

# *Learning Governing Equations of Power Systems using Deep Symbolic Regression*

Hannes Wolf<sup>†</sup>

Version 53de59a

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

## *Abbreviations*

**SINDy** sparse identification of nonlinear dynamics

**DSR** deep symbolic regression

## *Background and Motivation*

Modern power grids are undergoing a transformation driven by the integration of renewable energy sources. These changes introduce increasingly complex dynamics that may not be fully captured by traditional first-principles models. The governing equations of these systems are often nonlinear, high-dimensional. Capturing these dynamics effectively is crucial for ensuring robust control and stability. Data-driven approaches offer a promising alternative to traditional physics-based models by learning governing equations directly from system measurements, thus providing insights beyond classical modeling assumptions.

sparse identification of nonlinear dynamics (SINDy) is a state-of-the-art technique for discovering governing equations from data. It leverages predefined candidate functions and numerical differentiation to infer system dynamics in a sparse representation, yielding an data-driven interpretable system model. However, SINDy faces one significant limitation. For SINDy to work best in practice, the user must have a good understanding of the system's dynamics and choose a small number of appropriate basis functions. Already a slightly larger set of basis functions can lead to a combinatorial explosion of possible terms, which makes the method computationally infeasible.

These limitations motivate the use of deep symbolic regression (DSR), which, unlike SINDy, does not rely on a predefined set of candidate functions and instead autonomously constructs the governing equations. This strongly weakens the necessary assumptions about the system and shifts the regression towards a black-box approach.

Furthermore, due to the inherent limitations of numerical differentiation, SINDy is sensitive to noise and may struggle with large sampling intervals where differentiation errors become significant. Therefore, both approaches may be combined with integration-based optimization, avoiding numerical differentiation entirely.

For some first reads, see [1–5].

## *Research Objectives*

The objective of this research is to compare three approaches for discovering governing equations in power systems:

1. *Sparse Identification of Nonlinear Dynamical Systems (SINDy)*: Uses predefined candidate functions and numerical differentiation to infer state derivatives.
2. *Deep Symbolic Regression (DSR)*: Discovers governing equations without predefined basis functions but still relies on numerical differentiation.
3. *Integration-Optimized Symbolic Regression*: Uses a differentiable solver to predict state evolution without numerical differentiation, optimizing symbolic equations via backpropagation through the solver.

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

- Can deep symbolic regression recover governing dynamics without predefined function libraries?
- Does integration-optimized symbolic regression improve stability and accuracy compared to differentiation-based methods when data has a small temporal resolution?
- How do these methods compare in terms prediction accuracy and computational cost?

## *Methodology*

### *Literature Research*

A thorough review of the state-of-the-art advancements in symbolic regression, deep symbolic regression, and their integration with numerical solvers will be conducted. Key areas of investigation include:

- Symbolic regression techniques such as SINDy, genetic programming and others.
- Advances in deep symbolic regression
- Methods that integrate symbolic regression with numerical solvers
- Applications of these methods in control systems, system identification, and power system modeling.

### *Dataset Generation (Power System Simulation)*

- Simulate small power grids with droop-controlled grid-forming inverters using Python
- Collect state and control input trajectories (voltage, frequency, active/reactive power and setpoints)

For this, there is already a simulation environment available that can be used to generate the data.

*Sparse Identification of Nonlinear Dynamical Systems (SINDy)*

- Define different function libraries based on assumptions about the system dynamics. Here, we start from some exact knowledge, preconstructing concise terms occurring in the dynamics. We then move away towards assumptions about the dynamics, considering we do not know the exact dynamics (e.g., polynomials, trigonometric functions).
- Solve for sparse coefficients using regularized least squares regression with different optimizers.

Some work has been done here, you are not starting from scratch.

*Deep Symbolic Regression (DSR)*

- Research SR/DSR frameworks in python
- Use genetic algorithms (e.g., PySR) to evolve symbolic expressions.
- Optimize for accuracy and complexity trade-offs.

*Integration-Optimized Symbolic Regression*

- Solve the DSR problem by leveraging numerical integration instead of finite differences.
- Use a solver (e.g., Runge-Kutta, implicit methods) to integrate forward:

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x, u) dt \quad (1)$$

- Compute loss based on  $x(t + \Delta t)$  instead of numerical derivatives.
- Evolve and optimize symbolic equations via gradient-based learning.

*Evaluation and Comparison*

The three methods will be compared based on:

- Interpretability
- Prediction accuracy
- Computational cost
- Data requirements (sampling rate, amount of required data, noise, etc.)

*Expected Contributions*

- Development of a novel integration-optimized symbolic regression framework for power system modeling.

- Comparison of classical (SINDy), deep learning (DSR), and solver-optimized approaches.
- Potential applications in power systems identification.

### *Tools and Implementation*

- SINDy: `pysindy` (Sparse Identification of Nonlinear Dynamics)
- Deep Symbolic Regression: PySR or others
- Integration-based Optimization: `torchdiffeq` (Differentiable solvers in PyTorch) or others

### *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), optimization (desired)
- *Grades*: Above average grades in relevant courses (desired)

#### Benefits:

- *Skills*: Machine learning, optimization, dynamical systems, power systems
- *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] D. Angelis, F. Sofos, and T. E. Karakasidis. Artificial Intelligence in Physical Sciences: Symbolic Regression Trends and Perspectives. *Archives of Computational Methods in Engineering*, 30(6):3845–3865, July 2023. ISSN 1134-3060, 1886-1784. doi: 10.1007/s11831-023-09922-z. URL <https://link.springer.com/10.1007/s11831-023-09922-z>.
- [2] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, Apr. 2016. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1517384113. URL <http://arxiv.org/abs/1509.03580>. arXiv:1509.03580 [math].

- [3] P. Kidger. On Neural Differential Equations, 2022. URL <https://arxiv.org/abs/2202.02435>. Version Number: 1.
- [4] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients, Apr. 2021. URL <http://arxiv.org/abs/1912.04871>. arXiv:1912.04871 [cs].
- [5] Y. A. Radwan, G. Kronberger, and S. Winkler. A Comparison of Recent Algorithms for Symbolic Regression to Genetic Programming, June 2024. URL <http://arxiv.org/abs/2406.03585>. arXiv:2406.03585 [cs].