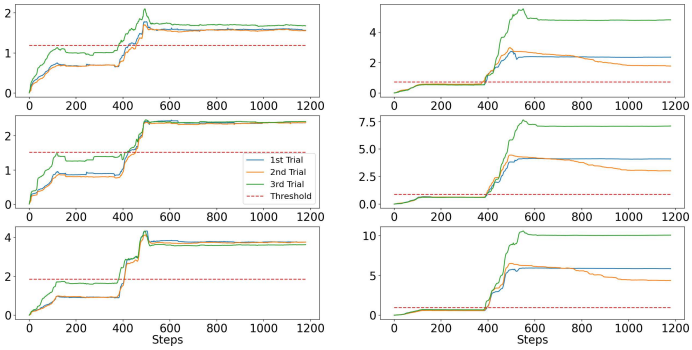

Fig. 8. Three different norm errors of faulty trajectories and their corresponding threshold for each state position (x,y,z). Fault 1 (left) and Fault 2 (right).

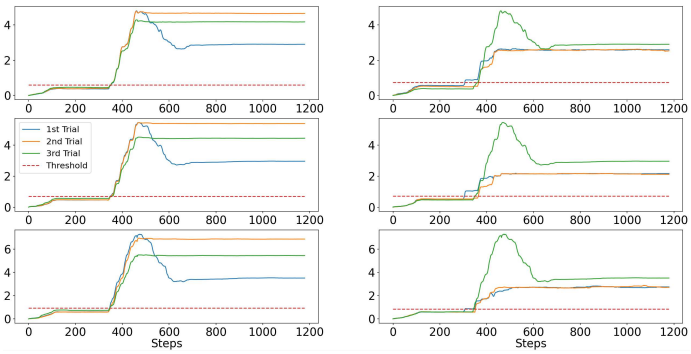

Fig. 9. Three different norm errors of faulty trajectories and their corresponding threshold for each state position (x,y,z). Fault 3 (left) and Fault 4 (right).

VII. CONCLUSIONS

In this paper, we presented a novel machine learning approach, deterministic learning, for modeling a soft trunk robot using a radial basis function neural network. By employing average NN weights, the error between the actual system dynamics and dynamic estimation converged to a small neighborhood of zero, resulting in a more accurate model estimation.

Subsequently, based on the model and the corresponding error of dynamic estimation, we implemented a fault detection process to identify the occurrence of four different faults. For each fault, a threshold value was established for each positional state to facilitate fault identification. Additionally, fault detection time was determined for each distinct fault. The results validated the accuracy of the model and demonstrated the rapid detection of each fault through various trials.

TABLE I
THRESHOLD VALUES FOR EACH DIRECTION AND FAULT DETECTION TIME FOR EACH FAULT

Fault	Threshold Values (mm)			FD Time (s)
	x	y	z	
1. Weight hanging	1.19	1.52	1.85	1.20
2. Cable tie	0.72	0.88	0.96	0.84
3. Elastic band	0.59	0.69	0.92	0.96
4. Motor shutdown	0.73	0.71	0.83	1.08

REFERENCES

[1] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft robot arm inspired by the octopus," *Advanced robotics*, vol. 26, no. 7, pp. 709–727, 2012.

[2] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics*, vol. 3, no. 16, p. eaar3449, 2018.

[3] S. Seok, C. D. Onal, K.-J. Cho, R. J. Wood, D. Rus, and S. Kim, "Meshworm: a peristaltic soft robot with antagonistic nickel titanium coil actuators," *IEEE/ASME Transactions on mechatronics*, vol. 18, no. 5, pp. 1485–1497, 2012.

[4] N. G. Cheng, M. B. Lobovsky, S. J. Keating, A. M. Setapen, K. I. Gero, A. E. Hosoi, and K. D. Iagnemma, "Design and analysis of a robust, low-cost, highly articulated manipulator enabled by jamming of granular media," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 4328–4333.

[5] Z. Wang, Y. Torigoe, and S. Hirai, "A prestressed soft gripper: design, modeling, fabrication, and tests for food handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1909–1916, 2017.

[6] C. Li, X. Gu, and H. Ren, "A cable-driven flexible robotic grasper with lego-like modular and reconfigurable joints," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2757–2767, 2017.

[7] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[8] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, "Discrete cosserat approach for multisection soft manipulator dynamics," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1518–1533, 2018.

[9] M. Thieffry, A. Kruszewski, T.-M. Guerra, and C. Duriez, "Lpv framework for non-linear dynamic control of soft robots using finite element model," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 7312–7318, 2020.

[10] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches," *IEEE transactions on industrial electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[11] H. Gu, H. Wang, and W. Chen, "Toward state-unsaturation guaranteed fault detection method in visual servoing of soft robot manipulators," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3942–3947.

[12] H. Gu, H. Hu, H. Wang, and W. Chen, "Soft manipulator fault detection and identification using anc-based lstm," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1702–1707.

[13] T. S. Le, H. Schlegel, W. G. Drossel, and A. Hirsch, "Fault detection and fault-tolerant control when using sma actuators in soft robotics," in *Solid State Phenomena*, vol. 260. Trans Tech Publ, 2017, pp. 92–98.

[14] J. Zhang, X. Chen, P. Stegagno, and C. Yuan, "Nonlinear dynamics modeling and fault detection for a soft trunk robot: An adaptive nn-based approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7534–7541, 2022.

[15] G. Zheng, Y. Zhou, and M. Ju, "Robust control of a silicone soft robot using neural networks," *ISA transactions*, vol. 100, pp. 38–45, 2020.

[16] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2020, pp. 417–423.

[17] P. Wu, W. Jiangbei, and F. Yanqiong, "The structure, design, and closed-loop motion control of a differential drive soft robot," *Soft robotics*, vol. 5, no. 1, pp. 71–80, 2018.

[18] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[19] Z. Y. Ding, J. Y. Loo, V. M. Baskaran, S. G. Nurzaman, and C. P. Tan, "Predictive uncertainty estimation using deep learning for soft robot multimodal sensing," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 951–957, 2021.

[20] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 26, 2019.

[21] R. F. Reinhart and J. J. Steil, "Hybrid mechanical and data-driven modeling improves inverse kinematic control of a soft robot," *Procedia Technology*, vol. 26, pp. 12–19, 2016.

[22] J. F. Queißer, K. Neumann, M. Rolf, R. F. Reinhart, and J. J. Steil, "An active compliant control mode for interaction with a pneumatic soft robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 573–579.

[23] C. Yuan and C. Wang, "Performance of deterministic learning in noisy environments," *Neurocomputing*, vol. 78, no. 1, pp. 72–82, 2012.

[24] ——, "Design and performance analysis of deterministic learning of sampled-data nonlinear systems," *Science China Information Sciences*, vol. 57, no. 3, pp. 1–18, 2014.

[25] C. Wang and D. J. Hill, "Learning from neural control," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 130–146, 2006.

[26] ——, "Deterministic learning and rapid dynamical pattern recognition," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 617–630, 2007.

[27] J. Trivisonno, C. Amaral, L. Garofalo, X. Chen, E. Jandaghi, and C. Yuan, "Automatic control of a soft trunk robot actuated by strings," *IEEE MIT Undergraduate Research Technology Conference*. [Online]. Available: https://par.nsf.gov/biblio/10345167

[28] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[29] ——, "Approximation and radial-basis-function networks," *Neural computation*, vol. 5, no. 2, pp. 305–316, 1993.