Submitted 5/24; Revised 10/24; Published 11/24

PyDMD: A Python Package for Robust Dynamic Mode Decomposition

Sara M. Ichinaga[1] SARAMI7@UW.EDU
Francesco Andreuzzi[2,4] ANDREUZZI.FRANCESCO@GMAIL.COM
Nicola Demo[2] NICOLA.DEMO@SISSA.IT
Marco Tezzele[3] MARCO.TEZZELE@EMORY.EDU
Karl Lapo[5] KARL-ERIC.LAPO@UIBK.AC.AT
Gianluigi Rozza[2] GIANLUIGI.ROZZA@SISSA.IT
Steven L. Brunton[6] SBRUNTON@UW.EDU
J. Nathan Kutz[1] KUTZ@UW.EDU
[1] *Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA*
[2] *Mathematics Area, mathLab, SISSA, via Bonomea 265, I-34136 Trieste, Italy*
[3] *Department of Mathematics, Emory University, Atlanta, GA 30322, USA*
[4] *CERN, Geneva, Switzerland*
[5] *Department of Atmospheric and Cryospheric Sciences, University of Innsbruck, Innsbruck, Austria*
[6] *Department of Mechanical Engineering, University of Washington, Seattle, WA 98195, USA*

Editor: Joaquin Vanschoren

Abstract

The *dynamic mode decomposition* (DMD) is a powerful data-driven modeling technique that reveals coherent spatiotemporal patterns from dynamical system snapshot observations. `PyDMD` is a Python package that implements DMD and several of its major optimizations and methodological extensions. In this paper, we introduce the version 1.0 release of `PyDMD`, which includes new data preprocessors, plotting tools, and a number of cutting-edge DMD methods specifically designed to handle real-world data that may be noisy, multi-scale, parameterized, prohibitively high-dimensional, and even strongly nonlinear. The package is friendly to install, thoroughly-documented, supplemented with extensive code examples, and modularly-structured to support future additions. The entire codebase is released under the MIT license and is available at `https://github.com/PyDMD/PyDMD`.

Keywords: dynamic mode decomposition, dynamical systems, open-source, Python

1. Introduction and Motivation

In recent years, the availability and abundance of high-fidelity data across the sciences has greatly motivated the utilization of, as well as the necessity for, algorithms that are accurate, efficient, intuitive, and purely data-driven. One algorithm that has emerged as a powerful method for analyzing dynamical system data is the *dynamic mode decomposition* (DMD) (Schmid, 2010; Tu et al., 2014; Kutz et al., 2016a; Schmid, 2022), which generally seeks a low-dimensional set of key spatiotemporal modes that describe a set of observations. This information allows for a variety of tasks, including dimensionality reduction, state reconstruction, future-state prediction, and system control. Hence despite its conception as a method for analyzing fluid flows (Schmid, 2010; Schmid and Sesterhenn, 2008), DMD has since been applied to data sets across multiple scientific disciplines, including epidemiology (Proctor and Eckhoff, 2015), neuroscience (Brunton et al., 2016), finance (Mann and Kutz, 2016), plasma physics (Taylor et al., 2018), video processing (Grosek and Kutz, 2014), and robotics (Berger et al., 2015). DMD has even become the standard approach for approximating the Koopman operator from data (Rowley et al., 2009; Brunton et al., 2022). It is thus imperative that DMD and its growing suite of innovations remain both intuitive and accessible for scientists and engineers with diverse backgrounds in mathematics.

`PyDMD` is a Python package that provides the tools necessary for executing the DMD pipeline within an abstract, user-friendly interface. Initially released in 2018, the original `PyDMD` package (Demo et al., 2018) (version 0.1) implemented the vast majority of DMD algorithms available at the time of release. We provide an exhaustive list of these methods in Table 1. However since `PyDMD`'s initial release, a multitude of DMD variants crucial for real-world data analysis have been developed. For example, optimized DMD (Askham and Kutz, 2018) and its stabilized variant bagging, optimized DMD (BOP-DMD) (Sashidhar and Kutz, 2022) have arisen, both of which frame the DMD objective as a nonlinear optimization problem that can be solved with variable projection. This approach not only optimally suppresses model bias due to measurement noise, but it also allows for snapshots that are unevenly sampled in time, as well as the application of model regularizers and constraints. The multiresolution coherent spatiotemporal scale separation (mrCOSTS) algorithm (Lapo et al., 2024) was also developed, which substantially improves upon multiresolution DMD (Kutz et al., 2016b) and is capable of decomposing real-world data from multi-scale systems. Parametric DMD for analyzing data parameterizations (Andreuzzi et al., 2023; Hess et al., 2023), randomized DMD for optimally compressing large data sets (Erichson et al., 2019b), and physics-informed DMD for enforcing model constraints and operator structure (Baddoo et al., 2023) have also arisen, hence motivating their incorporation into `PyDMD`.

In this version 1.0 release of `PyDMD`, we introduce users to state-of-the-art DMD methods, algorithms, and tools that are specifically geared towards real-world data modeling scenarios. In addition to implementing the aforementioned DMD algorithms, `PyDMD` version 1.0 introduces a variety of methods that are critical for analyzing highly nonlinear systems, including extended DMD (Williams et al., 2015a,b), the Hankel alternative view of Koopman (HAVOK) analysis (Brunton et al., 2017; Hirsh et al., 2021), and the linear and nonlinear disambiguation optimization (LANDO) (Baddoo et al., 2022). See Table 1 for a list of all available algorithms. We also introduce a suite of data preprocessing modules such as data centering (Hirsh et al., 2020) and time-delay (Arbabi and Mezić, 2017; Brunton et al., 2017), as well as new plotting features for quick and simple result viewing and professional figure generation. Our code is unit tested, regularly maintained, and completely open-source under the MIT license. All implementations are backed by standard Python libraries such as NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), Matplotlib (Hunter, 2007) and Scikit-learn (Pedregosa et al., 2011), and a user-friendly, Scikit-learn-style API (Buitinck et al., 2013) is utilized in order to standardize module output and support future extensions of the package. Thorough documentation `[link]` and an extensive suite of Jupyter Notebook tutorials `[link]` are readily available for new users and developers of the package. Through this update, `PyDMD` continues to be, to our knowledge, the only DMD toolkit that encompasses nearly every significant DMD variant in the literature.

Method Category	DMD Method	`PyDMD` `0.1`	`PyDMD` `1.0`
Original method	Exact DMD (Tu et al., 2014)	✓	✓
Noise-robust methods	Forward-backward DMD (Dawson et al., 2016)	✓	✓
	Total least-squares DMD (Hemati et al., 2017)	✓	✓
	Closed-form DMD (Héas and Herzet, 2022)	✓	✓
	Subspace DMD (Takeishi et al., 2017)	✓	✓
	Physics-informed DMD (Baddoo et al., 2023)	✗	✓
	Optimized DMD (Askham and Kutz, 2018)	✗	✓
	BOP-DMD (Sashidhar and Kutz, 2022)	✗	✓
Inputs and control	DMD with control (Proctor et al., 2016)	✓	✓
Methods for data compression and sparsity promotion	Sparsity-promoting DMD (Jovanović et al., 2014)	✓	✓
	Compressed DMD (Erichson et al., 2019a)	✓	✓
	Randomized DMD (Erichson et al., 2019b)	✗	✓
Latent variable methods for partial measurements	Hankel DMD (Arbabi and Mezić, 2017)	✓	✓
	Higher order DMD (Le Clainche and Vega, 2017)	✓	✓
	HAVOK (Brunton et al., 2017; Hirsh et al., 2021)	✗	✓
Methods for multi-scale data	Multiresolution DMD (Kutz et al., 2016b)	✓	✓
	Sliding mrDMD (Dylewsky et al., 2019)	✗	✓
	mrCOSTS (Lapo et al., 2024)	✗	✓
Methods for highly nonlinear systems	Extended DMD (Williams et al., 2015b,a)	✗	✓
	LANDO (Baddoo et al., 2022)	✗	✓
Parameterized data	Parametric DMD (Andreuzzi et al., 2023; Hess et al., 2023)	✗	✓

Table 1: DMD algorithms available across different `PyDMD` releases.

2. PyDMD Structure and Usage

Given snapshots $\mathbf{x}(t_k) \in \mathbb{C}^n$ collected at times $\{t_k\}_{k=1}^m$ and organized into the columns of $\mathbf{X} \in \mathbb{C}^{n\times m}$, the DMD algorithm generally seeks the rank-r spatiotemporal decomposition

$$\mathbf{X} \approx \begin{bmatrix} | & & | \\ \boldsymbol{\phi}_1 & \dots & \boldsymbol{\phi}_r \\ | & & | \end{bmatrix} \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_r \end{bmatrix} \begin{bmatrix} e^{\omega_1 t_1} & \dots & e^{\omega_1 t_m} \\ \vdots & \ddots & \vdots \\ e^{\omega_r t_1} & \dots & e^{\omega_r t_m} \end{bmatrix} = \boldsymbol{\Phi}\text{diag}(\mathbf{b})\mathbf{T}(\boldsymbol{\omega}),$$

with spatial modes $\boldsymbol{\phi}_j \in \mathbb{C}^n$, corresponding temporal frequencies $\omega_j \in \mathbb{C}$, and spatiotemporal mode amplitudes $b_j \in \mathbb{C}$. Variants of the algorithm arise when one alters the data that is decomposed or the approach that is used in order to compute $\boldsymbol{\Phi}, \boldsymbol{\omega}$, $\mathbf{b}$.

The `PyDMD` package is modular, with most DMD variants possessing their own module within the package. The `DMDBase` class establishes a consistent API across modules, as it implements a variety of basic functionalities and forms a foundation for the majority of DMD variants in `PyDMD`. In general, `PyDMD` modules can (1) accept and store parameters for their respective DMD variant, (2) compute and store spatiotemporal DMD components via the `fit` method, and (3) fetch and utilize the spatiotemporal diagnostics for tasks such as mode visualization and data reconstruction. We direct readers to the documentation `[link]` for exhaustive descriptions of the front-end API, as well as to our developer tutorial `[link]` for more information on the back-end API.

If we let `X` denote our data matrix, we can preprocess our data, perform DMD, and visualize the resulting spatiotemporal modes with the following code. This particular example

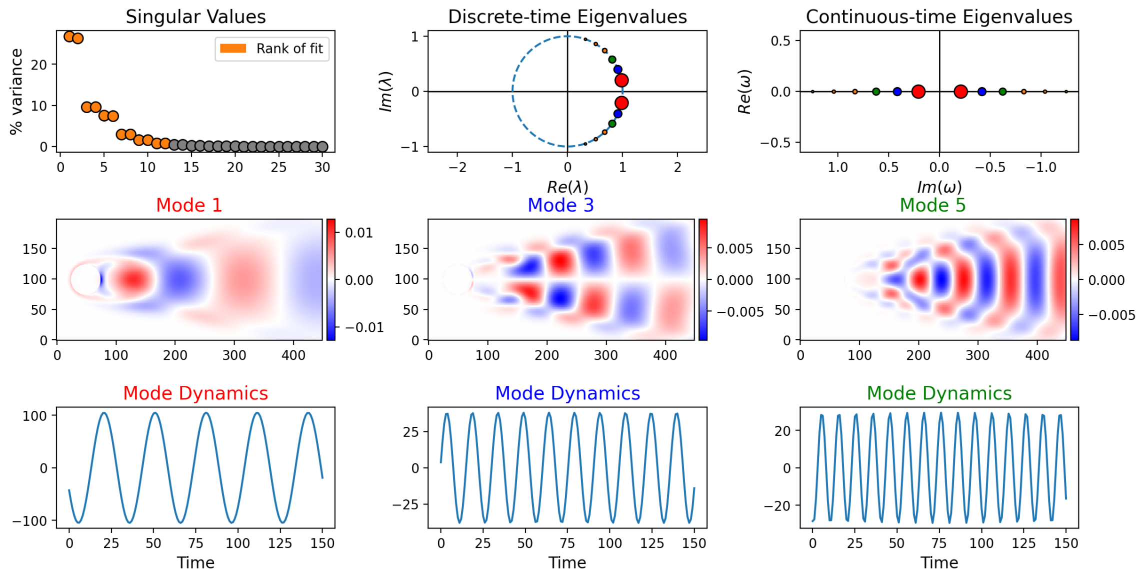

Figure 1: Sample **plot_summary** function output using fluid flow past a cylinder data with Reynolds number $Re = 100$. Data is available at dmdbook.com/DATA.zip.

demonstrates the use of exact DMD with data centering. In order to utilize alternative DMD algorithms and preprocessors, one must simply invoke the proper `PyDMD` modules.

```python
from pydmd import DMD
from pydmd.preprocessing import zero_mean_preprocessing
from pydmd.plotter import plot_summary

dmd = DMD(svd_rank=12) # Build DMD model.
dmd = zero_mean_preprocessing(dmd) # Wrap with data preprocessor.
dmd.fit(X) # Fit DMD model to snapshot data.
plot_summary(dmd) # Plot key spatiotemporal modes.
```

Once a `PyDMD` module is fitted, the **plot_summary** routine may then be used to visualize the components of $\mathbf{\Phi}, \boldsymbol{\omega}, \mathbf{b}$. This function accepts a variety of plotting parameters and can be used to generate figures like Figure 1. For more examples that highlight specific DMD variants and use-cases, see our complete set of Jupyter Notebook tutorials `[link]`.

3. Conclusion

The `PyDMD` package is an open-source project that enables users with diverse backgrounds in mathematics to apply DMD within a user-friendly Pythonic environment. Our latest updates featured in `PyDMD` version 1.0 additionally make it easier than ever for users to apply, and visualize results from, state-of-the-art DMD methods that are capable of extracting coherent spatiotemporal structures from real-world data sets. We hope that through this work and future works like this, `PyDMD` can continue to serve as a practical data analysis tool and as an ever-expanding centralized code base for DMD methods.

Acknowledgments

We wish to acknowledge the support of the National Science Foundation AI Institute in Dynamic Systems grant 2112085 (S.M.I, S.L.B. and J.N.K.). S.L.B. acknowledges support from The Boeing Company. This work has been partially supported by the consortium iNEST (Interconnected North-East Innovation Ecosystem), Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS00000043, supported by the European Union's NextGenerationEU program, and by European Union Funding for Research and Innovation—Horizon Europe Program—in the framework of European Research Council Executive Agency: ERC POC 2022 ARGOS project 101069319 "Advanced Reduced order modellinG: Online computational web server for complex parametric Systems" P.I. Professor Gianluigi Rozza. Additional funding was in part provided by the Austrian Science Fund (FWF) [10.55776/ESP214] (K.L.).

References

Francesco Andreuzzi, Nicola Demo, and Gianluigi Rozza. A dynamic mode decomposition extension for the forecasting of parametric dynamical systems. *SIAM Journal on Applied Dynamical Systems*, 22(3):2432–2458, 2023. doi: 10.1137/22M1481658.

Hassan Arbabi and Igor Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126, 2017. doi: 10.1137/17M1125236.

Travis Askham and J. Nathan Kutz. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 17(1):380–416, 2018. doi: 10.1137/M1124176.

Peter J. Baddoo, Benjamin Herrmann, Beverley J. McKeon, and Steven L. Brunton. Kernel learning for robust dynamic mode decomposition: linear and nonlinear disambiguation optimization. *Proceedings of the Royal Society A*, 478(2260):20210830, 2022. doi: 10.1098/rspa.2021.0830.

Peter J. Baddoo, Benjamin Herrmann, Beverley J. McKeon, J. Nathan Kutz, and Steven L. Brunton. Physics-informed dynamic mode decomposition. *Proceedings of the Royal Society A*, 479(2271):20220576, 2023. doi: 10.1098/rspa.2022.0576.

Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Advanced Robotics*, 29:331–343, 2015. doi: 10.1080/01691864.2014.981292.

Bingni W. Brunton, Lise A. Johnson, Jeffrey G. Ojemann, and J. Nathan Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of Neuroscience Methods*, 258:1–15, 2016. doi: 10.1016/j.jneumeth.2015.10.010.

Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, Eurika Kaiser, and J. Nathan Kutz. Chaos as an intermittently forced linear system. *Nature Communications*, 8(19): 1–9, 2017. doi: 10.1038/s41467-017-00030-8.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern Koopman theory for dynamical systems. *SIAM Review*, 64(2):229–340, 2022. doi: 10.1137/21M1401243.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Scott T. Dawson, Maziar S. Hemati, Matthew O. Williams, and Clarence W. Rowley. Characterizing and correcting for the effect of sensor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(42):1–19, 2016. doi: 10.1007/s00348-016-2127-7.

Nicola Demo, Marco Tezzele, and Gianluigi Rozza. PyDMD: Python dynamic mode decomposition. *Journal of Open Source Software*, 3(22):530, 2018. doi: 10.21105/joss.00530.

Daniel Dylewsky, Molei Tao, and J. Nathan Kutz. Dynamic mode decomposition for multiscale nonlinear physics. *Physical Review E*, 99(6):063311, 2019. doi: 10.1103/PhysRevE.99.063311.

N. Benjamin Erichson, Steven L. Brunton, and J. Nathan Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, 16:1479–1492, 2019a. doi: 10.1007/s11554-016-0655-2.

N. Benjamin Erichson, Lionel Mathelin, J. Nathan Kutz, and Steven L. Brunton. Randomized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(4): 1867–1891, 2019b. doi: 10.1137/18M1215013.

Jacob Grosek and J. Nathan Kutz. Dynamic mode decomposition for real-time background/foreground separation in video, 2014. Preprint, https://arxiv.org/abs/1404.7592.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825): 357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

Patrick Héas and Cédric Herzet. Low-rank dynamic mode decomposition: An exact and tractable solution. *Journal of Nonlinear Science*, 32(8):8, 2022. doi: 10.1007/s00332-021-09770-w.

Maziar S. Hemati, Clarence W. Rowley, Eric A. Deem, and Louis N. Cattafesta. De-biasing the dynamic mode decomposition for applied Koopman spectral analysis of noisy datasets. *Theoretical and Computational Fluid Dynamics*, 31:349–368, 2017. doi: 10.1007/s00162-017-0432-2.

Martin W. Hess, Annalisa Quaini, and Gianluigi Rozza. A data-driven surrogate modeling approach for time-dependent incompressible Navier-Stokes equations with dynamic mode decomposition and manifold interpolation. *Advances in Computational Mathematics*, 49 (22):22, 2023. doi: 10.1007/s10444-023-10016-4.

Seth M. Hirsh, Kameron Decker Harris, J. Nathan Kutz, and Bingni W. Brunton. Centering data improves the dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 19(3):1920–1955, 2020. doi: 10.1137/19M1289881.

Seth M. Hirsh, Sara M. Ichinaga, Steven L. Brunton, J. Nathan Kutz, and Bingni W. Brunton. Structured time-delay models for dynamical systems with connections to Frenet–Serret frame. *Proceedings of the Royal Society A*, 477(2254):20210097, 2021. doi: 10.1098/rspa.2021.0097.

John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Mihailo R. Jovanović, Peter J. Schmid, and Joseph W. Nichols. Sparsity-promoting dynamic mode decomposition. *Physics of Fluids*, 26(2):024103, 2014. doi: 10.1063/1.4863670.

J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016a. doi: 10.1137/1.9781611974508.

J. Nathan Kutz, Xing Fu, and Steven L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016b. doi: 10.1137/15M1023543.

Karl Lapo, Sara M. Ichinaga, and J. Nathan Kutz. Unsupervised multi-scale diagnostics, 2024. Preprint, https://arxiv.org/abs/2408.02396.

Soledad Le Clainche and José M. Vega. Higher order dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 16(2):882–925, 2017. doi: 10.1137/15M1054924.

Jordan Mann and J. Nathan Kutz. Dynamic mode decomposition for financial trading strategies. *Quantitative Finance*, 16:1643–1655, 2016. doi: 10.1080/14697688.2016.1170194.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011.

Joshua L. Proctor and Philip A. Eckhoff. Discovering dynamic patterns from infectious disease data using dynamic mode decomposition. *International Health*, 7(2):139–145, 2015. doi: 10.1093/inthealth/ihv009.

Joshua L. Proctor, Steven L. Brunton, and J. Nathan Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016. doi: 10.1137/15M1013857.

Clarence W. Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S. Henningson. Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127, 2009. doi: 10.1017/S0022112009992059.

Diya Sashidhar and J. Nathan Kutz. Bagging, optimized dynamic mode decomposition for robust, stable forecasting with spatial and temporal uncertainty quantification. *Proceedings of the Royal Society A*, 380(2229):20210199, 2022. doi: 10.1098/rsta.2021.0199.

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

Peter J. Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54:225–254, 2022. doi: 10.1146/annurev-fluid-030121-015835.

Peter J. Schmid and Joern Sesterhenn. Dynamic mode decomposition of numerical and experimental data. In *61st Annual Meeting of the APS Division of Fluid Dynamics*. American Physical Society, 2008. http://meetings.aps.org/link/BAPS.2008.DFD.MR.7.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Subspace dynamic mode decomposition for stochastic Koopman analysis. *Physical Review E*, 96(3):033310, 2017. doi: 10.1103/PhysRevE.96.033310.

Roy Taylor, J. Nathan Kutz, Kyle Morgan, and Brian A. Nelson. Dynamic mode decomposition for plasma diagnostics and validation. *Review of Scientific Instruments*, 89(5): 053501, 2018. doi: 10.1063/1.5027419.

Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and J. N. Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1:391–421, 2014. doi: 10.3934/jcd.2014.1.391.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25:1307–1346, 2015a. doi: 10.1007/s00332-015-9258-5.

Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2015b. doi: 10.3934/jcd.2015005.