





pyCLINE: A Python package using the CLINE method for discovery of nullcline structures in oscillatory dynamics

Bartosz Prokop ¹, Nikit Frolov ¹, and Lendert Gelens ¹

¹ Laboratory of Dynamics in Biological Systems, Department of Cellular and Molecular Medicine, KU Leuven  Corresponding author

DOI: [10.21105/joss.08112](https://doi.org/10.21105/joss.08112)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Mark A. Jensen](#)  

Reviewers:

- [@kumiori](#)
- [@majensen](#)

Submitted: 23 March 2025

Published: 11 September 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

Dynamical processes in physics, biology, chemistry, and engineering—such as planetary motion, climate variability, and cell cycle oscillations—are crucial to understanding complex systems. Traditionally, mathematical models describing these systems rely on differential equations derived from empirical data using established modeling principles and scientific intuition. However, the increasing availability of high-dimensional, complex datasets has rendered classical model derivation increasingly challenging.

As a result, data-driven or machine learning methods emerged that are able to handle high-dimensional data sets. However existing methods either are limited by data quality or the interpretability of their results. Thus we developed the **CLINE** (Computational Learning and Inference of NullclinEs) ([Prokop et al., 2025](#)) and introduce its Python implementation pyCLINE that allows users to extract and identify static phase-space structures without prior knowledge directly from time series data.

Statement of need

Machine learning and data-driven approaches have revolutionized the study of dynamical systems. Two primary methodologies exist:

- **Black-box methods** (e.g., neural networks) approximate system behavior but lack interpretability regarding underlying mechanisms.
- **White-box methods** derive symbolic differential equations directly from data but require high-quality datasets to ensure accuracy ([Prokop & Gelens, 2024](#)).

To bridge this gap, **grey-box methods** integrate the strengths of both approaches, handling large, structured datasets while preserving interpretability. Examples include Physics-Informed Neural Networks (PINNs) ([Karniadakis et al., 2021](#)), Biology-Informed Neural Networks (BINNs) ([Lagergren et al., 2020](#)), and Universal Differential Equations ([Rackauckas et al., 2020](#)). However, most of these methods focus on forecasting rather than extracting fundamental structural properties of dynamical systems.

To address this limitation, our method **CLINE** is able to extract static phase-space features, specifically the structure of nullclines, from time series data without forecasting.

Understanding nullcline structure of a dynamical system provides several key benefits ([Prokop et al., 2024](#)):

- **Comprehensive System Characterization:** Nullclines fully describe the system's steady-state behavior and provide richer insights than time series data alone.

- **Reduced Complexity for Symbolic Model Identification:** Once nullcline structures are identified, symbolic equations can be inferred using sparse regression techniques, such as sparse identification of nonlinear dynamics (SINDy) (Brunton et al., 2016) or symbolic regression (SR) (Schmidt & Lipson, 2009), with significantly lower computational complexity compared to direct time-series-based approaches.
- **Bias Reduction through Model-Free Inference:** Unlike traditional white-box methods, CLINE does not rely on predefined candidate terms (e.g., library-based functions), minimizing biases in model formulation and increasing adaptability to diverse systems.

pyCLINE is a Python package that allows one to easily set up and use the CLINE method as explained and shown in Prokop et al. (2025). It is based on the Python Torch implementation pyTorch (Paszke et al., 2019) and enables rapid identification of nullcline structures from simulated or measured time series data. The implementation of pyCLINE can generate example data sets from scratch, correctly prepare data for training and set up the feed forward neural network for training.

pyCLINE was designed to be used by researchers experienced with the use of machine learning or nonspecialists that are interested in applying the method to either different models or measured data. This allows for simple and fast implementation in many fields that are interested in discovering nullcline structures from measured data, that can help develop novel or confirm existing models of dynamical (oscillatory) systems.

Methodology

The main aspects of the CLINE method are explained in Prokop et al. (2025), nevertheless we provide a brief explanation of the method. In order to identify nullclines for a set of ordinary differential equations (ODEs) with system variables u and v , we have to set the derivative to 0:

$$\begin{aligned}u_t = f(u, v) &\rightarrow u_t = f(u, v) = 0 \\v_t = g(u, v) &\rightarrow v_t = g(u, v) = 0\end{aligned}$$

The functions of f and g are not known *a priori*. However, to learn the functions we can reformulate the nullcline equations to:

$$\begin{aligned}u &= f^{-1}(v, u_t) \text{ or } v = f^{-1}(u, u_t) \\u &= g^{-1}(v, v_t) \text{ or } v = g^{-1}(u, v_t)\end{aligned}$$

Now we have to learn the inverse functions f^{-1} and g^{-1} which describe the relationship between the measured variables u and v with additional derivative information u_t or v_t . As such, the target functions can be expressed as a feed-forward neural network with e.g. inputs u and u_t , to learn v .

After training, we can provide a set of u together with $u_t = 0$ (requirement for a nullcline) as inputs and learn the corresponding values of v that describe $u_t = f(u, v) = 0$. As a result, we learn the structure of a nullcline in the phase space u, v , to which other white-box methods can be applied to learn the symbolic equations, yet on a decisively simpler optimization problem than that on time series data.

Usage

The pyCLINE package can be downloaded and installed using pip:

```
pip install pyCLINE
```

The pyCLINE package includes three main modules (see Figure 1):

- `pyCLINE.generate_data()`: This module generates data which has been used in Prokop et al. (2025) along with many additional models that can be found under `pyCLINE.model()`.
- `pyCLINE.recovery_methods.data_preparation()`: Splits and normalizes that data for training, with many more features for the user to change the data.
- `pyCLINE.recovery_methods.nn_training()`: The pyTorch implementation that sets up the model and trains it.

The `pyCLINE.model()` currently includes a set of different models:

- FitzHugh-Nagumo model
- Bicubic model
- Gene expression model
- Glycolytic oscillation model
- Goodwin model
- Oregonator model
- Lorenz system
- Roessler system
- Delay oscillator (self-inhibitory gene)

Some of the models are three-dimensional and can be used to further study the limitations of the method, when applied to higher dimensional systems.

To demonstrate the method, we also provide `pyCLINE.example()` which contains full examples of how `pyCLINE` can be used. Here, `pyCLINE.example()` can be used to generate prediction data for four systems: The FitzHugh-Nagumo model with time scale separation variable $\varepsilon = 0.3$ (FHN), the Bicubic model (Bicubic), gene expression model (GeneExpression) and the delay oscillator model (DelayOscillator).

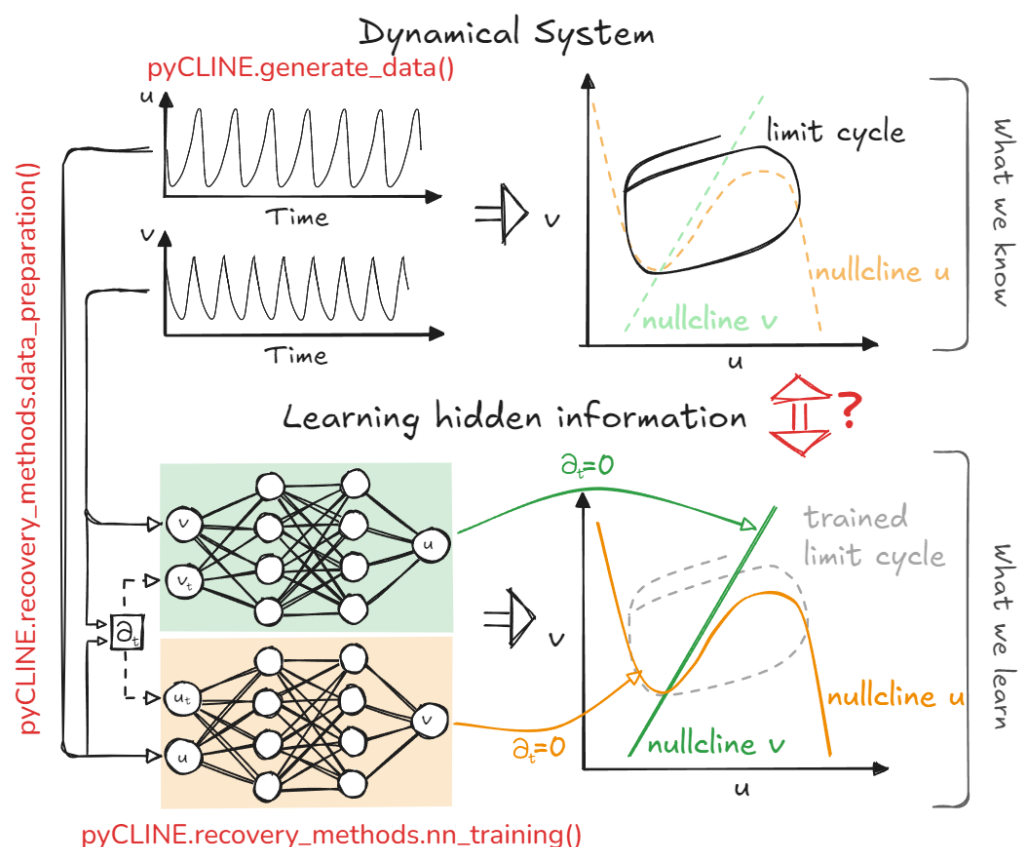


Figure 1: The method CLINE explained by using Figure 1 from Prokop et al. (2025). In red the main modules of the pyCLINE package are shown.

References

- Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15), 3932–3937. <https://doi.org/10.1073/pnas.1517384113>
- Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6), 422–440. <https://doi.org/10.1038/s42254-021-00314-5>
- Lagergren, J. H., Nardini, J. T., Baker, R. E., Simpson, M. J., & Flores, K. B. (2020). Biologically-informed neural networks guide mechanistic modeling from sparse experimental data. *PLOS Computational Biology*, 16(12), e1008462. <https://doi.org/10.1371/journal.pcbi.1008462>
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*, 8026–8037. <https://doi.org/10.48550/arXiv.1912.01703>
- Prokop, B., Billen, J., Frolov, N., & Gelens, L. (2025). Machine learning identifies nullclines in oscillatory dynamical systems. *Preprint at ArXiv*. <https://doi.org/10.48550/arXiv.2503.16240>
- Prokop, B., Frolov, N., & Gelens, L. (2024). Enhancing model identification with SINDy via

- nullcline reconstruction. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 34(6), 063135. <https://doi.org/10.1063/5.0199311>
- Prokop, B., & Gelens, L. (2024). From biological data to oscillator models using SINDy. *iScience*, 27(4), 109316. <https://doi.org/10.1016/j.isci.2024.109316>
- Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., Ramadhan, A., & Edelman, A. (2020). Universal Differential Equations for Scientific Machine Learning. *Preprint at ArXiv*. <https://doi.org/10.48550/arXiv.2001.04385>
- Schmidt, M., & Lipson, H. (2009). Distilling Free-Form Natural Laws from Experimental Data. *Science*, 324(5923), 81–85. <https://doi.org/10.1126/science.1165893>