The Journal of Open Source Software

kooplearn: A scikit-learn compatible library of algorithms for evolution operator learning

Giacomo Turri [1], **Grégoire Pacreau** [2], **Giacomo Meanti** [3], **Timothée Devergne** [4,1], **Daniel Ordoñez-Apraez** [1], **Erfan Mirzaei** [1], **Bruno Belucci** [5], **Karim Lounici** [2], **Vladimir R. Kostic** [1,6], **Massimiliano Pontil** [1,7], **and Pietro Novelli** [1]

1 Computational Statistics and Machine Learning, Italian Institute of Technology, Genoa, Italy **2** Centre de Mathématiques Appliquées, École Polytechnique, Palaiseau, France **3** Centre Inria de l'Université Grenoble Alpes, Montbonnot, France **4** Atomistic Simulations, Italian Institute of Technology, Genoa, Italy **5** Paris Dauphine University, Paris, France **6** Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Novi Sad, Serbia **7** AI Centre, Department of Computer Science, University College London, London, UK

DOI: 10.21105/joss.10342

Software

- Review
- Repository
- Archive

Editor: Johan Larsson

Reviewers:

- @sdahdah
- @gmarupilla
- @sepandhaghighi

Submitted: 13 January 2026
Published: 25 June 2026

Summary

kooplearn is a machine learning library that implements linear, kernel, and deep learning estimators of *dynamical operators* and their spectral decompositions. kooplearn can model both discrete-time evolution operators (Koopman/transfer) and continuous-time infinitesimal generators. By learning these operators, users can analyze dynamical systems via spectral methods, derive data-driven reduced-order models, and forecast future states and observables. kooplearn's interface is compliant with the `scikit-learn` API (Pedregosa et al., 2011), facilitating its integration into existing machine learning and data science workflows. Additionally, kooplearn includes curated benchmark datasets to support experimentation, reproducibility, and the fair comparison of learning algorithms. The software is available at https://github.com/Machine-Learning-Dynamical-Systems/kooplearn.

Statement of Need

From fluid flows down to atomistic motions, dynamical systems permeate every scientific discipline. Among the data-driven frameworks for modeling dynamical systems, evolution operator learning (Kostic et al., 2022) is both general and principled, and is especially well suited for interpretability (Mezić, 2005; Schütte et al., 2001) and dimensionality reduction (Klus et al., 2018). An evolution operator E characterizes dynamical systems, either stochastic $x_{t+1} \sim p(\cdot|x_t)$, or deterministic $x_{t+1} \sim \delta(\cdot - F(x_t))$, as follows: for every function f of the state of the system, $(\mathsf{E}f)(x_t)$ is the expected value of f one step ahead in the future, given that at time t the system was found in x_t

$$(\mathsf{E}f)(x_t) = \int p(dy|x_t) f(y) = \mathbb{E}_{y \sim X_{t+1}|X_t}[f(y)|x_t].$$

Notice that E is an operator because it maps any function f to another function, $x_t \mapsto (\mathsf{E}f)(x_t)$, and is *linear* because $\mathsf{E}(f + \alpha g) = \mathsf{E}f + \alpha \mathsf{E}g$. When the dynamics are deterministic, E is known as the *Koopman operator* (Koopman, 1931), while in the stochastic case it is known as the *transfer operator* (Applebaum, 2009). Arguably, the most important feature of evolution operators is their spectral decomposition (Mezić, 2005), which can be used to express the dynamics as a linear superposition of *modes*. These ideas lie at the core of the celebrated

Time-lagged Independent Component Analysis (Molgedey & Schuster, 1994), and Dynamic Mode Decomposition (DMD) (Kutz et al., 2016; Schmid, 2010).

Evolution operator learning is best understood from the perspective of *latent linear dynamical models*, as schematically depicted in Figure 1. In this framework, the dynamical state x_t is first mapped into a latent space defined by a (fixed or learned) representation φ. Then, a *linear evolution* map E is learned to approximate the dynamics of the latents. The pair (φ, E) provides an approximation of E restricted to the d-dimensional subspace spanned by the components of φ, given the data. `kooplearn` implements methods to learn φ, E, and the associated spectral decomposition of E.

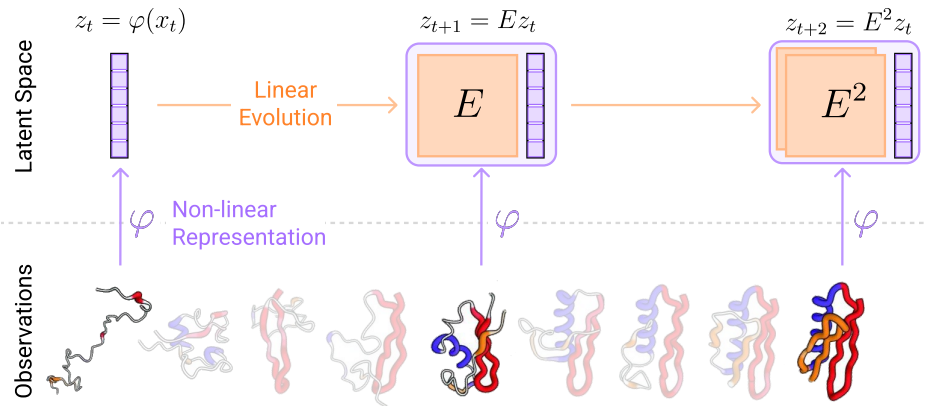

Figure 1: Sketch of the action of an evolution operator on a protein folding trajectory. The dynamics of the protein are linearized by means of a nonlinear representation φ and subsequently evolved by means of the linear map E.

State of the Field

The ecosystem of Python libraries that support operator-based modeling has grown considerably in recent years, with a predominant focus on the DMD family of methods. `PyDMD` (Ichinaga et al., 2024) emphasizes classical and kernel DMD variants; `pykoopman` (Pan et al., 2024) implements classical DMD methods with dictionary-based feature maps; `pykoop` (Dahdah & Forbes, 2025) offers a modular framework for lifting-function construction with a focus on system identification and control; `DLKoopman` (Dey & Davis, 2023) focuses on autoencoder approaches, while `KoopmanLab` (Xiong et al., 2023) targets Koopman neural operators. `kooplearn` addresses the general problem of learning evolution operators, and it is the result of a multi-year research effort in innovative operator learning algorithms. While it provides standard prediction and spectral decomposition utilities, it extends the state of the art in evolution operator learning codes by implementing fast kernel estimators (Meanti et al., 2023; Turri et al., 2026), infinitesimal generator models for SDEs (Devergne et al., 2024; Kostic, Lounici, Halconruy, et al., 2024), and specialized losses for deep representation learning (Kostic, Novelli, et al., 2024; Kostic, Lounici, Pacreau, et al., 2024; Mardt et al., 2018; Turri et al., 2025). We now provide a concise overview of the functionality of `kooplearn`.

Learning Linear Evolution Maps E

`kooplearn` implements algorithms for learning evolution operators when the representation φ is fixed. The library offers estimators in both their linear and kernel formulations (see the `Ridge` and `KernelRidge` classes), which bridge the gap between recent theoretical advances (Kostic et al., 2022, 2023; Kostic, Lounici, Inzerilli, et al., 2024; Kostic, Novelli, et al., 2024) and practical code implementations. A key model in `kooplearn` is the kernel-based *Reduced Rank Regression* (Kostic et al., 2022). This estimator provably outperforms traditional methods (Williams et

al., 2015) in approximating the operator's spectrum (Kostic et al., 2023), as illustrated in Figure 2. To our knowledge, `kooplearn` provides the only open-source implementation of this algorithm. To handle large datasets, `kooplearn` also includes randomized (Turri et al., 2026) and Nyström-based (Meanti et al., 2023) kernel estimators, which significantly speed up the fitting process, making it one of the fastest libraries for kernel-based operator learning, as shown in Figure 3.

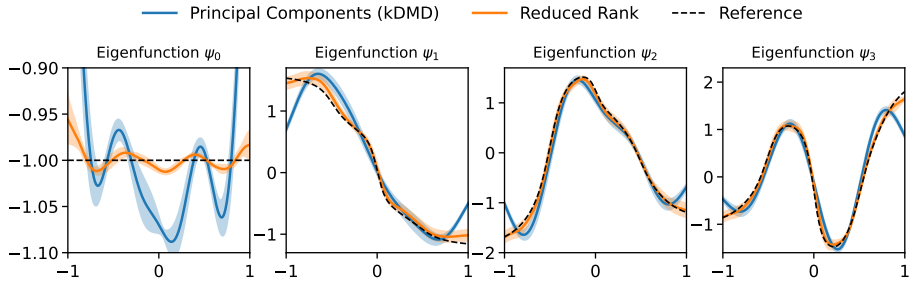

Figure 2: Comparison between kernel DMD (kDMD) and Reduced Rank estimators. The Reduced Rank estimator provides a more accurate approximation of the leading eigenfunctions of the transfer operator for the overdamped Langevin dynamics.

Figure 3: Fit time of a kernel model (Gaussian kernel) on a dataset of 5000 observations from the Lorenz-63 dynamical system. The results are the median of three independent runs on a system equipped with an Intel Core i9-9900X CPU (3.50 GHz) and 48 GB of RAM.

Learning the Representation φ

`kooplearn` also exposes theoretically-grounded loss functions — implemented in both PyTorch (Paszke et al., 2019) and JAX (Bradbury et al., 2018) — suited for learning the representation φ with neural network models. This allows the incorporation of structural priors, such as graph-based encoders. Within this deep learning approach, two main families are supported: (i) *encoder-decoder* schemes with the loss proposed in Lusch et al. (2018), and (ii) *encoder-only* schemes, for which `kooplearn` implements the VAMP loss (Mardt et al., 2018) and the spectral contrastive loss (Turri et al., 2025).

Learning the Infinitesimal Generator of Diffusion Processes

In continuous-time dynamics, the system's evolution operator can be expressed as the exponential of the *infinitesimal generator* L, a differential operator defined by the equations of motion (Applebaum, 2009, Chapter 3). Formally, for time-homogeneous dynamics, the generator relates to the evolution operator via $\mathrm{E} = e^{\mathrm{L}}$, and consequently $\mathbb{E}[f(X_t)|x_0] = (e^{t\mathrm{L}} f)(x_0)$. Since the exponential of an operator preserves its eigenfunctions, one can use knowledge of L (or its properties) to learn dynamical behavior without requiring lag-time data. In other words, it becomes possible to construct a physics-informed kinetic model E solely from static (equilibrium) data. To this end, `kooplearn` provides implementations of recent kernel-based algorithms for diffusion processes with Dirichlet boundary conditions from Kostic,

Lounici, Halconruy, et al. (2024), as well as neural representations as proposed by Devergne et al. (2024). As demonstrated by Devergne et al. (2025), these approaches improve sample complexity compared to estimators that rely solely on lag-time trajectory data.

Datasets

To foster reproducibility and rigorous benchmarking, `kooplearn` includes a `kooplearn.datasets` module, containing utilities to easily generate trajectories for systems that range from deterministic chaos (e.g., *Lorenz-63*, *Duffing oscillator*, *Logistic Map*) to stochastic and metastable dynamics (e.g., *stochastic linear systems*, *regime-switching models*, *Langevin dynamics*). A distinguishing feature of the library is the inclusion of benchmarks with accessible ground-truth spectral decompositions—such as the *Noisy Logistic Map* (Ostruszka et al., 2000) and *Overdamped Langevin Dynamics* in a quadruple-well potential (Prinz et al., 2011). These allow users to quantify the accuracy of learned eigenvalues and eigenfunctions directly (as demonstrated in Figure 2). Finally, the suite includes the *Ordered MNIST* from Kostic et al. (2022) to evaluate performance on high-dimensional structured data. Examples of trajectories generated using the `kooplearn.datasets` module are illustrated in Figure 4.

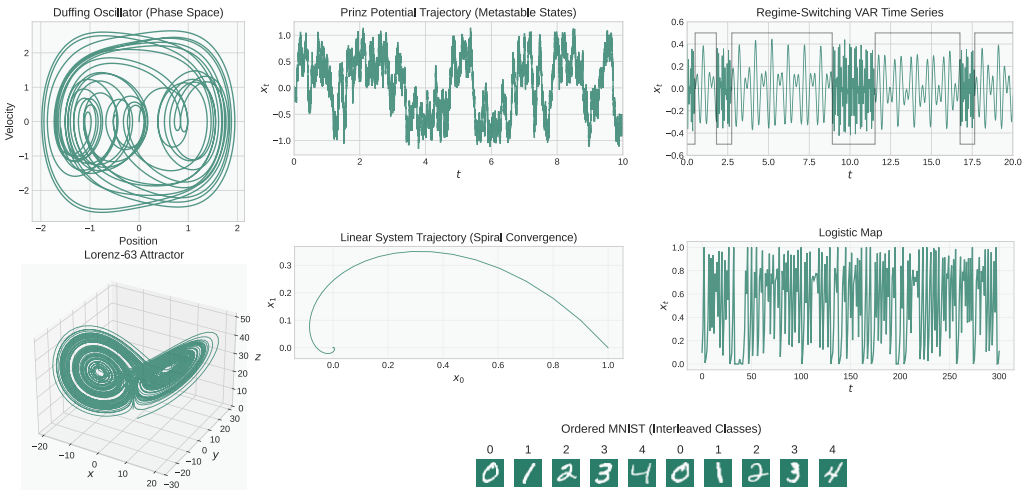

Figure 4: Samples from the datasets included in `kooplearn`.

Software Design

`kooplearn`'s design is founded on three core principles: (i) to provide a user-friendly, modular, object-oriented API, (ii) to embrace community standards by integrating with popular libraries like `scikit-learn` (Pedregosa et al., 2011), `PyTorch` (Paszke et al., 2019), and `JAX` (Bradbury et al., 2018), and (iii) to provide optimized implementations of state-of-the-art algorithms.

The library's architecture is modular, with subpackages for different modeling approaches: `kooplearn.linear_model`, `kooplearn.kernel`, and deep learning utilities in `kooplearn.torch` and `kooplearn.jax`. The estimators within the `linear_model` and `kernel` subpackages, such as `Ridge` and `KernelRidge`, inherit from `sklearn.base.BaseEstimator`, strictly adhering to the `scikit-learn` API. This design choice ensures that users familiar with `scikit-learn` can easily adopt `kooplearn` into their workflows, using methods like `fit`, `predict`, and `score`. To ensure high performance in learning the estimators, the library leverages optimized `scipy.linalg` and `scipy.sparse.linalg` (Virtanen et al., 2020) routines, wrapping standard libraries like LAPACK and ARPACK, and offers a flexible selection of solvers. Users can choose from `dense`, `arpack`, or `randomized` strategies to optimize computational efficiency according to the problem size.

For deep learning applications, `kooplearn` provides a suite of theoretically-grounded loss functions, such as the `VampLoss` and `SpectralContrastiveLoss`, which are implemented as `torch.nn.Module` objects. This allows seamless integration with custom neural network architectures in `PyTorch` or `JAX`, enabling users to leverage GPUs for accelerated training. This combination of a high-level, user-friendly API with low-level computational flexibility makes `kooplearn` a powerful tool for both rapid prototyping and advanced research.

Research Impact Statement

`kooplearn` provides a robust and accessible platform for the study of dynamical systems through the lens of operator theory. The package is the result of a collaborative effort between researchers at the Italian Institute of Technology and the École Polytechnique.

By providing state-of-the-art implementations of algorithms like Reduced Rank Regression (Kostic et al., 2022) and fast kernel methods (Meanti et al., 2023; Turri et al., 2026), `kooplearn` lowers the barrier to entry for complex techniques in operator learning, making it valuable for research, education, rapid prototyping, and exploratory analysis of dynamical systems. The package has been used in a number of publications at the intersection of machine learning and dynamical systems theory (Bevanda et al., 2023, 2025; Kostic et al., 2022, 2023; Kostic, Lounici, Inzerilli, et al., 2024; Kostic, Novelli, et al., 2024; Turri et al., 2025).

To further support the scientific community, `kooplearn` includes a `datasets` module with a collection of benchmark systems. These curated datasets, many with known ground-truth spectral properties, facilitate reproducible research and fair comparison between different algorithms. The library is available under the MIT license and can be installed via PyPI (`pip install kooplearn`). Its documentation, alongside many worked-out examples, is available on the webpage https://kooplearn.readthedocs.io/.

AI Usage Disclosure

OpenAI Codex and ChatGPT (GPT-5 models) were used only for minor auxiliary tasks during the preparation of the software repository and manuscript, including code refactoring and formatting, assistance with documentation and unit tests, and proofreading for typographical or grammatical errors. All scientific ideas, software design decisions, experiments, analyses, and interpretations were developed by the authors. The authors reviewed and verified all AI-assisted changes and text before submission.

Acknowledgements

This work is partially funded by the European Union - NextGenerationEU and by the Ministry of University and Research (MUR), National Recovery and Resilience Plan (NRRP), through the PNRR MUR Project PE000013 CUP J53C22003010006 "Future Artificial Intelligence Research (FAIR)" and EU Project ELIAS under grant agreement No. 101120237.

References

Applebaum, D. (2009). *Lévy processes and stochastic calculus*. Cambridge University Press. https://doi.org/10.1017/cbo9780511809781

Bevanda, P., Beier, M., Capone, A., Sosnowski, S. G., Hirche, S., & Lederer, A. (2025). Koopman-equivariant Gaussian processes. In Y. Li, S. Mandt, S. Agrawal, & E. Khan (Eds.), *Proceedings of the 28th international conference on artificial intelligence and statistics* (Vol. 258, pp. 3151–3159). PMLR. https://proceedings.mlr.press/v258/bevanda25a.html

Bevanda, P., Beier, M., Lederer, A., Sosnowski, S., Hüllermeier, E., & Hirche, S. (2023). Koopman kernel regression. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 16207–16221). Curran Associates, Inc. https://doi.org/10.52202/075280-0713

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/jax-ml/jax

Dahdah, S., & Forbes, J. R. (2025). Pykoop: A Python library for Koopman operator approximation. *Journal of Open Source Software*, *10*(114), 7947. https://doi.org/10.21105/joss.07947

Devergne, T., Kostic, V. R., Parrinello, M., & Pontil, M. (2024). From biased to unbiased dynamics: An infinitesimal generator approach. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (Vol. 37, pp. 75495–75521). Curran Associates, Inc. https://doi.org/10.52202/079017-2404

Devergne, T., Kostic, V. R., Pontil, M., & Parrinello, M. (2025). Slow dynamical modes from static averages. *The Journal of Chemical Physics*, *162*(12), 124108. https://doi.org/10.1063/5.0246248

Dey, S., & Davis, E. W. (2023). DLKoopman: A deep learning software package for Koopman theory. In N. Matni, M. Morari, & G. J. Pappas (Eds.), *Proceedings of the 5th annual learning for dynamics and control conference* (Vol. 211, pp. 1467–1479). PMLR. https://proceedings.mlr.press/v211/dey23a.html

Ichinaga, S. M., Andreuzzi, F., Demo, N., Tezzele, M., Lapo, K., Rozza, G., Brunton, S. L., & Kutz, J. N. (2024). PyDMD: A Python package for robust dynamic mode decomposition. *Journal of Machine Learning Research*, *25*(417), 1–9. http://jmlr.org/papers/v25/24-0739.html

Klus, S., Nüske, F., Koltai, P., Wu, H., Kevrekidis, I., Schütte, C., & Noé, F. (2018). Data-driven model reduction and transfer operator approximation. *Journal of Nonlinear Science*, *28*(3), 985–1010. https://doi.org/10.1007/s00332-017-9437-7

Koopman, B. O. (1931). Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, *17*(5), 315–318. https://doi.org/10.1073/pnas.17.5.315

Kostic, V. R., Lounici, K., Halconruy, H., Devergne, T., & Pontil, M. (2024). Learning the infinitesimal generator of stochastic diffusion processes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (Vol. 37, pp. 137806–137846). Curran Associates, Inc. https://doi.org/10.52202/079017-4377

Kostic, V. R., Lounici, K., Inzerilli, P., Novelli, P., & Pontil, M. (2024). Consistent long-term forecasting of ergodic dynamical systems. In R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, & F. Berkenkamp (Eds.), *Proceedings of the 41st international conference on machine learning* (Vol. 235, pp. 25370–25395). PMLR. https://proceedings.mlr.press/v235/kostic24a.html

Kostic, V. R., Lounici, K., Novelli, P., & Pontil, M. (2023). Sharp spectral rates for Koopman operator learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 32328–32339). Curran Associates, Inc. https://doi.org/10.52202/075280-1403

Kostic, V. R., Lounici, K., Pacreau, G., Turri, G., Novelli, P., & Pontil, M. (2024). Neural conditional probability for uncertainty quantification. In A. Globerson, L. Mackey, D.

Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (Vol. 37, pp. 60999–61039). Curran Associates, Inc. https://doi.org/10.52202/079017-1950

Kostic, V. R., Novelli, P., Grazzi, R., Lounici, K., & Pontil, M. (2024). Learning invariant representations of time-homogeneous stochastic dynamical systems. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, & Y. Sun (Eds.), *International conference on representation learning* (Vol. 2024, pp. 54329–54341). https://proceedings.iclr.cc/paper_files/paper/2024/file/eeab2e00835c71d64458ad1821e05664-Paper-Conference.pdf

Kostic, V. R., Novelli, P., Maurer, A., Ciliberto, C., Rosasco, L., & Pontil, M. (2022). Learning dynamical systems via Koopman operator regression in reproducing kernel Hilbert spaces. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35, pp. 4017–4031). Curran Associates, Inc. https://doi.org/10.52202/068431-0290

Kutz, J. N., Brunton, S. L., Brunton, B. W., & Proctor, J. L. (2016). *Dynamic mode decomposition*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9781611974508

Lusch, B., Kutz, J. N., & Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, *9*(1). https://doi.org/10.1038/s41467-018-07210-0

Mardt, A., Pasquali, L., Wu, H., & Noé, F. (2018). VAMPnets for deep learning of molecular kinetics. *Nature Communications*, *9*(1). https://doi.org/10.1038/s41467-017-02388-1

Meanti, G., Chatalic, A., Kostic, V. R., Novelli, P., Pontil, M., & Rosasco, L. (2023). Estimating Koopman operators with sketching to provably learn large scale dynamical systems. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 77242–77276). Curran Associates, Inc. https://doi.org/10.52202/075280-3378

Mezić, I. (2005). Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, *41*(1), 309–325. https://doi.org/10.1007/s11071-005-2824-x

Molgedey, L., & Schuster, H. G. (1994). Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, *72*(23), 3634–3637. https://doi.org/10.1103/physrevlett.72.3634

Ostruszka, A., Pakoński, P., Słomczyński, W., & Życzkowski, K. (2000). Dynamical entropy for systems with stochastic perturbation. *Phys. Rev. E*, *62*, 2018–2029. https://doi.org/10.1103/PhysRevE.62.2018

Pan, S., Kaiser, E., Silva, B. M. de, Kutz, J. N., & Brunton, S. L. (2024). PyKoopman: A Python package for data-driven approximation of the Koopman operator. *Journal of Open Source Software*, *9*(94), 5881. https://doi.org/10.21105/joss.05881

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, 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. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(85), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Prinz, J.-H., Wu, H., Sarich, M., Keller, B., Senne, M., Held, M., Chodera, J. D., Schütte, C., & Noé, F. (2011). Markov models of molecular kinetics: Generation and validation. *The Journal of Chemical Physics*, *134*(17). https://doi.org/10.1063/1.3565032

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, *656*, 5–28. https://doi.org/10.1017/S0022112010001217

Schütte, Ch., Huisinga, W., & Deuflhard, P. (2001). Transfer operator approach to conformational dynamics in biomolecular systems. In B. Fiedler (Ed.), *Ergodic theory, analysis, and efficient simulation of dynamical systems* (pp. 191–223). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-56589-2_9

Turri, G., Bonati, L., Zhu, K., Pontil, M., & Novelli, P. (2025). *Self-supervised evolution operator learning for high-dimensional dynamical systems*. https://arxiv.org/abs/2505.18671

Turri, G., Kostic, V. R., Novelli, P., & Pontil, M. (2026). A randomized algorithm to solve reduced rank operator regression. *SIAM Journal on Mathematics of Data Science*, *8*(1), 23–45. https://doi.org/10.1137/24M1632097

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Williams, M. O., Rowley, C. W., & Kevrekidis, I. G. (2015). A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, *2*(2), 247–265. https://doi.org/10.3934/jcd.2015005

Xiong, W., Ma, M., Huang, X., Zhang, Z., Sun, P., & Tian, Y. (2023). KoopmanLab: Machine learning for solving complex physics equations. *APL Machine Learning*, *1*(3), 036110. https://doi.org/10.1063/5.0157763