

Application of a State-Space based and a LSTM Neural Network for Identification of the Dynamic System of a Soft Robot

Franklin Andrade de Brito^{1*}, Ivon Luiz Martinez Garcia¹, Taniel Silva Franklin¹, Lucas Cruz da Silva²

¹ SENAI CIMATEC, BIGDATA, Salvador, Bahia, Brazil

² SENAI CIMATEC, ROBOTICS, Salvador, Bahia, Brazil

*SENAI CIMATEC; franklin.brito@fbter.org.br;

Abstract: The control process of a nonlinear robotic system is a complex task that has been the subject of study in the literature. The ability to identify a dynamic system of a Piecewise Constant Curvature Soft Robot (PCC) with respect to the robot's trajectory over time has shown potential to improve when applying Machine Learning techniques, which positively affects the ability to control the system. In this work, both a state-space-based neural network approach and a Long Short-Term Memory (LSTM) approach are proposed to verify their potential in predicting sequential time series data using simulated PCC trajectory datasets. The results provide a comparison between the two proposed models' performances, presenting positive findings for both approaches, and basis for future studies and applications.

Keywords: Control. Soft robot. Neural Network. System Identification.

Abbreviations: PCC, Piecewise Constant Curvature. LSTM, Long Short-Term Memory.

1. Introduction

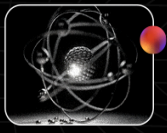
The control and identification of dynamic systems in Soft Robots have been widely studied due to their complexity and potential applications. A frequent modeling approach is the Piecewise Constant Curvature (PCC) model, which simplifies continuous dynamics by segmenting the robot into constant-curvature parts. Despite its efficiency, this model faces challenges due to system nonlinearity and real-world factors like resistance forces [1,6], making robust identification methods essential [3].

Since soft robotic manipulators follow time-sequenced positions, the problem is inherently sequential, which aligns with the strengths of machine learning methods, especially Recurrent Neural Networks (RNNs) [4]. Within this category, Long Short-Term

Memory (LSTM) networks have proven effective in identifying complex systems [5].

In parallel, state-space-based neural networks have also shown promising results in data-driven modeling of dynamic systems [7]. A complementary approach involves state-space modeling with L-BFGS-B optimization to identify linear and nonlinear systems, notably in industrial robotics, integrating L1 and group-Lasso regularizations to handle nonlinear complexity [6].

This study contributes to the state of the art by exploring both state-space-based neural networks and LSTM techniques to identify a nonlinear two-segment Soft Robot manipulator. The identification is based on sequential time series data derived from a simulated 3D trajectory using the PCC model. The objective is to assess how well each technique captures the manipulator's dynamics and to analyze their



performance based on the results.

This work contributes mainly with literature review proposed approaches, state-space-based neural Network and LSTM for soft robot dynamic system identification, and provides a comparative discussion of the results.

This work has 4 sections. Section 2 presents the methodology used to structure the experiments and resulting analyses. Section 3 explores and discusses the results and Section 4 presents a final conclusion.

2. Methodology

In this section, the implementation method used for both approaches is presented.

2.1 Approach Using LSTM

As mentioned, due to its proven effectiveness for time-series prediction problems, the LSTM approach was investigated in this work.

The data collection was gathered based on the trajectory simulation of a 2-segment Soft Robot manipulator. The trajectory was obtained by numerically solving an ODE (Equation 1) based on a PCC model [6] that described the dynamics and kinematics of the system.

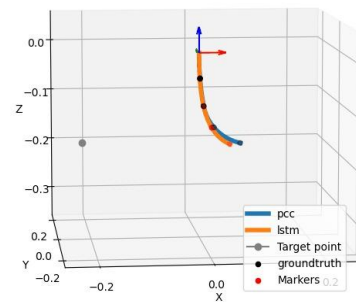
$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + D\dot{q} + Kq = \tau \quad (1)$$

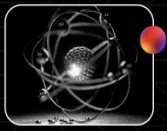
where: $q \in \mathbb{R}^n$ is the vector of generalized coordinates (the curvatures of the segments).

$M(q) \in \mathbb{R}^{n \times n}$ is the configuration-dependent mass-inertia matrix; $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$ is the matrix of Coriolis and centrifugal forces; $D \in \mathbb{R}^{n \times n}$ is the material damping matrix; $K \in \mathbb{R}^{n \times n}$ is the material stiffness matrix; $\tau \in \mathbb{R}^{n \times n}$ is the vector of actuation torques applied to the system.

The simulated robot consists of two segments, each with a length of 0.1036 m and a mass of 1.0 kg, operating exclusively in the negative Z region of Cartesian space. The experiment involves a sequence of movements generated by the PCC model over defined timesteps, with the goal of predicting the next set of manipulator actions. The dataset includes 4 input values representing applied torque and 12 output values corresponding to the x, y, and z coordinates of 4 marker points along the manipulator, as shown in Figure 1.

Figure 1. LSTM prediction on a simulation frame compared to the ground truth (PCC model).





The hyperparameters used to train the LSTM model are listed at Table 1. In order to obtain the dataset, it was necessary to input initial values for the curvature angle (θ) and the azimuthal angle (φ) as well as the torque (τ) applied to the robot.

Table 1. Summary of training parameters for the LSTM-based model.

Parameter	Value
Scaler range	$[-1, 1]$
Time steps	500
n_{dense} (Dense layer size)	800
Epochs	200
η_{min} (Minimum learning rate)	5×10^{-5}
Decay factor	0.2
LSTM units	256
Batch size	500
Stateful	TRUE
Return sequences	TRUE
Dropout	0.2

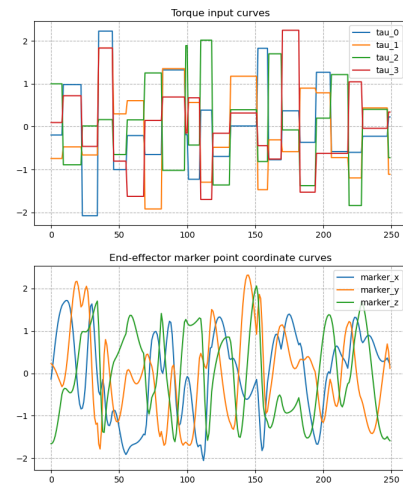
Figure 2 presents a sample of dataset values throughout time where the upper plot represents the τ values applied to the system and the lower plot represents the end-effector marker point output trajectory.

Tests were carried out using training datasets ranging from 12,000 to 64,000 samples with static hyperparameters to determine the optimal data volume for training the LSTM model.

The data was normalized according to

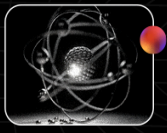
the *scaler range* in Table 1, and a grid search was performed to minimize validation loss and improve the R^2 -score. The test dataset was fixed at 2,000 samples to ensure consistent evaluation of training set size effects. The model's prediction effectiveness was also assessed.

Figure 2. Dataset sample chart (500 timesteps).



In the LSTM approach, the input phase specifies the data format, including batch size, number of features, and time steps to align with the training structure. The *LSTM phase* processes the time series using a set number of LSTM cells possibly stacked and a *tanh* activation function for cell state updates, allowing the network to learn temporal patterns and internal states.

The *dense phase* uses a fully connected layer with 20% dropout and a *ReLU* activation function to introduce non-linearity, prevent vanishing gradients, and enforce non-negative outputs. The *compiler phase* sets the loss functions (mse and mae), the Adam optimizer, and the R^2 -score as the evaluation metric. Lastly,



the *output phase* produces a multi-step sequence of predicted robot positions, increasing the complexity of the task.

2.2 Approach using State-space-based Neural Network

Following the same procedure as with the LSTM, tests were conducted using identical training dataset sizes and fixed hyperparameters to assess how data quantity affects the performance of the state-space nonlinear model. This method combines linear and nonlinear components, with the nonlinear part modeled by a Neural Network, forming a gray-box approach in contrast to the black-box nature of LSTM.

The Jax-sysid package [5] was used to define and train the neural state-space model, leveraging just-in-time (JIT) compilation, automatic differentiation, and GPU acceleration, all of which support efficient training of dynamic models with high-dimensional parameter spaces.

The *input*, *compiler* and *output* phases are analog to the ones described in the LSTM section. The *definition* phase, related to the State Function & Output Function, describes how the model hypothesis is formed along, defining the model structure. The *initialization* phase defines the parameters initialization regarding the structure of the state-space system representation and weights of the neural network.

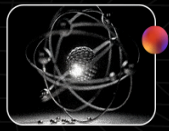
The system identification model combines linear state-space components with neural network-based nonlinearities, represented by Equations 1 and 2:

$$x_{k+1} = Ax_k + Bu_k + W_3 \sigma(W_1 x_k + x_k + W_2 u_k + b_1) + b_2 \quad (2)$$

$$y_k = C x_k + W_5 \sigma(W_4 x_k + b_3) + b_4 \quad (3)$$

where $x_k \in R^{n_x}$ is the state vector, $u_k \in R^{n_u}$ is the input vector, $y_k \in R^{n_y}$ is the output vector, and σ represents the sigmoid activation function. The matrices were initialized with small scales to ensure stable training by avoiding exploding gradients. System matrix A initialized as $0.5I_{n_x}$ to provide moderate state persistence. Input matrix B had random initialization with $\mathcal{N}(0,0.1)$ weights. Output matrix C had Random initialization with $\mathcal{N}(0,0.1)$ weights. Neural network weights W_1 - W_5 initialized with scales between 0.1 and 0.5 to balance linear and nonlinear contributions. Bias terms b_1 - b_4 initialized to zero as common practice

To ensure reproducible convergence, a seed was used to control for the randomness in the matrix initialization process. The learning rate was kept at $\eta_{ss} = 0.01$. The sampling time was also set as constant, $T = 100$ ms. The regularization parameters were fixed at



$\rho_{\theta} = \rho_x = 0.00001$. The model order was chosen as $n_x = 24$. With respect to the test data, x_0 was reconstructed by performing 100 epochs of Rauch-Tung-Striebel (RTS) smoothing [7]. For the optimization of the algorithm, the L-BFGS-B and Adam methods were used while varying their respective iterations to find the best results.

Table 2. Hyperparameters used for State-space-based training and evaluation.

Hyperparameter	Value
Seed usage	Yes
Sampling time (T)	100 ms
Learning rate (η_{ss})	0.01
Regularization (p_{θ} , p_x)	0.00001
Model order (n_x)	24
RTS smoothing epochs	100

3. Results and Discussions

In this section, the relevant results found by both approaches are presented.

3.1 LSTM Approach

For the LSTM approach, the results showed effectiveness using a training dataset size of 23500 and a testing dataset size of 2000, 256 LSTM units, 800 dense layer neurons, 5×10^{-5} as the initial learning rate with decay over time, and only Adam as the optimizer. The best model

found had a validation loss of 0.0227 and a training loss of 0.0095.

Table 3 presents an overview of the R^2 -score values obtained from the LSTM model after test predictions.

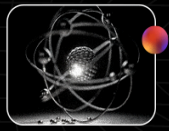
Table 3. Test R^2 -scores for each marker coordinate using LSTM

Marker Point	R2-score (x, y, z)
Marker 1	(10.04%, 8.37%, 24.09%)
Marker 2	(12.41%, 18.29%, 25.04%)
Marker 3	(90.29%, 13.22%, 27.73%)
Marker 4	(38.44%, 16.68%, 17.16%)

3.1.1 Overall Performance and Interpretation

During training, the LSTM model performs very well for x-coordinates (R^2 between 85% and 90%), showing it effectively captures longitudinal motion. For z-coordinates, performance is moderate (R^2 between 13% and 28%), suggesting partial learning of vertical motion, possibly due to limited lateral movement, data imbalance, or complex lateral dynamics.

In the test phase, generalization varies: for the x-axis, three out of four markers show large R^2 drops (e.g., from 88% to 10%), suggesting overfitting, while Marker 3 retains strong generalization (R^2 of 90.29%). For z-coordinates, R^2 remains moderate (17% to 28%), similar to or slightly better than training. Interestingly, y-coordinate predictions improve during testing (R^2 between 8 and 18), unlike the



poor or negative values during training, possibly due to less noisy or more structured lateral motion in the test sequence.

3.1.2 Axis-Wise and Marker-Wise Analysis

To better understand the model's performance, Table 4 summarizes the approximate average R^2 scores across all markers for each axis.

Table 4. Average R^2 -scores by axis across all markers

Axis	Avg. R2	Performance comment
x	38%	Moderate, possibly affected by overfitting
y	13%	Low predictive power
z	24%	Regular

From a spatial perspective, Marker 3 emerges as the most consistently predicted across both training and testing phases, especially for the x and z coordinates. This suggests that the motion at this location is more regular or less prone to non-linear effects than at other markers, making it more learnable by the LSTM. Conversely, Marker 1 and Marker 2 show particularly poor generalization for the x and y coordinates, possibly because they are located in more dynamic regions of the robot's body where deformations are more nonlinear and harder to model.

3.1.3 Discussion on Soft Robot Dynamics and Model Behavior

The soft robot's nonlinear, time-dependent behavior challenges modeling. While the LSTM captures main longitudinal trends, it struggles with lateral and vertical motions, likely due to limited training data, sequence length, or model capacity.

The LSTM's poor y -axis performance likely stems from data imbalance or small, noisy displacements, while overfitting in x indicates memorization of training sequences rather than true generalization. Incorporating physics-based regularization and prior knowledge could improve convergence and overall model performance.

3.2 State-Space Based Approach

After a grid search of the appropriate dataset size for the present dynamical system identification problem, the training dataset size of 23500 and the testing dataset size of 2000 were chosen along with 5000 iterations of the L-BFGS-B and Adam optimizers and the seed = 42 for the repeatability of initialization of the matrices, weights and bias parameters.

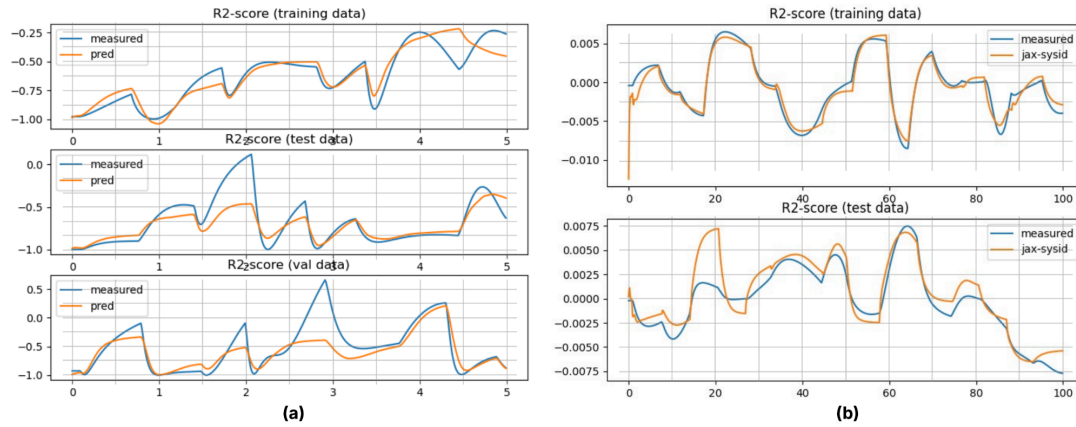
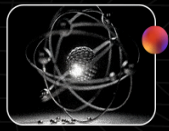


Figure 3. Best coordinate R2-Score comparison between ground truth and prediction made by LSTM (a), and state-space based neural network (b).

3.3 Quantitative and Qualitative Analysis

The R^2 -score values show that the State-space-based model effectively learns and generalizes the 3D behavior of the soft robot with high accuracy. Unlike traditional models that depend on physical simplifications, this data-driven approach captures nonlinear dynamics, particularly along the horizontal axes.

For all marker points, x and y predictions perform strongly, with R^2 -scores mostly above 70% and many surpassing 80%, indicating structured, repeatable horizontal behavior. For example, Marker 3 achieves 79.73% (x) and 83.24% (y), reflecting stable and learnable dynamics in that region. However, performance in the vertical (z) axis is more variable. While Markers 2 and 3 show good results (65.59% and 76.25%, respectively), Marker 1 performs poorly in z, with a score of 29.86%, possibly due to complex

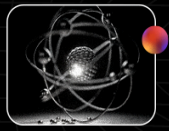
Table 5. Test R^2 -scores for each marker coordinate using State-space-based NN.

Marker Point	Test R2-Score (x, y, z)
Marker 1	(82.45%, 79.37%, 29.86%)
Marker 2	(79.68%, 82.38%, 65.59%)
Marker 3	(79.73%, 83.24%, 76.25%)
Marker 4	(70.59%, 64.76%, 54.38%)

deformation near the base or insufficient training data on vertical transitions.

Figure 3 presents a plot of R^2 -Score, illustrating the proximity between the ground truth (blue line) and the best prediction (orange line) over time for each approach.

Notably, the State-space-based model shows greater consistency across markers and axes compared to the LSTM model, which had issues like overfitting and negative R^2 in y. In contrast, even the lowest z-score in State-space-based is positive, and all coordinates across all markers present a meaningful level of predictive performance. Table 6 summarizes the



state-space-based performance based on R^2 score average across markers.

Table 6. Average R^2 -scores by coordinate axis across all markers.

Axis	Avg. R^2	Performance Comment
x	78.61%	Consistent across markers
y	77.81%	Best overall performance;
z	56.52%	Potentially higher nonlinearity

Analyzing the standard deviation of the R^2 score, it's noted that the z-axis not only has lower mean R^2 (56.52%) but also the highest variability ($\pm 17.22\%$), confirming instability in predictions compared to x and y.

4. Conclusions

Soft Robot control has been evolving, offering new possibilities for implementation and advancement in this research area. This work presented results from both LSTM and state-space-based neural network approaches for identifying a nonlinear Soft Robot dynamic system. Both methods demonstrated adequate performance and learning potential, with the state-space-based method standing out for its effectiveness and future applicability.

A valuable direction for future work includes training the models on Soft Robots with varying segment numbers, exploring hyperparameter fine-tuning, applying autoencoders for dimensionality reduction, and

evaluating the models using different performance metrics.

Acknowledgement

This research was carried out in partnership between SENAI CIMATEC and Shell Brasil Petróleo LTDA. The authors would like to thank Shell Brasil Petróleo LTDA, the Brazilian Company for Industrial Research and Innovation (EMBRAPII) and the National Agency of Petroleum, Natural Gas and Biofuels (ANP) for their support and investments in RDI.

References

- [1] C. Della Santina, C. Duriez, and D. Rus, "Model-based control of soft robots: A survey of the state of the art and open challenges," IEEE Control Systems Magazine, 2023.
- [2] A. Rostamijavanani, S. Li, and Y. Yang, "Data-driven identification of nonlinear dynamical systems with LSTM autoencoders and normalizing flows," arXiv preprint arXiv:2503.03977, 2025.
- [3] J. Gonzalez and W. Yu, "Non-linear system modeling using LSTM neural networks," IFAC-PapersOnLine, vol. 51, no. 13, pp. 485–490, 2018.
- [4] A. Bemporad, "An L-BFGS-B approach for linear and nonlinear system identification under l_1 and group-lasso regularization," IEEE Transactions on Automatic Control, 2025.
- [5] J. Bradbury, R. Frostig, P. Hawkins, M. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. Vander-Plas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," GitHub repository, <https://github.com/google/jax>, 2018. [Accessed: Jul. 29, 2025].
- [6] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," Applied Bionics and Biomechanics, vol. 5, no. 3, pp. 99–117, 2008.
- [7] H. E. Rauch, C. T. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems," AIAA Journal, vol. 3, no. 8, pp. 1445–1450, 1965.