Machine learning for discovering sparse models of fluids, plasmas, and much more

Alan A. Kaptanoglu, Jared L. Callaham, Christopher J. Hansen, Steven L. Brunton, + the UW data-driven dynamics lab

University of Maryland, College Park, MD
University of Washington, Seattle, WA

PySINDy Paper:
https://joss.theoj.org/papers/10.21105/joss.03994

Code:
github.com/dynamicslab/pysindy

Youtube lectures:
https://www.youtube.com/playlist?list=PLN90bHJU-JLoOfEk0KyBs2qLTV7OkMZ25

Overview

- What is system identification and why is it a useful scientific tool?
- What is *sparse* system identification?
- Towards very sophisticated data-driven models: overview of advances in the SINDy method
- Implementation of the open-source Python package, PySINDy.

System identification is used to identify dynamical models directly from data

"the art and science of building mathematical models of dynamic systems from observed input–output data"

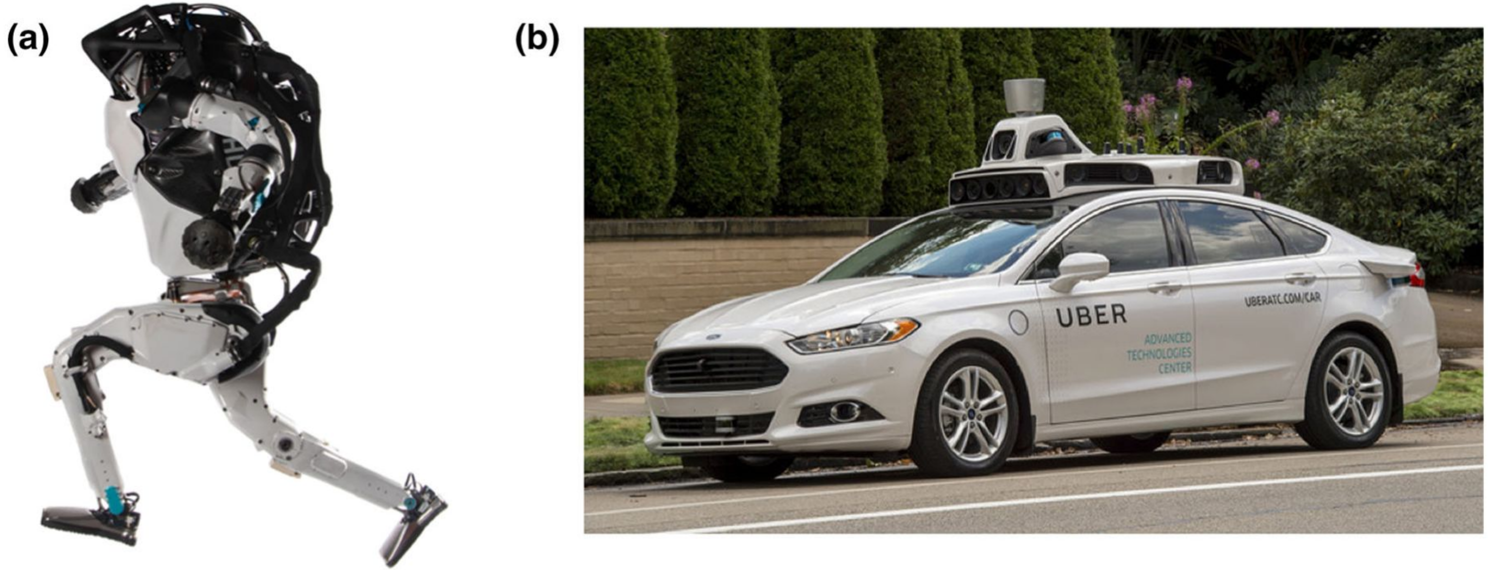
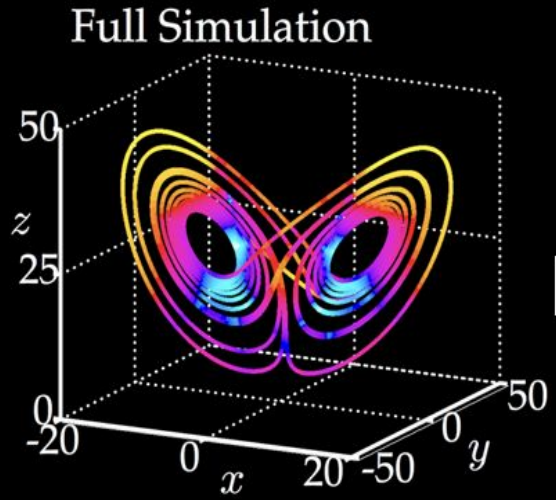


Fig. 2 Modern autonomy leverages sensors to build data-driven models that are physics-based *inductive* models for robotics (**a**) and non-physics-based, deductive statistical models for self-driving cars (**b**)

Inductive models: sparse system identification

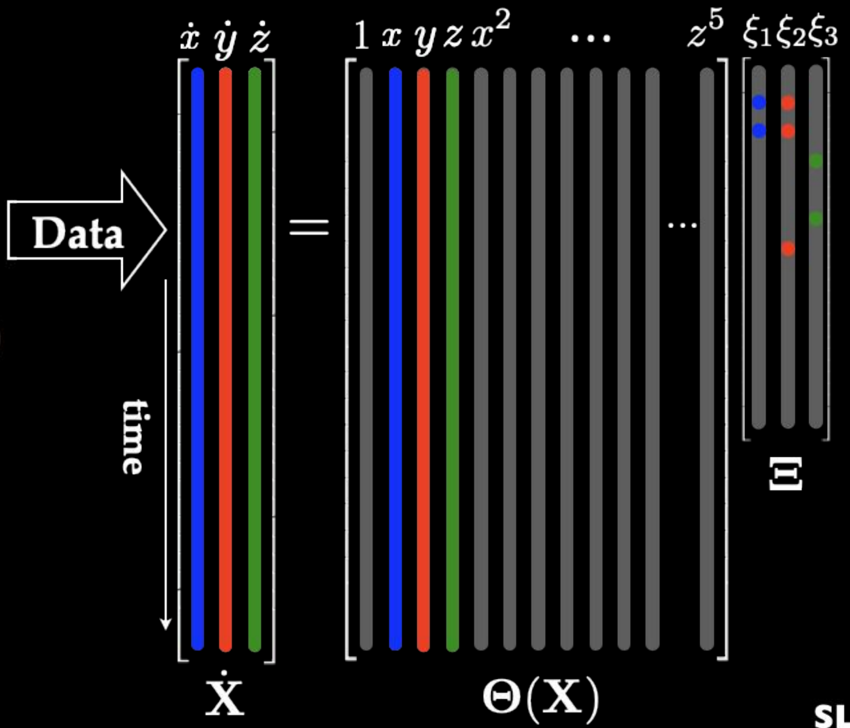
$$\underset{\xi}{\arg\min} \left[\frac{1}{2} \| \Theta \boldsymbol{\xi} - \dot{\boldsymbol{X}} \|^2 + \lambda \| \boldsymbol{\xi} \|_0 \right]$$



$$\dot{x} = -10x + 10y,$$

$$\dot{y} = x(28 - z) - y,$$

$$\dot{z} = xy - \frac{8}{3}z.$$



''	'xi_1'	'xi_2'	'xi_3'
'1'	[0]	[0]	[0]
'x'	[-9.9996]	[27.9980]	[0]
'y'	[9.9998]	[-0.9997]	[0]
'z'	[0]	[0]	[-2.6665]
'xx'	[0]	[0]	[0]
'xy'	[0]	[0]	[1.0000]
'xz'	[0]	[-0.9999]	[0]
'yy'	[0]	[0]	[0]
'yz'	[0]	[0]	[0]
'zz'	[0]	[0]	[0]
...	...	...	...
'yzzzz'	[0]	[0]	[0]
'zzzzz'	[0]	[0]	[0]

SLB, Proctor, Kutz, *PNAS* 2016.

Sparse regression improves models

```
# Instantiate and fit the SINDy model
feature_names = ['x', 'y', 'z']
sparse_regression_optimizer = ps.STLSQ(threshold=0)  # default is lambda = 0.1
model = ps.SINDy(feature_names=feature_names, optimizer=sparse_regression_optimizer)
model.fit(x_train, t=dt)
model.print()
```

```
(x)' = -0.001 1 + -10.005 x + 10.003 y
(y)' = -0.015 1 + 27.991 x + -0.998 y + 0.002 z + -1.000 x z
(z)' = 0.008 1 + 0.006 x + -0.004 y + -2.666 z + 0.001 x^2 + 0.999 x y
```

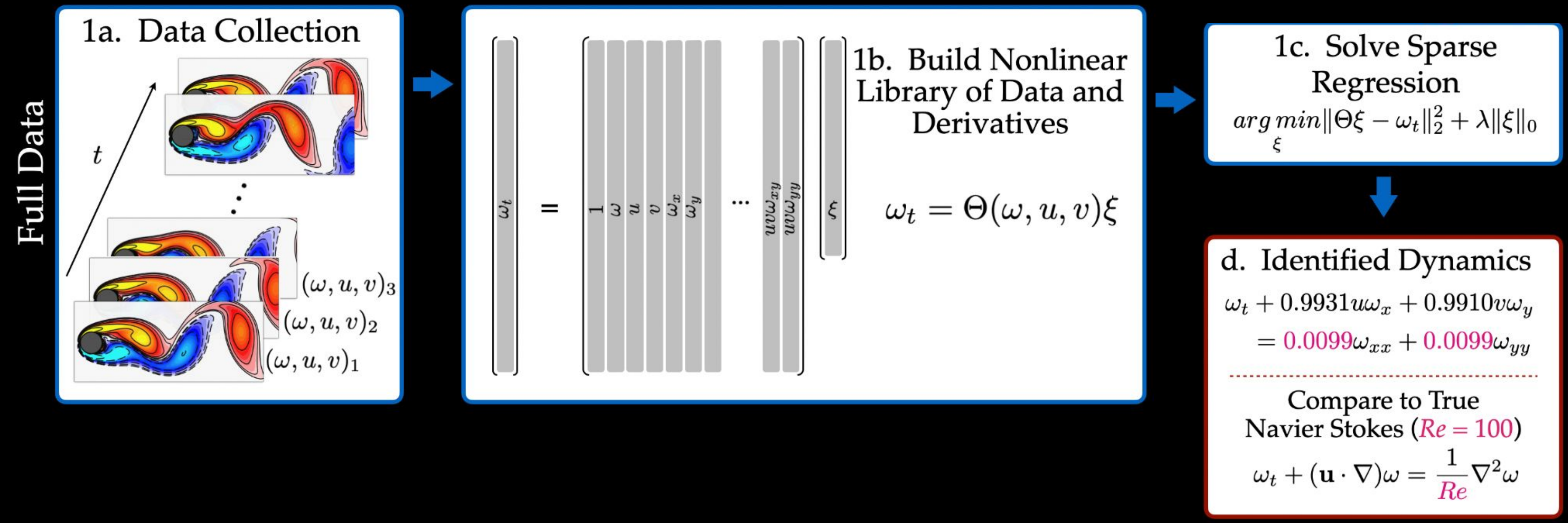
```
sparse_regression_optimizer = ps.STLSQ(threshold=0.1)
model = ps.SINDy(feature_names=feature_names, optimizer=sparse_regression_optimizer)
model.fit(x_train, t=dt)
model.print()
```

```
(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
```

Expanding SINDy for identifying PDEs from data

PDEs

Rudy, SLB, Proctor, Kutz
Science Advances, 2017



Physical priors can be incorporated into system ID

Innovation: Enforcing known constraints

$$\int_{\Omega} \boldsymbol{u} \cdot (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \, \mathrm{d}\Omega = 0 \Rightarrow \boldsymbol{a} \cdot \mathcal{N}(\boldsymbol{a}) = 0$$

- **Skew-symmetric quadratic nonlinearities to enforce energy conservation**
- **Improved stability**

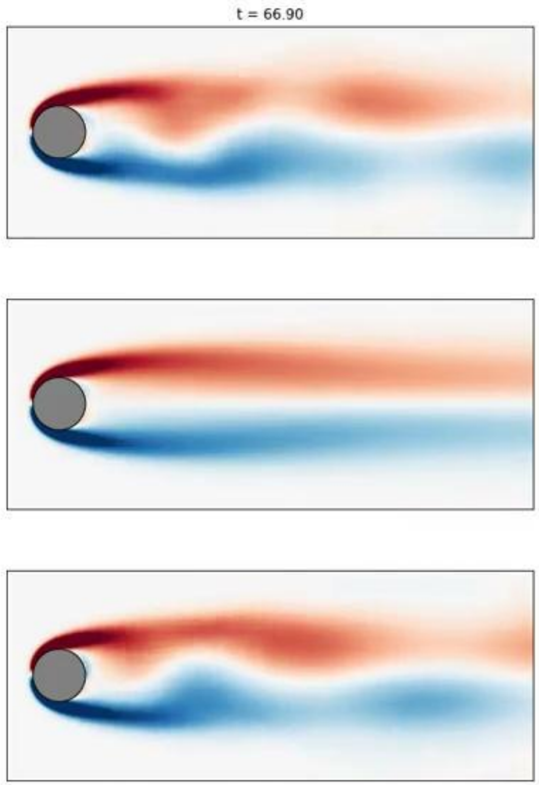
$$\min_{\xi, z} \|\boldsymbol{\Theta}(\boldsymbol{X})\boldsymbol{\Xi} - \dot{\boldsymbol{X}}\|_2^2 + z^T(\boldsymbol{C}\boldsymbol{\xi} - \boldsymbol{d})$$

Additional skew-symmetric quadratic nonlinearities occur in magnetohydrodynamics!

Loiseau & SLB, *JFM* 838, 2018

Schlegel & Noack, *JFM*, 2015

Trapping SINDy – globally stable models by construction



Ground truth

9-mode Galerkin model from Noack et al. (2003)

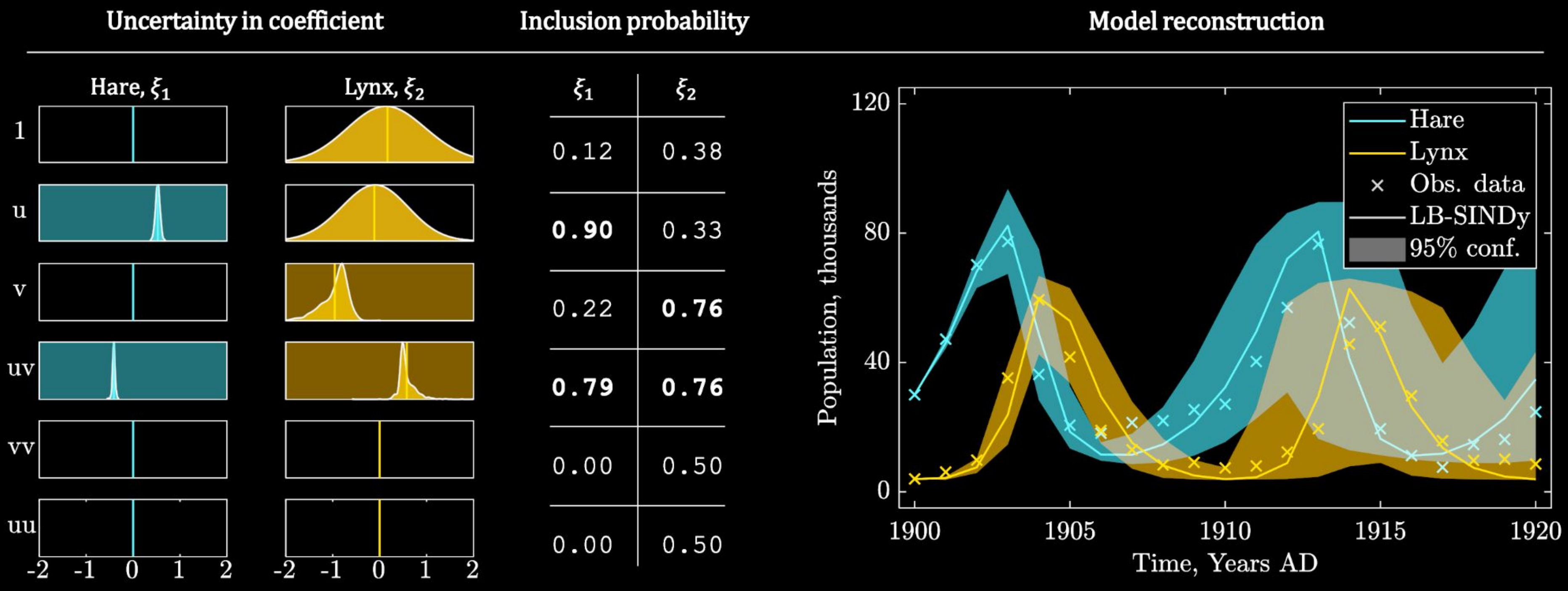
Trapping SINDy model

- Skew-symmetry constraint helps for obtaining more stable, accurate data-driven models.
- Schlegel & Noack - conditions for globally stable, quadratic, reduced-order models.
- Idea of Trapping SINDy – build these conditions directly into the SINDy objective function and therefore obtain a-priori globally stable models.

Kaptanoglu, Callaham, Aravkin, Hansen, SLB, *PRFluids*, 2021

Fasel, et al. *arXiv:2111.10992*

Model ensembles reduce model sensitivity to noise and allow for UQ



Reinbold et al., *PRE*, 2020
Messenger & Bortz, *JCP*, 2021

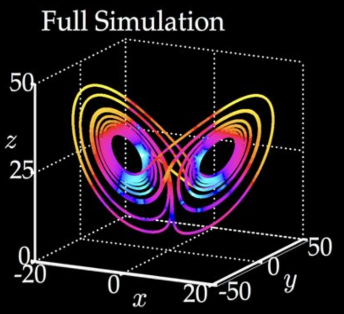
Identifying weak formulations drastically reduces sensitivity of system ID to noise

$$\partial_t \mathbf{q}(\mathbf{x}, t) \approx \boldsymbol{\Theta}(\mathbf{x}, t)\boldsymbol{\Xi}$$

$$\int_{\Omega_k} \mathbf{w}(\mathbf{x}, t) \cdot \partial_t \mathbf{q}(\mathbf{x}, t) dS \approx \left(\int_{\Omega_k} \mathbf{w}(\mathbf{x}, t) \cdot \boldsymbol{\Theta}(\mathbf{x}, t) dS \right) \boldsymbol{\Xi} := \Theta_Q \boldsymbol{\Xi}$$

$$\dot{\mathbf{Q}} \approx \boldsymbol{\Theta}_Q \boldsymbol{\Xi}$$

Identifying weak formulations drastically reduces sensitivity of system ID to noise

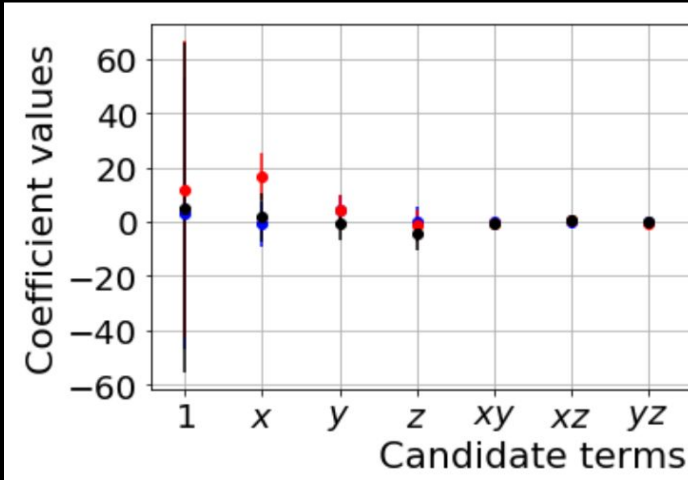


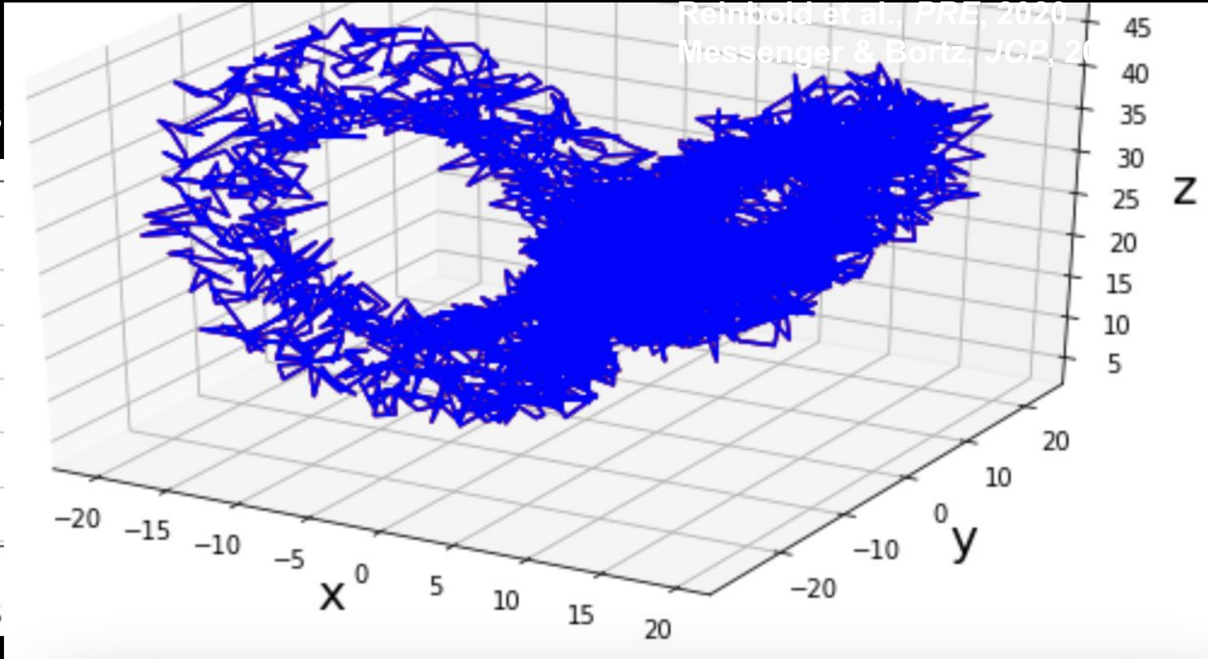
$$\dot{x} = -10x + 10y,$$
$$\dot{y} = x(28 - z) - y,$$
$$\dot{z} = xy - \frac{8}{3}z.$$

+ White noise added N(0, 0.1 * mean(abs(training data)))
  to every single data point in the training data

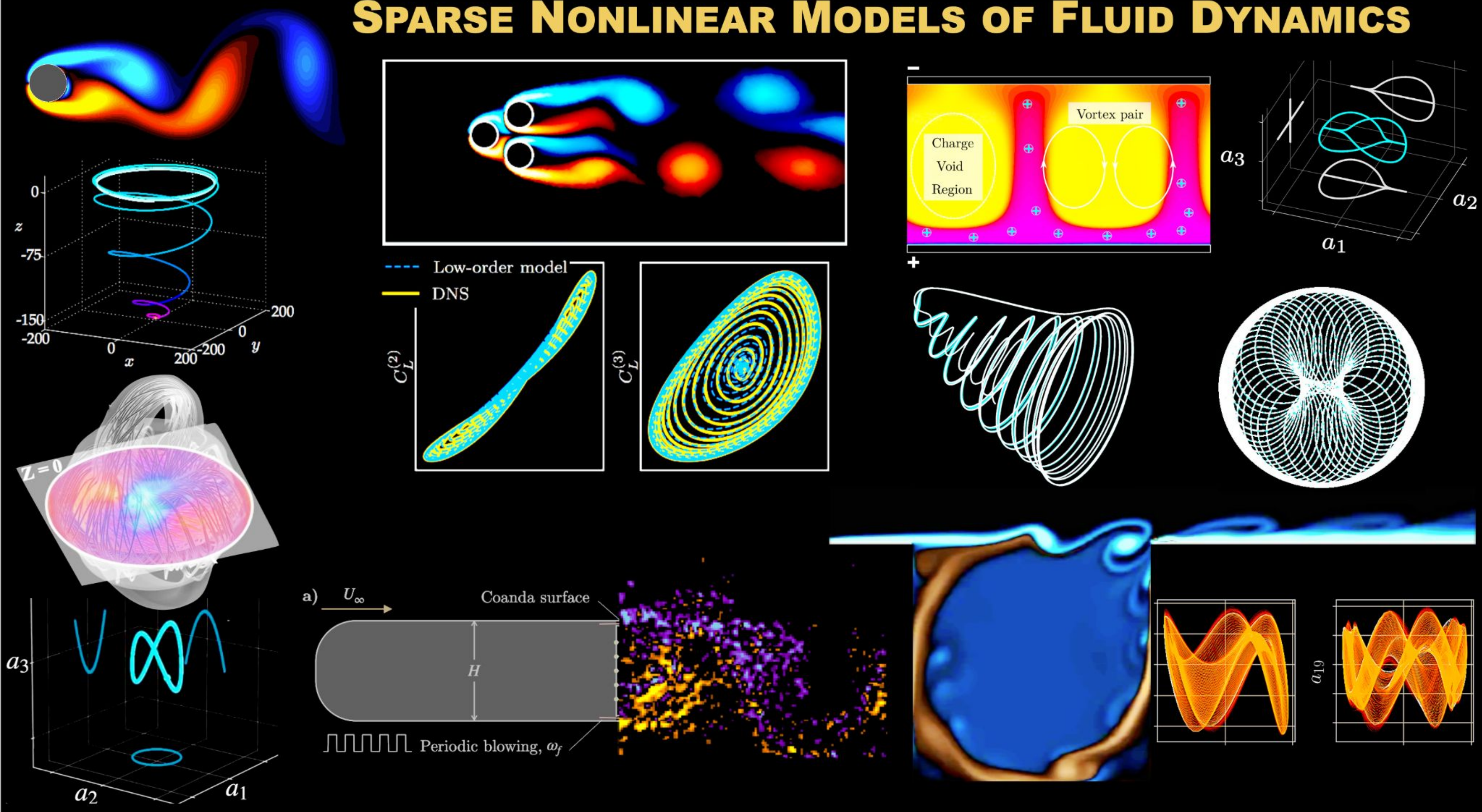
Reinbold et al., *PRE*, 2020
Messenger & Bortz, *JCP*, 2(

Regular S





Sparse Nonlinear Models of Fluid Dynamics



PySINDy: a python code for using SINDy

- Python code built by the data-driven dynamics lab at UW that originally implemented only the traditional SINDy method.

- Why use PySINDy?
  - open-source, high-quality code (PEP8 stylistic standards, unit tests, etc.), many examples, and reproduces some SINDy papers in the literature
  - PySINDy developers can check code for bugs, code quality, etc.
  - Very easy to use
  - Lots of advanced functionality

$$\mathbf{g}(\mathbf{q}, \mathbf{q}_t, \mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_{xx}, \ldots, \mathbf{u}) = 0$$

Kaptanoglu et al., JOSS, 2022

Possible applications in plasma physics

➢ Lots of system ID work in plasma physics – most of the work has been with linear control models or fully black-box neural networks, but have good reasons to want *interpretable* and *nonlinear* ROMs.

➢ Almost no one has started to use the recent system ID advancements for noisy data, UQ, stability, etc., although PySINDy tool is available, so lots of opportunities for papers:
  - Space physics: sophisticated nonlinear Dst models
  - Heliophysics: helicity-preserving ROMs
  - Gyrokinetics: ROMs for forecasting
  - Tokamaks: ROMs for divertor dynamics
  - LAPD: ROMs coupled with optimal sensor placement algorithms
  - + much more

Summary

- System identification can be useful for physical insight, forecasting, and even real-time control of complex dynamical systems.

- *Sparse* system identification produces *parsimonious* dynamical models that reduce overfitting. Further advancements:
  - Identification of general PDEs, systems with control inputs, etc.
  - Constraints from dynamical symmetries
  - Trapping SINDy can build a-priori globally stable fluid models
  - Ensembles of models can be used to improve statistics + UQ
  - Weak formulation of SINDy drastically reduces noise sensitivity

- Open-source PySINDy code makes all of these new tools available and easy-to-use.