

# Augmented Neural Ordinary Differential Equations for Input-Output System Identification

Hannes Wolf<sup>1</sup>

Version 53de59a

<sup>1</sup> Automation and Sensorics in  
Networked Systems Group,  
University of Kassel, Germany,  
h.wolf@uni-kassel.de

## Abbreviations

**NODE** neural ordinary differential equation

**ODE** ordinary differential equation

**NN** neural network

**ML** machine learning

**MLP** multi-layer perceptron

**A-NODE** augmented neural ordinary differential equation

## Background and Motivation

In many real-world scenarios, the internal dynamics of a system are either too complex to model from first principles or are entirely unknown. Despite this, accurate models are essential for tasks such as prediction, control, optimization, or anomaly detection. Black-box system identification offers a powerful alternative by learning models directly from input-output data without requiring explicit knowledge of the system's internal structure. By relying solely on observed behavior, black-box identification provides a data-driven pathway to build accurate models. It allows engineers and researchers to develop controllers, design simulations, or make forecasts for systems ranging from power systems and robotics to biological systems and many more. The dynamics of these systems can often be described by ordinary differential equations (ODEs) that describe the evolution of the system state over time in the form

$$\dot{x}(t) = f(x(t), u(t), t) \quad (1)$$

where  $x(t)$  is the state vector,  $u(t)$  is the control input, and  $f$  is a function that describes the system dynamics. Trajectories of the system are then obtained by integrating the ODE over time with

$$x(t) = x(t_0) + \int_{t_0}^t f(x(t), u(t)) dt \quad (2)$$

In 2018, so called neural ordinary differential equation (NODE) were introduced which can model system dynamics by replicating the continuous dynamics  $f$  with a neural network (NN). This is achieved by coupling the NN with a numerical solver during training. Typical applications were often standard machine learning (ML) tasks like image classification and generative models like continuous normalizing flows. Since then, many extensions have

been proposed. One of them, so called augmented neural ordinary differential equations (A-NODEs) elevate the dynamics into a higher state space compared to dimensionality of the data used for training.

Unfortunately, the application to input-output system identification is still limited. One problem is, that typically not all states of a system can be measured (even though the system may be fully observable). This is a problem for NODEs as they require the full state vector to be used as a initial condition. Here, A-NODEs can offer a solution by taking a partly measured state and elevating it into the original dimensionality of the system. Unfortunately, this typically violates the markov property of the system, as the actual state trajectory depends on the full initial state vector and not only on the partial measurement of the current state. Assuming the system is observable, past measurements may be used to reconstruct the full state vector and thereby mitigate the effect.

This study therefore aims to investigate the potential of A-NODEs for input-output system identification using different augmentation strategies. In that, the system(s) can be chosen freely (as long as they can be described in the form of Eq.(1) but should contain one power system that we provide.

### *Research Objectives*

The objective of this research is to compare different augmentation strategies for A-NODEs in the context of input-output system identification:

1. (*"Static"*) *zero augmentation*: Extends the measured initial state vector by zeros to match the target dimensionality.
2. (*"Static"*) *augmentation with a NN* : Extends or entirely replaces the measured initial state-vector by feeding it to a NN, typically a multi-layer perceptron (MLP).
3. (*"Dynamic"*) *augmentation with a NN*: Constructs an initial state-vector from past measured data  $t_n-t_0$ . Here, recurrent or convolutional structures may be used for augmentation.

From here, we can formulate the following key research questions:

- How does the augmentation result in a violation of the markov property and what are the consequences the identification based on the accuracy of trajectory predictions?
- How can we mitigate this effect a much as possible by exploiting historic data in the augmentation strategy?
- How well do the strategies perform over different systems and different noise levels?
- How many samples are required with respect to the systems order and time constants to achieve accurate results?

## *Methodology*

### *Literature Research*

A thorough review of NODEs. Key areas of investigation include:

- fundamentals of NODEs, covering architecture and training (a good overview is given in [5], the original paper is [1]),
- concept of A-NODEs and different augmentation strategies ([2], [7], [5], [4] and others),
- concept of data-controlled/parameterized NNs ([7], [6], [9], [3]),
- application of NODEs in power systems ([9], [8]).

If not familiar already, you should also read up on

- fundamentals of ODEs and their numerical solution and the markov property,
- fundamentals of machine learning: training of NNs, concept of backpropagation.

### *Dataset Generation, Modeling*

In order to train models, we need data. There are two possibilities:

1. Model and simulate the system of interest.
2. Make use of open source benchmark datasets:

### *System Identification with NODEs*

In a first step, we assume ideal conditions, that is we have access to the full state vector for training.<sup>2</sup>

- Train on residuals of predictions with a already available implementation for NODEs.
- Analyze accuracy of system identification under different hyperparameter settings for all systems.

<sup>2</sup> Note that the required implementation for NODEs and a ML pipeline for hyperparameter optimization is available. This can be more or less done immediately after data is available without further struggle.

### *System Identification with A-NODEs - "static" augmentation*

In a second step, we will use the A-NODEs to identify the systems. Therefore, we will assume not all state are available in training.

- Extend the current framework with augmentation strategies for A-NODEs.
- Repeat the training with different augmentation strategies.
- Analyze the accuracy of system identification under different hyperparameter settings for all systems.

### *System Identification with A-NODEs - "dynamic" augmentation*

In a last step, we will develop dynamic augmentation strategies to mitigate the effect of violation of the markov property.

- Develop and implement an augmentation strategies for A-NODEs based on historic measurements.
- Repeat the training as before.
- Analyze the accuracy of system identification under different hyperparameter settings for all systems.

### *Expected Contributions*

- Development of a novel augmentation strategy for NODEs.
- Comparison of this compared to state of the art augmentation strategies.
- Analyze the potential for identifying different systems.

### *Tools and Implementation*

Everything will be implemented in Python.

- ML, NNs: pytorch
- NODEs, A-NODEs: torchdiffeq (used in our framework for NODEs)

### *Requirements and benefits*

#### Requirements

- *Programming Languages:* Python (proficient required)
- *Version Control:* Git (desired)
- *Machine Learning Frameworks:* differentiable libraries, such as pytorch (desired)
- *Theoretical Background:* Dynamical systems and ordinary differential equations (required) and machine learning (desired)
- *Grades:* Above average grades in relevant courses (desired)

#### Benefits:

- *Skills:* Machine learning, dynamical systems, system identification
- *Tools:* Experience with state-of-the-art machine learning libraries
- *Research:* Contribution to a novel research area that offers potential for publication, if successful (always good for your CV)

*References*

- [1] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. 2018. doi: 10.48550/ARXIV.1806.07366. URL <https://arxiv.org/abs/1806.07366>. Publisher: arXiv Version Number: 5.
- [2] E. Dupont, A. Doucet, and Y. W. Teh. Augmented Neural ODEs, Oct. 2019. URL <http://arxiv.org/abs/1904.01681>. arXiv:1904.01681 [stat].
- [3] M. Forgiione and D. Piga. Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59:69–81, 2021. ISSN 0947-3580. doi: <https://doi.org/10.1016/j.ejcon.2021.01.008>. URL <https://www.sciencedirect.com/science/article/pii/S0947358021000169>.
- [4] M. Forgiione, M. Mejari, and D. Piga. Learning neural state-space models: do we need a state estimator?, 2022. URL <https://arxiv.org/abs/2206.12928>.
- [5] P. Kidger. On Neural Differential Equations, 2022. URL <https://arxiv.org/abs/2202.02435>. Version Number: 1.
- [6] K. Lee and E. J. Parish. Parameterized neural ordinary differential equations: applications to computational physics problems. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2253):20210162, Sept. 2021. ISSN 1364-5021, 1471-2946. doi: 10.1098/rspa.2021.0162. URL <https://royalsocietypublishing.org/doi/10.1098/rspa.2021.0162>.
- [7] S. Massaroli, M. Poli, J. Park, A. Yamashita, and H. Asama. Dissecting Neural ODEs, 2020. URL <https://arxiv.org/abs/2002.08071>. Version Number: 4.
- [8] A. Rahman, J. Drgoňa, A. Tuor, and J. Strube. Neural Ordinary Differential Equations for Nonlinear System Identification, Mar. 2022. URL <http://arxiv.org/abs/2203.00120>. arXiv:2203.00120 [cs, eess].
- [9] H. M. H. Wolf and C. A. Hans. Identification of Power Systems with Droop-Controlled Units Using Neural Ordinary Differential Equations, 2024. URL <https://arxiv.org/abs/2411.08678>. Version Number: 1.