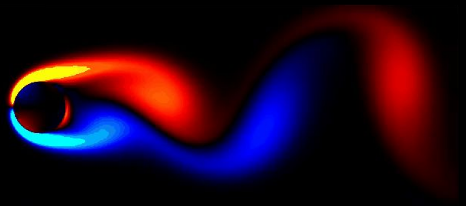
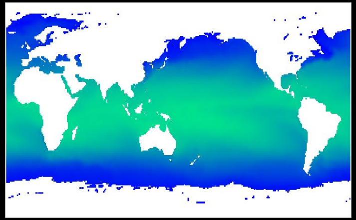
Open-Source Algorithms for Physics-Informed Data-Driven Modeling in Python





Sara M. Ichinaga
Department of Applied Mathematics
University of Washington

Steven Brunton
University of Washington

J. Nathan Kutz
University of Washington

APPLIED MATHEMATICS
UNIVERSITY of WASHINGTON

Before we get started…

Presentation overview:

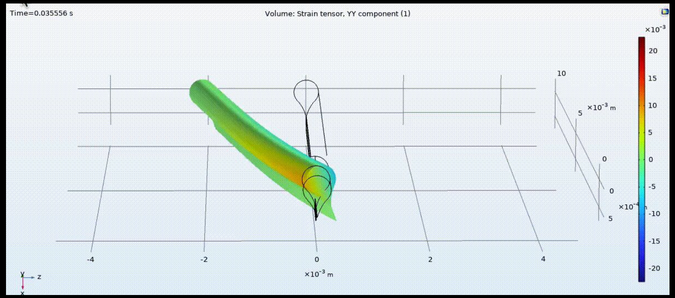
- Introduction to learning dynamics from data
- Method 1: dynamic mode decomposition (DMD)
  - Fluid dynamics code demo with PyDMD
- Method 2: sparse identification of nonlinear dynamics (SINDy)
  - Predator-prey code demo with PySINDy
- Conclusion and method comparison

Code and slides available at:

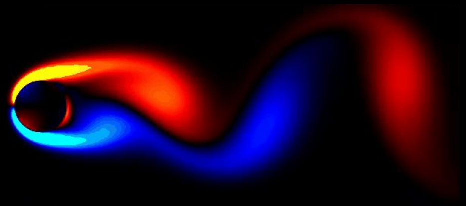
github.com/sichinaga/python-dynamics-tutorial

Introduction and motivation

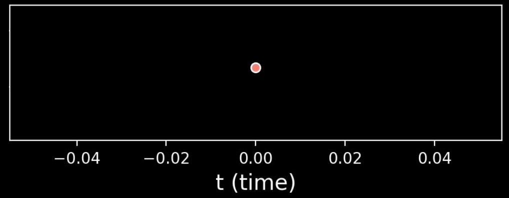
Aeroelasticity



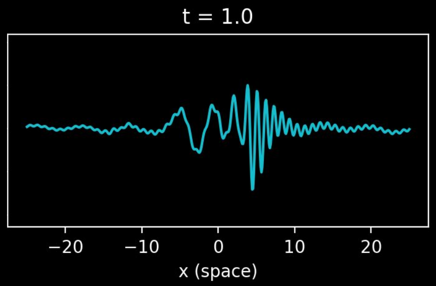
Fluid dynamics



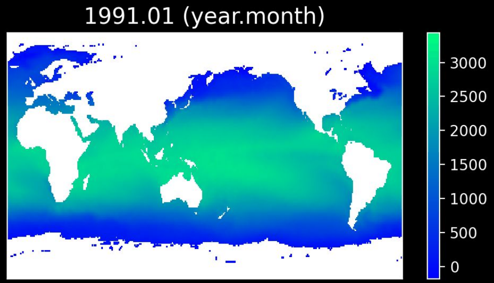
Time-series analysis



Waves and optics



Climate science

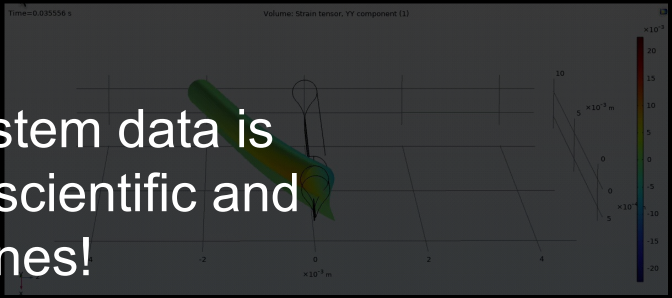


Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

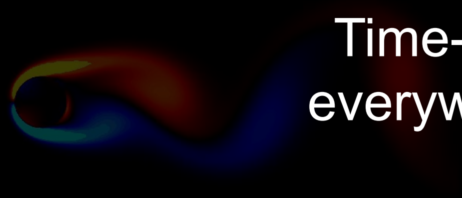
Kutz, Brunton, Brunton, Proctor, *SIAM* 2016.
Rayner, et al., *Journal of Geophysical Research: Atmospheres* 2003.

Introduction and motivation

Aeroelasticity

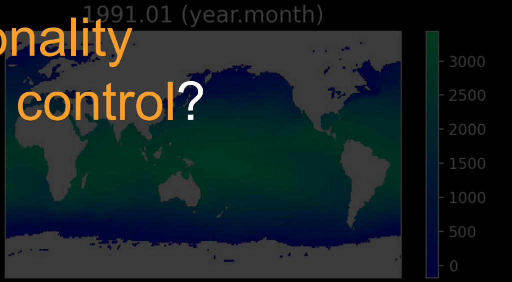


Fluid dynamics

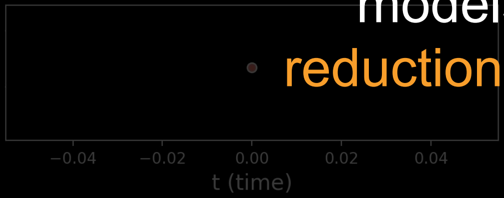


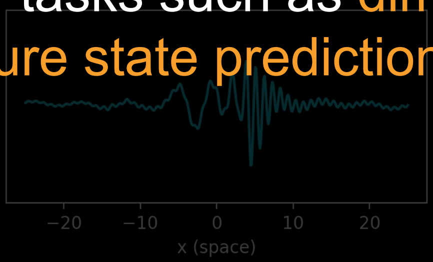
Time-varying dynamical system data is everywhere across multiple scientific and engineering disciplines!

Climate science



Time-series analysis





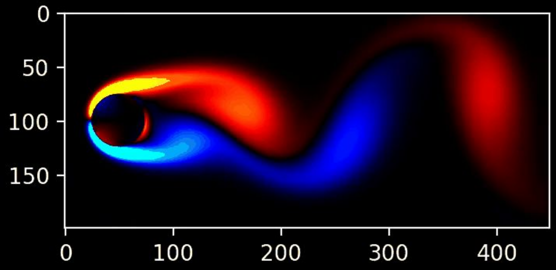
How might we leverage data in order to craft models for tasks such as dimensionality reduction, future state prediction, and control?

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

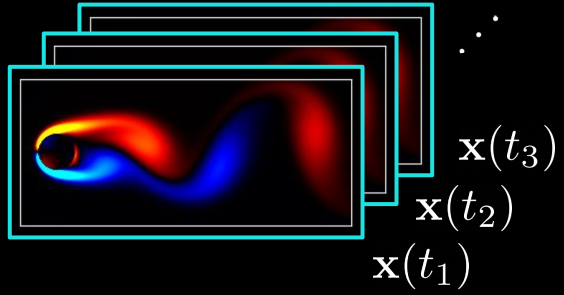
Kutz, Brunton, Brunton, Proctor, *SIAM* 2016.
Rayner, et al., *Journal of Geophysical Research: Atmospheres* 2003.

Learning dynamics from data

Collect data



Each snapshot is an observation of the state



$$\mathbf{x}(t) \in \mathbb{R}^n$$

Find a system of equations that best-describes the observed dynamics

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$$

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Method 1: Dynamic Mode Decomposition (DMD)

Dynamic mode decomposition (DMD)

The simplest model that we can find is a linear one. $\underline{\mathbf{A} \in \mathbb{R}}^{n\times n}$

$$\dot{\mathbf{x}}(t) = \underline{\mathbf{A}}\mathbf{x}(t)$$

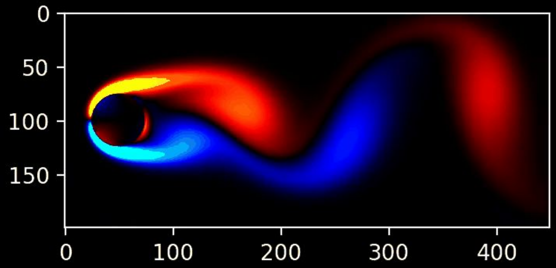
The solution to this general system of equations is known:

$$\mathbf{x}(t) = \underbrace{\begin{bmatrix} | & & | \\ \phi_1 & \dots & \phi_r \\ | & & | \end{bmatrix}}_{\text{Eigenvectors of A}} \underbrace{\begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_r \end{bmatrix}}_{\text{Amplitudes for reconstruction}} \underbrace{\begin{bmatrix} e^{\omega_1 t} \\ \vdots \\ e^{\omega_r t} \end{bmatrix}}_{\text{Eigen-values of A}} = \mathbf{\Phi}\mathrm{diag}(\mathbf{b})e^{\boldsymbol{\omega} t}$$

Schmid, *JFM* 2010.
Rowley, Mezic, Bagheri, Schlatter, Henningson, *JFM* 2009.
Tu, Rowley, Luchtenburg, Brunton, Kutz, *JCD* 2014.
Kutz, Brunton, Brunton, Proctor, *SIAM* 2016.

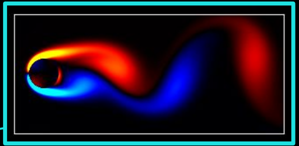
Dynamic mode decomposition (DMD)

Collect data



Organize data into a matrix

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \dots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}$$



Snapshot at time t□

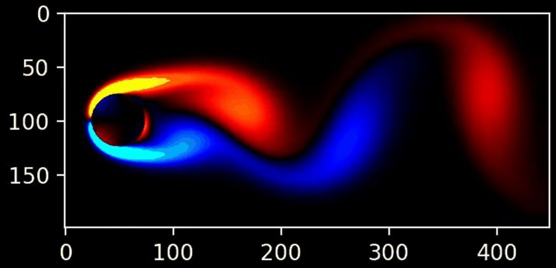
Decompose into… Spatial Modes Amplitudes Time Dynamics

$$\mathbf{X} \approx \begin{bmatrix} | & & | \\ \phi_1 & \dots & \phi_r \\ | & & | \end{bmatrix} \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_r \end{bmatrix} \begin{bmatrix} e^{\omega_1 t_1} & \dots & e^{\omega_1 t_m} \\ \vdots & \ddots & \vdots \\ e^{\omega_r t_1} & \dots & e^{\omega_r t_m} \end{bmatrix} = \boldsymbol{\Phi}\mathrm{diag}(\mathbf{b})\mathbf{T}(\boldsymbol{\omega})$$

Schmid, *JFM* 2010.
Rowley, Mezic, Bagheri, Schlatter, Henningson, *JFM* 2009.
Tu, Rowley, Luchtenburg, Brunton, Kutz, *JCD* 2014.
Kutz, Brunton, Brunton, Proctor, *SIAM* 2016.

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Dynamic mode decomposition (DMD)



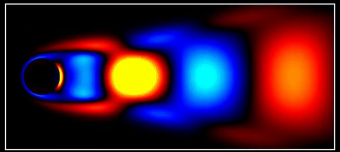
Spatial Modes Amplitudes Time Dynamics

$$\approx \begin{bmatrix} | & & | \\ \phi_1 & \cdots & \phi_r \\ | & & | \end{bmatrix} \begin{bmatrix} b_1 & & \\ & \ddots & \\ & & b_r \end{bmatrix} \begin{bmatrix} — & e^{\omega_1 t} & — \\ & \vdots & \\ — & e^{\omega_r t} & — \end{bmatrix} = \sum_{i=1}^{r} b_i \phi_i e^{\omega_i t}$$

$\approx$

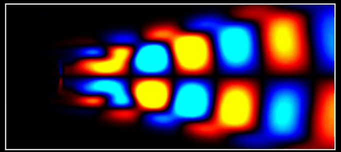
$b_1 \phi_1 e^{\omega_1 t}$

$+$



$b_2 \phi_2 e^{\omega_2 t}$

$+$



$b_3 \phi_3 e^{\omega_3 t}$

$+ \cdots$

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Schmid, *JFM* 2010.
Rowley, Mezic, Bagheri, Schlatter, Henningson, *JFM* 2009.
Tu, Rowley, Luchtenburg, Brunton, Kutz, *JCD* 2014.
Kutz, Brunton, Brunton, Proctor, *SIAM* 2016.

Optimized DMD

Optimization problem:

$$\boxed{\underset{\boldsymbol{\Phi}_{\mathbf{b}},\,\boldsymbol{\omega}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{X} - \boldsymbol{\Phi}_{\mathbf{b}}\mathbf{T}(\boldsymbol{\omega})\|_F^2}$$

Use variable projection for nonlinear least-squares problems with alternating updates:

- Update the modes: for a fixed set of DMD eigenvalues, compute

$$\hat{\boldsymbol{\Phi}}_{\mathbf{b}} = \mathbf{X}\left[\mathbf{T}(\boldsymbol{\omega})\right]^{\dagger}$$

- Update the eigenvalues: for a fixed set of DMD modes, compute a local optimizer

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{X} - \mathbf{X}\left[\mathbf{T}(\boldsymbol{\omega})\right]^{\dagger}\mathbf{T}(\boldsymbol{\omega})\|_F^2$$

via methods such as Levenberg-Marquardt.

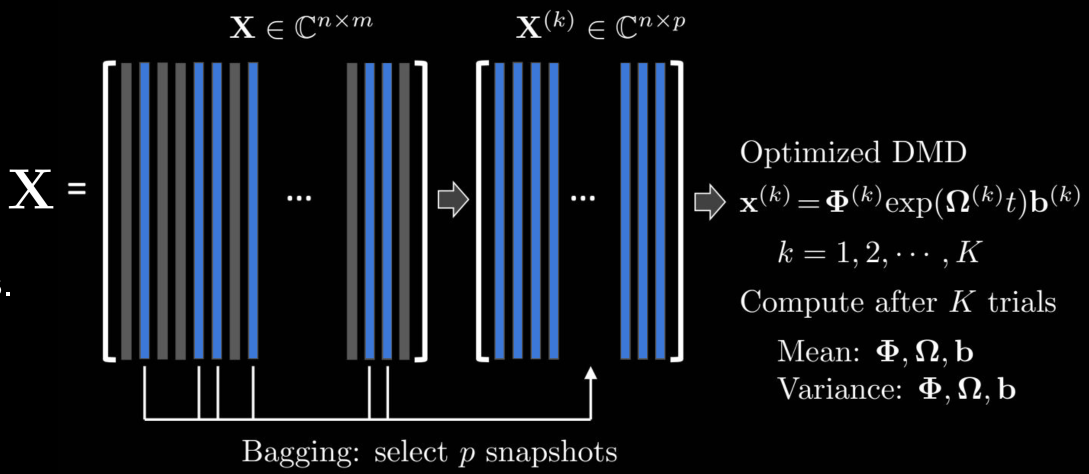
Askham, Kutz, *SIAM* 2018.

Optimized DMD with bagging (BOP-DMD)

$$\underset{\Phi_{\mathbf{b}},\,\boldsymbol{\omega}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{X} - \Phi_{\mathbf{b}}\mathbf{T}(\boldsymbol{\omega})\|_F^2$$

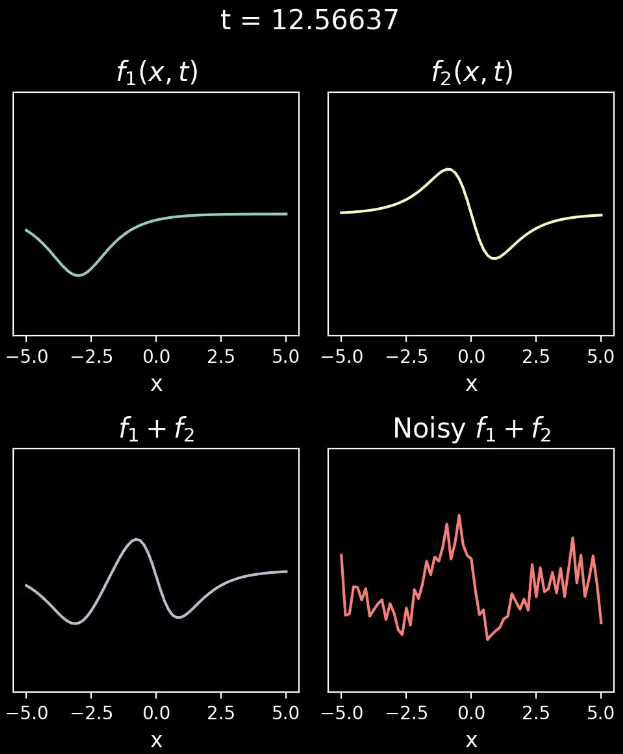
- Optimally suppresses bias from noise.
- Handles snapshots that are unevenly sampled in time.
- Allows for regularization and constraints.
- Requires solving a nonlinear optimization problem.
    - Stabilize with bagging.
    - Gives UQ metrics.

Optimized DMD with bagging



Askham, Kutz, *SIAM* 2018.
Sashidhar, Kutz, *PTRS* 2022.

PyDMD: A Python package for DMD




```python
from pydmd import BOPDMD
from pydmd.preprocessing import hankel_preprocessing
from pydmd.plotter import plot_summary

bopdmd = BOPDMD(
    svd_rank=4,
    num_trials=0,
    eig_constraints={"conjugate_pairs"},
)
delay_bopdmd = hankel_preprocessing(bopdmd, d=2)
delay_bopdmd.fit(X, t=t[:-1])
plot_summary(delay_bopdmd, x=x, t=t[:-1], d=2)
```


Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

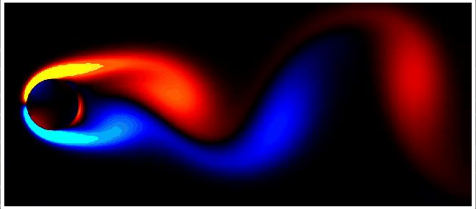
Demo, Tezzele, Rozza, *JOSS* 2018.
Ichinaga, Andreuzzi, Demo, Tezzele, Lapo, Rozza, Brunton, Kutz, *JMLR* 2024.

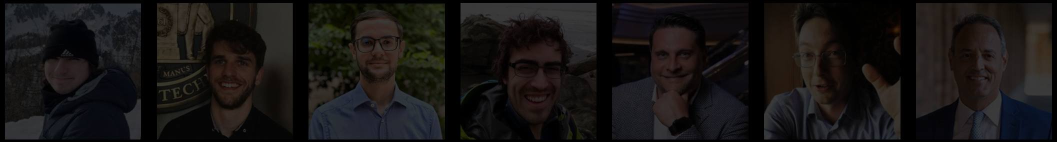
PyDMD: A Python package for DMD

PyDMD Demo: Flow Past a Cylinder

Follow along using the code found at

github.com/sichinaga/python-dynamics-tutorial





Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Demo, Tezzele, Rozza, *JOSS* 2018.
Ichinaga, Andreuzzi, Demo, Tezzele, Lapo, Rozza, Brunton, Kutz, *JMLR* 2024.

Method 2: Sparse Identification of Nonlinear Dynamics (SINDy)

Sparse Identification of Nonlinear Dynamics (SINDy)

Instead of a linear model (DMD)...

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t)$$

$$\dot{x}_1(t) = \underline{a}x_1(t) + \underline{b}x_2(t) + \cdots + \underline{c}x_n(t)$$

$$\vdots$$

Suppose we instead looked for a nonlinear model (SINDy), which permits the use of nonlinear terms such as:

$$ax_1^2(t) \qquad bx_1(t)x_2(t) \qquad c\sin(x_1(t))$$

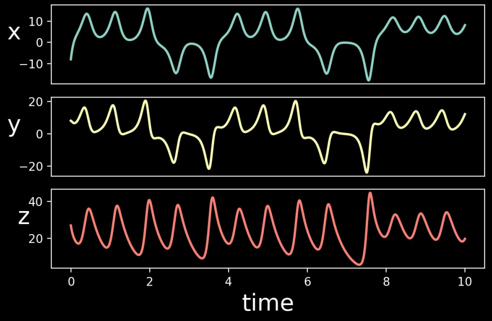
Higher-order polynomials — Nonlinear functions

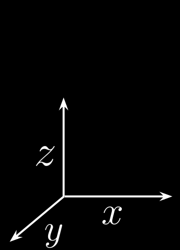
Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Brunton, Proctor, Kutz, *PNAS* 2016.

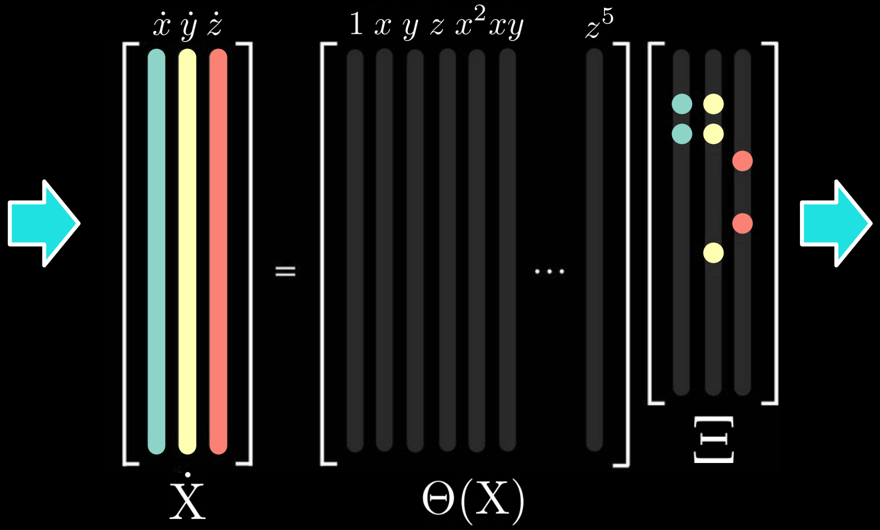
Sparse Identification of Nonlinear Dynamics (SINDy)

Collect dynamical system data





Apply SINDy to data

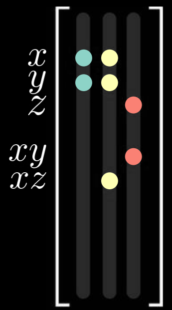


Compute time derivatives

Build nonlinear feature library

Perform sparse regression

Obtain system of equations

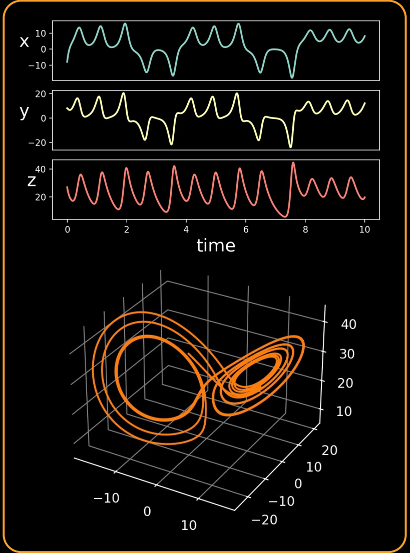


$$\begin{cases} \dot{x} = -10x + 10y \\ \dot{y} = 28x - y - xz \\ \dot{z} = -\frac{8}{3}z + xy \end{cases}$$

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

Brunton, Proctor, Kutz, *PNAS* 2016.

PySINDy: A Python package for SINDy

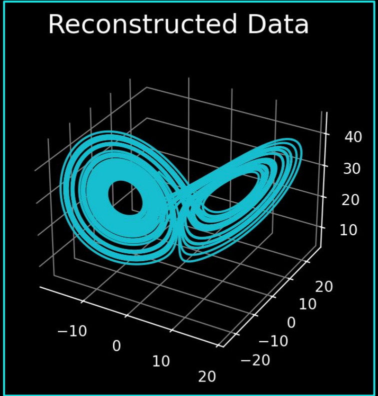



```python
import pysindy as ps

differentiation_method = ps.FiniteDifference(order=2)
feature_library = ps.PolynomialLibrary(degree=3)
optimizer = ps.STLSQ(threshold=0.2)

model = ps.SINDy(
    differentiation_method=differentiation_method,
    feature_library=feature_library,
    optimizer=optimizer,
    feature_names=["x","y","z"],
)
model.fit(X,t=t)

model.print()
X_reconstruction = model.simulate(x0=X[0],t=t_long)
```

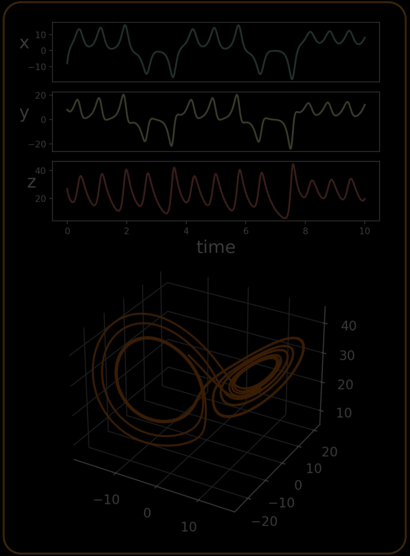



```
(x)' = -9.999 x + 9.999 y
(y)' = 27.992 x + -0.999 y + -1.000 x z
(z)' = -2.666 z + 1.000 x y
```

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

de Silva, Champion, Quade, Loiseau, Kutz, Brunton, *JOSS* 2020.
Kaptanoglu et al., *JOSS* 2022.

PySINDy: A Python package for SINDy




```
import pysindy as ps
```

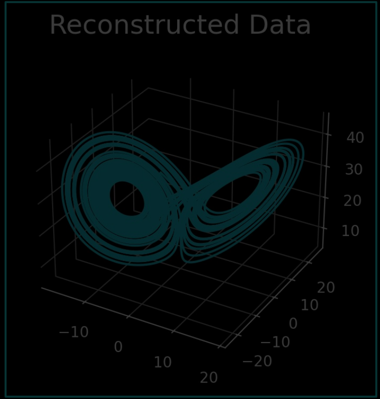
PySINDy Demo: Predator-Prey System

Follow along using the code found at

github.com/sichinaga/python-dynamics-tutorial

```
    optimizer=optimizer,
    feature_names=["x","y","z"],
)
model.fit(X,t=t)

model.print()
X_reconstruction = model.simulate(x0=X[0],t=t_long)
```




```
(x)' = -9.999 x + 9.999 y
(y)' = 27.992 x + -0.999 y + -1.000 x z
(z)' = -2.666 z + 1.000 x y
```

Download the slides and code and follow along!
github.com/sichinaga/python-dynamics-tutorial

de Silva, Champion, Quade, Loiseau, Kutz, Brunton, *JOSS* 2020.
Kaptanoglu et al., *JOSS* 2022.

Conclusion and Method Overview

Dynamic Mode Decomposition (DMD)	Sparse Identification of Nonlinear Dynamics (SINDy)
Pros: ● Models are simple and linear. ● Breaks data down into interpretable set of spatiotemporal components. ● Fast, noise-robust algorithms and method variants are available.	**Pros:** ● Models are nonlinear and can describe complex nonlinear dynamics. ● Models are concise and readable thanks to sparsity. ● Also has variants and fast algorithms.
Cons: ● Cannot model certain complex nonlinear behaviors. ● Sometimes requires the use of many spatiotemporal modes.	**Cons:** ● Sparse regression on dictionary matrix can be very large and costly. ● Data must be well-resolved enough to accurately compute derivatives.

Resources and Further Reading

DMD:

- DMD Book: https://epubs.siam.org/doi/book/10.1137/1.9781611974508
- Optimized DMD: https://epubs.siam.org/doi/10.1137/M1124176
- PyDMD package new paper (long ver): https://arxiv.org/abs/2402.07463
- PyDMD package new paper (short ver): https://www.jmlr.org/papers/v25/24-0739.html
- CODE: https://github.com/PyDMD/PyDMD

SINDy:

- Original SINDy paper: https://www.pnas.org/doi/10.1073/pnas.1517384113
- PySINDy package paper 1: https://arxiv.org/abs/2004.08424
- PySINDy package paper 2: https://arxiv.org/abs/2111.08481
- CODE: https://github.com/dynamicslab/pysindy

References

- P. J. Schmid, *Dynamic mode decomposition of numerical and experimental data*, Journal of Fluid Mechanics, 656 (2010), pp. 5–28.
- C. W. Rowley, I. Mezic, S. Bagheri, P. Schlatter, and D. S. Henningson, *Spectral analysis of nonlinear flows*, Journal of Fluid Mechanics, 641 (2009), pp. 115–127.
- J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, *On dynamic mode decomposition: Theory and applications*, Journal of Computational Dynamics, 1 (2014), pp. 391–421.
- J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor, *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016.
- T. Askham and J. N. Kutz, *Variable projection methods for an optimized dynamic mode decomposition*, SIAM Journal on Applied Dynamical Systems, 17 (2018), pp. 380–416.
- D. Sashidhar and J. N. Kutz, *Bagging, optimized dynamic mode decomposition for robust, stable forecasting with spatial and temporal uncertainty quantification*, Proceedings of the Royal Society A, 380 (2022), p. 20210199.
- N. A. Rayner, D. E. Parker, E. B. Horton, C. K. Folland, L. V. Alexander, D. P. Rowell, E. C. Kent, and A. Kaplan, *Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century*, Journal of Geophysical Research: Atmospheres, 108 (2003).

References

- N. Demo, M. Tezzele, and G. Rozza, *PyDMD: Python dynamic mode decomposition*, Journal of Open Source Software, 3 (2018), p. 530.
- S. M. Ichinaga, F. Andreuzzi, N. Demo, M. Tezzele, K. Lapo, G. Rozza, S. L. Brunton, and J. N. Kutz, *PyDMD: a Python package for robust dynamic mode decomposition*, JMLR, 25 (2024), pp. 1–9.
- S. L. Brunton, J. L. Proctor, and J. N. Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proceedings of the National Academy of Sciences, 113 (2016), pp. 3932–3937.
- B. M. de Silva, K. Champion, M. Quade, J. Loiseau, J. N. Kutz, and S. L. Brunton, *PySINDy: A Python package for the sparse identification of nonlinear dynamical systems from data*, Journal of Open Source Software, 5 (2020), p. 2104.
- A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J. Loiseau, J. N. Kutz, and S. L. Brunton, *PySINDy: A comprehensive Python package for robust sparse system identification*, Journal of Open Source Software, 7 (2022), p. 3994.