ICRA2026 Late Breaking Results Poster presented at
2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

nnodely – neuralize your model

Gastone Pietro Rosati Papini, Alice Plebe, Mojtaba Sharifzadeh, Mattia Piazza, Sebastiano Taddei, Giovanni Scialla, Francesco Baroni, Davide De Martini, Giovanni Maria Francesco La Scala, Gioele Defrancesco, Filippo Faccini

***Abstract*— Modeling and control of physical systems remain challenging for purely data-driven methods, which often lack interpretability and fail to leverage prior knowledge. Model-structured neural networks (MSNNs) embed physical laws into neural architectures; however, their design and implementation can be nontrivial. We present `nnodely`, an open-source framework that simplifies MSNN development through a modular workflow, improving interpretability, data efficiency, and deployment on resource-constrained platforms. The paper highlights the framework's features, positions it within the landscape of existing tool, and demonstrates its effectiveness in two case studies. `nnodely` is released under the MIT license and is available at https://github.com/tonegas/nnodely.**

I. Introduction

Neural networks are increasingly used in robotics for perception, estimation, and control, thanks to their ability to capture complex nonlinear dynamics from data. General-purpose frameworks such as PyTorch and TensorFlow provide flexible learning primitives. However, their development typically follows purely data-driven workflows, resulting in models that require large datasets and lack interpretability.

Model-structured neural networks (MSNNs) address these limitations by embedding model principles [1], [2]; physical laws [3]; and control architectures [4], [5] directly into the model, improving interpretability, data efficiency, and reliability. However, their development remains complex and largely manual, requiring expertise in both modeling and machine learning.

To address this gap, we introduce `nnodely`, an open-source framework for the structured design, training, and deployment of MSNNs. By providing modular building blocks and a unified workflow, `nnodely` enables the integration of physical knowledge into neural architectures while ensuring interoperability with existing robotic systems.

II. Nnodely framework

`nnodely` is an open-source framework for developing MSNNs, providing a modular, domain-oriented interface that simplifies design, training, and deployment. It enables modeling, estimation, and control of physical systems with partially unknown or complex dynamics, supporting end-to-end optimization and policy learning while allowing domain experts to embed their knowledge into neural architectures without deep machine learning expertise. The framework standardizes input tensors, separates high-level model specification from backend implementation, and provides a growing library of components, including FIR and linear filters, fuzzy and local models, parametric functions, and equation learners, for building interpretable and physically consistent networks.

`nnodely` automates temporal data handling, dataset construction, and training routines, offering support for custom and physics-informed losses, early stopping, and validation metrics. It allows complex architectures to be composed through feedforward or closed-loop connections, supports staged optimization of interconnected sub-models, and facilitates deployment via JSON, PyTorch, or ONNX export. By systematizing the full MSNN workflow, from model design to validation and deployment, `nnodely` lowers the barrier to building structured, data-efficient, and interpretable neural models for physical systems.

III. Case studies

A. Friction Estimation of a Unitree Z1

This case study uses an MSNN to estimate and compensate for actuator friction in the Unitree Z1 manipulator. The friction model is embedded within the robot dynamics by expressing the total torque $\boldsymbol{\tau}_k$ (at the k instant) as

$$\mathbf{f_{cv}}(\mathbf{q}_k, \dot{\mathbf{q}}_k) = \mathbf{K_v}(\mathbf{q}_k)\ \dot{\mathbf{q}}_k + \mathbf{K_c}(\mathbf{q}_k)\ \text{sign}(\dot{\mathbf{q}}_k),$$
$$\mathbf{f_{st}}(\mathbf{q}_k, \dot{\mathbf{q}}_k) = \text{sign}(\dot{\mathbf{q}}_k)(\mathbf{K_{st}}(\mathbf{q}_k) - \mathbf{K_c}(\mathbf{q}_k))e^{-(\dot{\mathbf{q}}_k/\dot{\mathbf{q}}_{st}(\mathbf{q}_k))^2},$$
$$\boldsymbol{\tau}_k = \boldsymbol{\tau_a}_k - (\mathbf{f_{cv}}(\mathbf{q}_k, \dot{\mathbf{q}}_k) + \mathbf{f_{st}}(\mathbf{q}_k, \dot{\mathbf{q}}_k)),$$

where $\mathbf{K_v}, \mathbf{K_c}, \mathbf{K_{st}}, \dot{\mathbf{q}}_{st}$ are configuration-dependent parameters learned from data. The model combines a parametric friction formulation (Coulomb-viscous and Stribeck effects) with learnable mappings of $\mathbf{q}$, while the $\text{sign}(\cdot)$ is approximated using $\tanh_\alpha(\cdot)$ to ensure differentiability.

This structured friction model is integrated into a differentiable forward dynamics block implemented via the ADAM library[1], enabling end-to-end training from joint states and torques to predicted accelerations.

The dataset, collected using MuJoCo simulator, comprises 15 episodes of 500 samples each, collected at a sampling

[1] https://github.com/gbionics/adam

The authors are with the Department of Industrial Engineering, University of Trento, Via Sommarive 9, 38123 Povo (TN), Italy (email: gastone.rosatipapini@unitn.it)

This work was supported under the FIS 2 Call, Grant Assignment Decree No. 1236 adopted on 01/08/2023 by the Italian Ministry of University and Research (MUR), for the project FIS-2023-03684 'Structured neural network framework for modeling and control of autonomous systems – Neu4mes', CUP E53C24003800001.

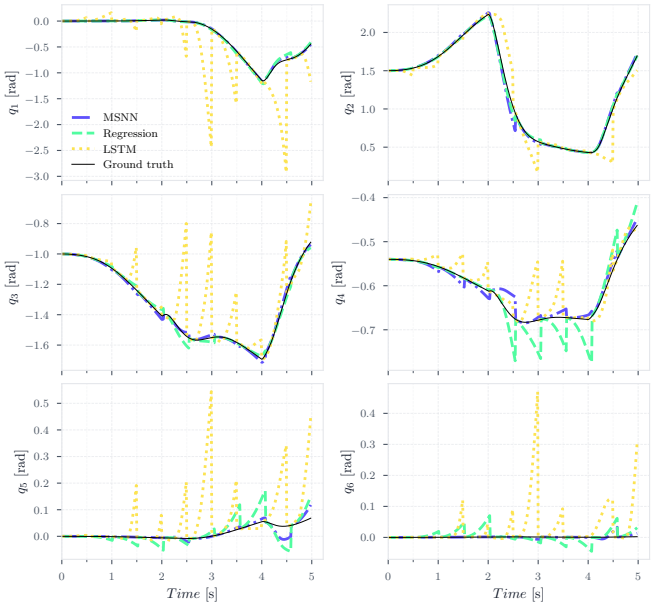

Fig. 1. Joint position prediction for the Z1 manipulator using classical system identification and MSNN. The predictions are carried out over a 0.5 s horizon, after which the system state is reinitialized to the ground truth.

frequency of 100 Hz. For comparison, friction parameters are also estimated using classical least-squares regression.

Figure 1 shows the predicted joint positions against the ground truth and the regression results. The MSNN best follows the actual trajectories, outperforming regression, with an MSE of $0.054\,\mathrm{rad}^2$ on the test set.

B. Vehicle Lateral Control Case Study

An MSNN network is proposed as a steering controller to follow a desired trajectory. The network was trained in a closed-loop configuration with a previously trained MSNN model of the vehicle, thereby enabling episodic training and ensuring robustness even outside the domain of the training telemetry data. The controller follows a target trajectory defined by future velocity and curvature. As inputs, it uses future windows of $n = 30$ samples $\boldsymbol{v_{t}}_k$, $\boldsymbol{\rho_{t}}_k$ and a past steering window $\boldsymbol{\delta_{c}}_k$ (from $k-n+1$ to $k-1$), which captures the internal state of the system. The control law is defined as:

$$\tilde{\delta}_k = K_t(a_k, v_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) + {\boldsymbol{w_{\delta_c}}}^\top \boldsymbol{\delta_{c}}_k, \tag{1}$$

where the second term represents a FIR filter. The contribution K_t combines a dynamic response term with an understeer correction function:

$$K_t(a_k, v_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) = \sum_{i=1}^{N_v} \phi_{vi}(v_k), \left({\boldsymbol{w_{\rho}}_i}^\top \nabla(a_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) \right), \tag{2}$$

where $\boldsymbol{w_{\rho}}_i$ are FIR filter weights, $\phi_{vi}(\cdot)$ are speed-depentent fuzzy membership functions. The term $\nabla(\cdot)$ is a parametric

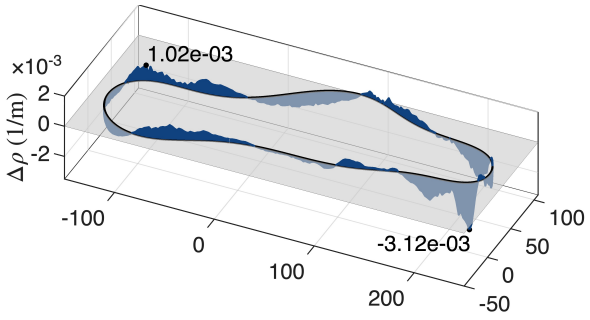

Fig. 2. Curvature tracking error $\Delta\rho$ along the test track

function compensating for understeer behaviour:

$$\nabla(a_k, \boldsymbol{v_{t}}_k, \boldsymbol{\rho_{t}}_k) = \boldsymbol{\rho_{t}}_k \odot \left(1 + A_m(a_k), \boldsymbol{v_{t}}_k^2\right), \tag{3}$$

$$A_m(a_k) = \sum_{i=1}^{N_a} \phi_{ai}(a_k), A_i, \tag{4}$$

where $\phi_{ai}(\cdot)$ are fuzzy membership functions over longitudinal acceleration and A_i are parameters inherited from the forward model.

Training and validation datasets were generated using a 14-DoF electric vehicle model in CarMaker, sampled at 20 Hz. On a separate test track (Fig. 2), the controller was tested in open-loop, achieving a curvature MSE of 4.2×10^{-7}, with a maximum lateral deviation of 2.26 m, demonstrating accurate and robust lateral tracking.

IV. Conclusion

We presented `nnodely`, an open-source framework for designing, training, and deploying model-structured neural networks for physical systems. By combining modular, interpretable components with a systematic workflow, `nnodely` enables integration of domain knowledge, improves data efficiency, and supports deployment on embedded platforms. The case studies on robotic friction estimation and vehicle lateral control demonstrate its ability to produce accurate, physically consistent models and controllers, highlighting the potential of MSNNs for real-world engineering applications.

References

[1] A. Antonucci, G. P. Rosati Papini, P. Bevilacqua, L. Palopoli, and D. Fontanelli, "Efficient prediction of human motion for real-time robotics applications with physics-inspired neural networks," *IEEE Access*, vol. 10, pp. 144–157, 2021.

[2] H. Perez-Villeda, J. Piater, and M. Saveriano, "Learning and extrapolation of robotic skills using task-parameterized equation learner networks," *Robotics and Autonomous Systems*, vol. 160, p. 104309, 2023.

[3] M. Da Lio, D. Bortoluzzi, and G. P. Rosati Papini, "Modelling longitudinal vehicle dynamics with neural networks," *Vehicle System Dynamics*, vol. 58, no. 11, pp. 1675–1693, 2020. [Online]. Available: https://doi.org/10.1080/00423114.2019.1638947

[4] M. Da Lio, R. Donà, G. P. Rosati Papini, F. Biral, and H. Svensson, "A mental simulation approach for learning neural-network predictive control (in self-driving cars)," *IEEE Access*, vol. 8, pp. 192 041–192 064, 2020.

[5] M. Piccinini, S. Taddei, M. Larcher, M. Piazza, and F. Biral, "A physics-driven artificial agent for online time-optimal vehicle motion planning and control," *IEEE Access*, vol. 11, pp. 46 344–46 372, 2023.