The University of Texas at Austin
Oden Institute for Computational Engineering and Sciences

MARCH 2024

PyDMD - DYNAMIC MODE DECOMPOSITION IN PYTHON

MARCO TEZZELE

Postdoctoral fellow, The University of Texas at Austin

Main contributors and maintainers

Nicola Demo

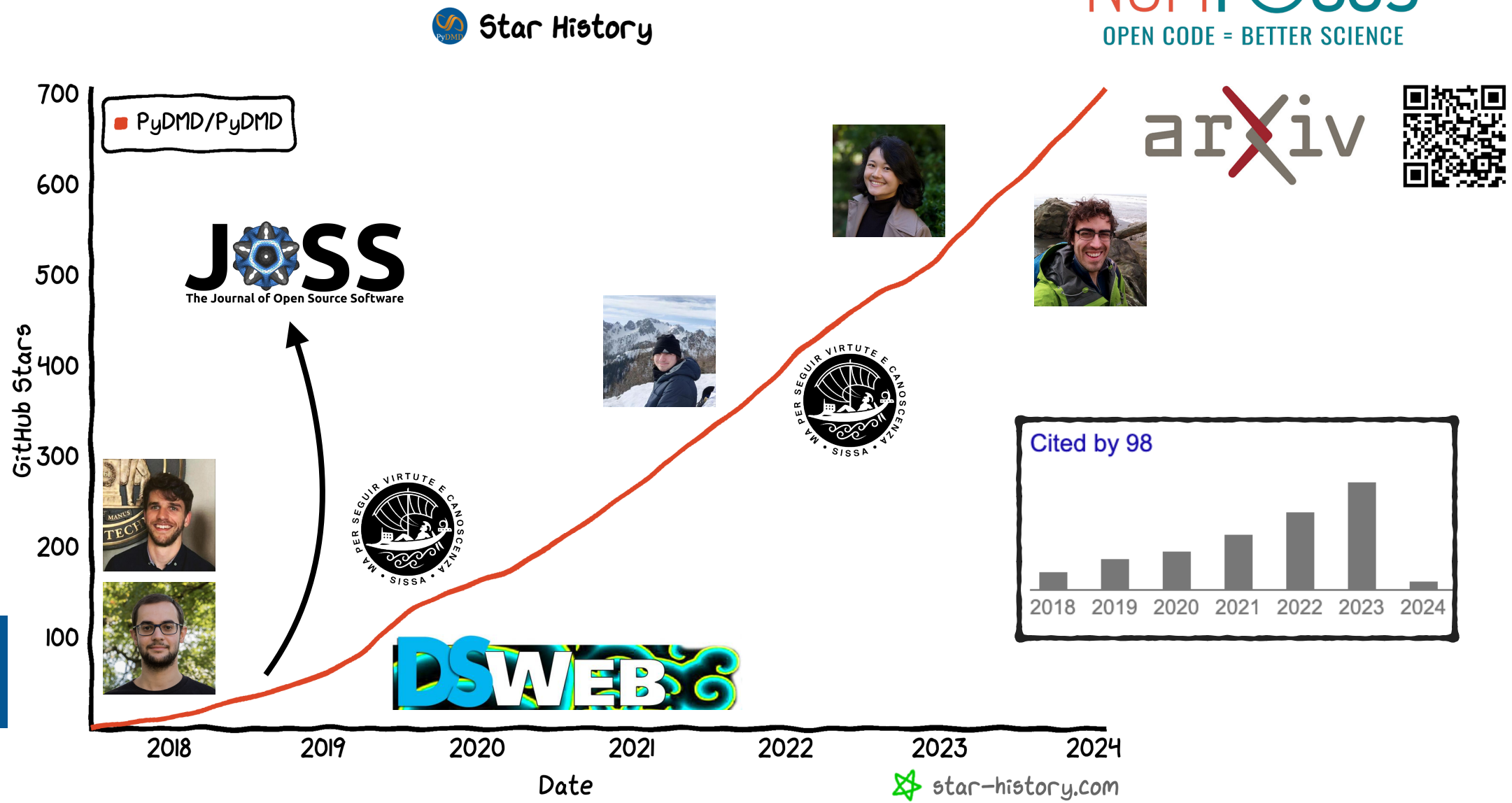
Marco Tezzele

Francesco Andreuzzi

Sara Ichinaga

FAST COMPUTING

Historical perspective



NUMFOCUS
OPEN CODE = BETTER SCIENCE

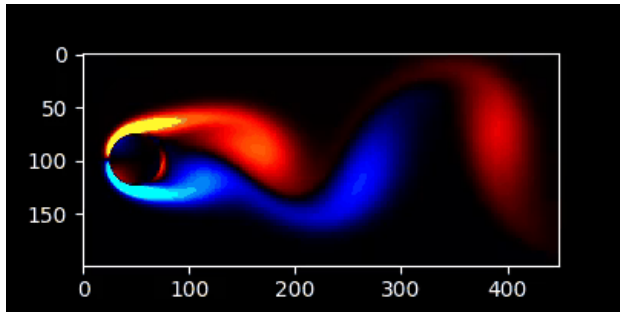
JOSS
The Journal of Open Source Software

arXiv

PyDMD

Dynamic mode decomposition at a glance

Collect data

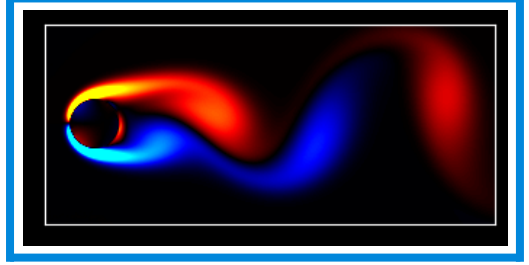


Build data matrices

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_{m-1} \\ | & | & & | \end{bmatrix}$$

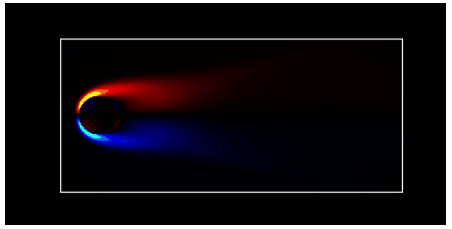
$$X' = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \cdots & x_m \\ | & | & & | \end{bmatrix}$$

$t = t_2$

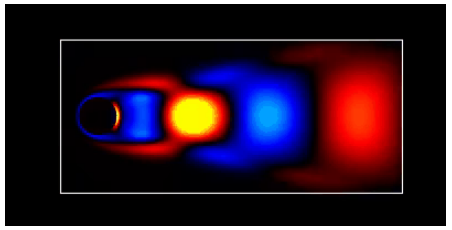


$$\mathrm{A} = X'X^{\dagger}$$
$$\mathrm{A}\Phi = \Phi\Lambda$$

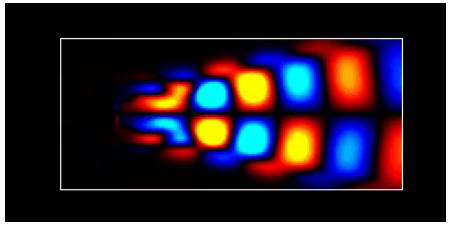
Obtain diagnostics



$f_1 = 0$
(λ_1, φ_1)



$f_2 = 0.208$
(λ_2, φ_2)



$f_4 = 0.416$
(λ_4, φ_4)

$$x_{k+1} = \sum_{j=1}^{r} \varphi_j \lambda_j^k b_j = \Phi \Lambda^k b$$

Schmid, JFM 2010.

Rowley, Mezic, et al., JFM 2009.

Tu, Rowley, et al., JCD 2014.

Kutz, Brunton, et al., SIAM 2016.

DMD algorithm

1. Collection of the snapshots

- Creation of two matrices: snapshots (**X**) and time-shifted snapshots (**Y**)

2. Eigenvalues and modes computation

- Approximation of the operator **A** $\mathbf{Y} \approx \mathbf{A}\mathbf{X}$
- SVD of the snapshots matrix $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$
- Reduction of the operator $\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}$
- Eigendecomposition $\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda}$
- Computation of the modes $\mathbf{\Phi} = \mathbf{Y}\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W}$ $\mathbf{\Phi} = \mathbf{U}\mathbf{W}$

3. Reconstruction and future state predictions

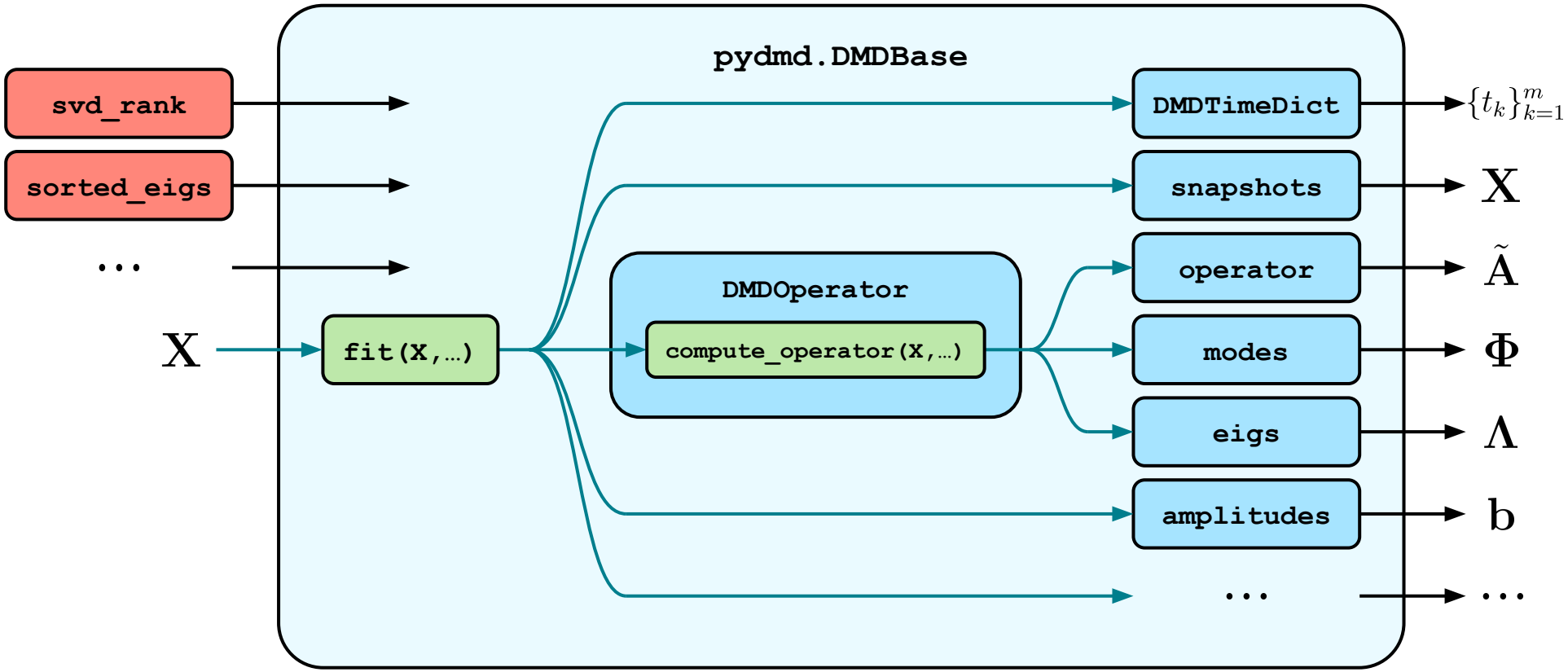
- Computation of the amplitudes **b** $\mathbf{b} = \mathbf{\Phi}^{\dagger}\mathbf{x}_0$
- Forecasting $\mathbf{x}_k = \mathbf{\Phi}\mathbf{\Lambda}^k\mathbf{b}$

DMDBase class

DMDBase

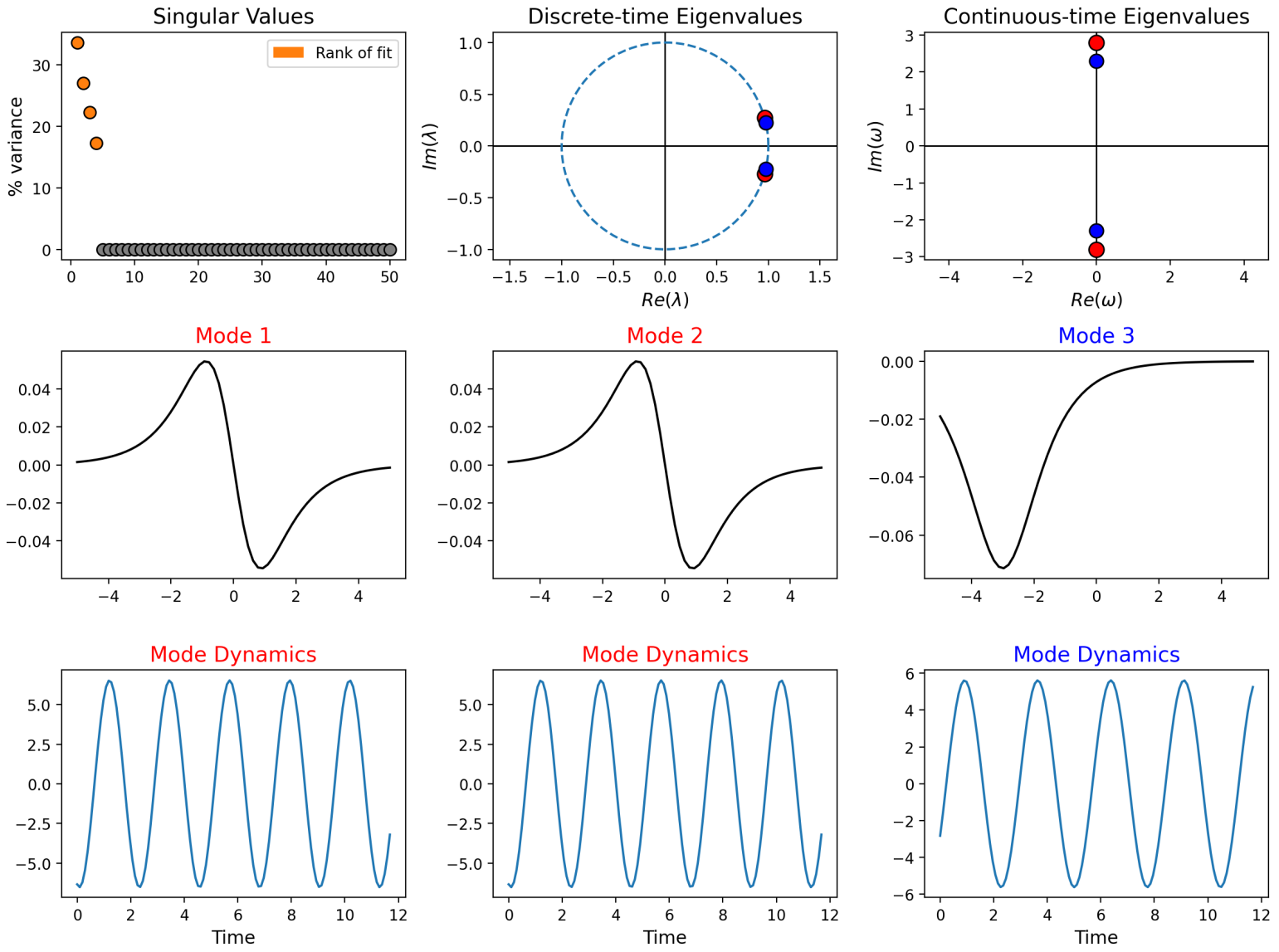
Base module for the DMD: *fit* method must be implemented in inherited classes

class **DMDBase**(*svd_rank=0, tlsq_rank=0, exact=False, opt=False, rescale_mode=None, forward_backward=False, sorted_eigs=False, tikhonov_regularization=None*) [source]

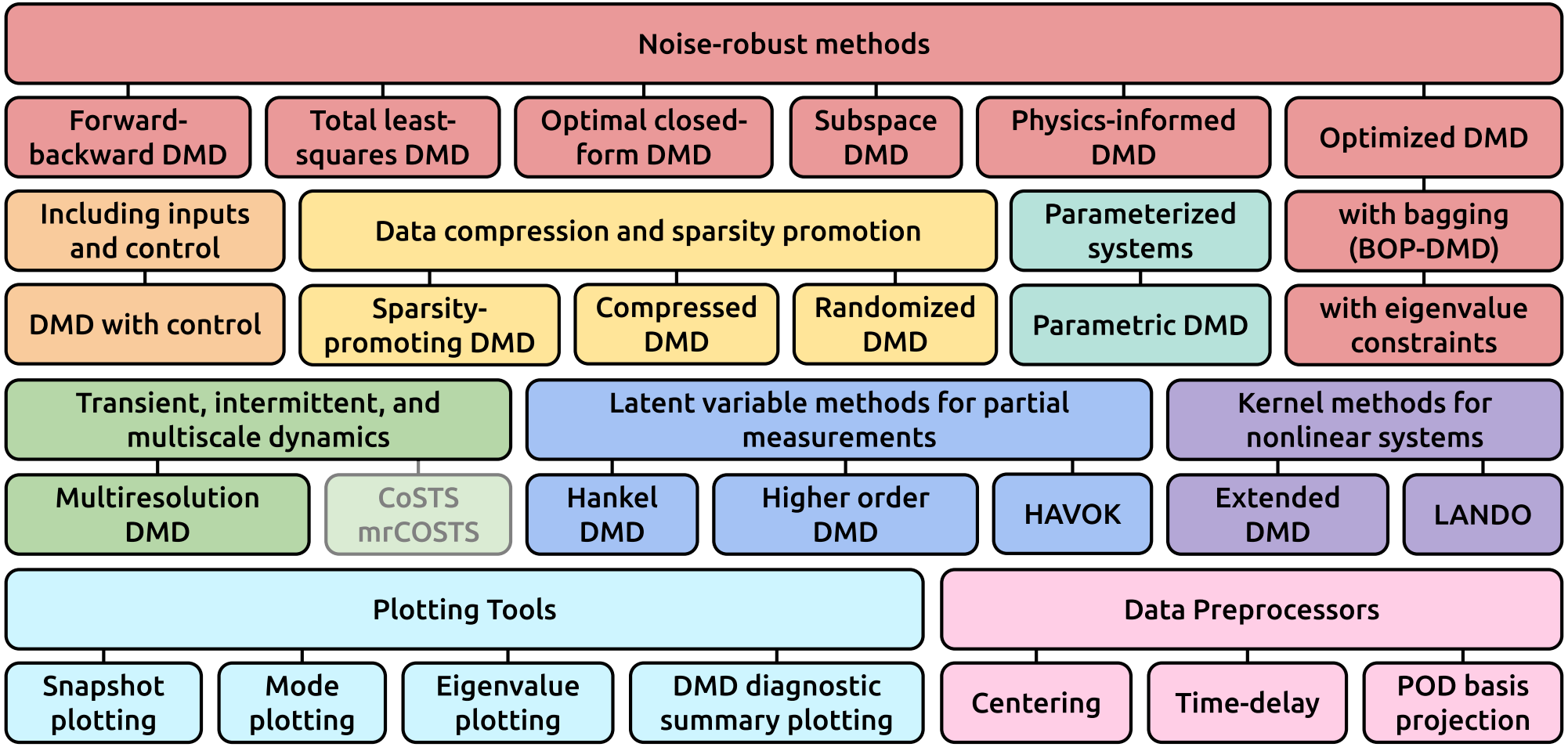


Plotting capabilities for diagnostics and data exploration

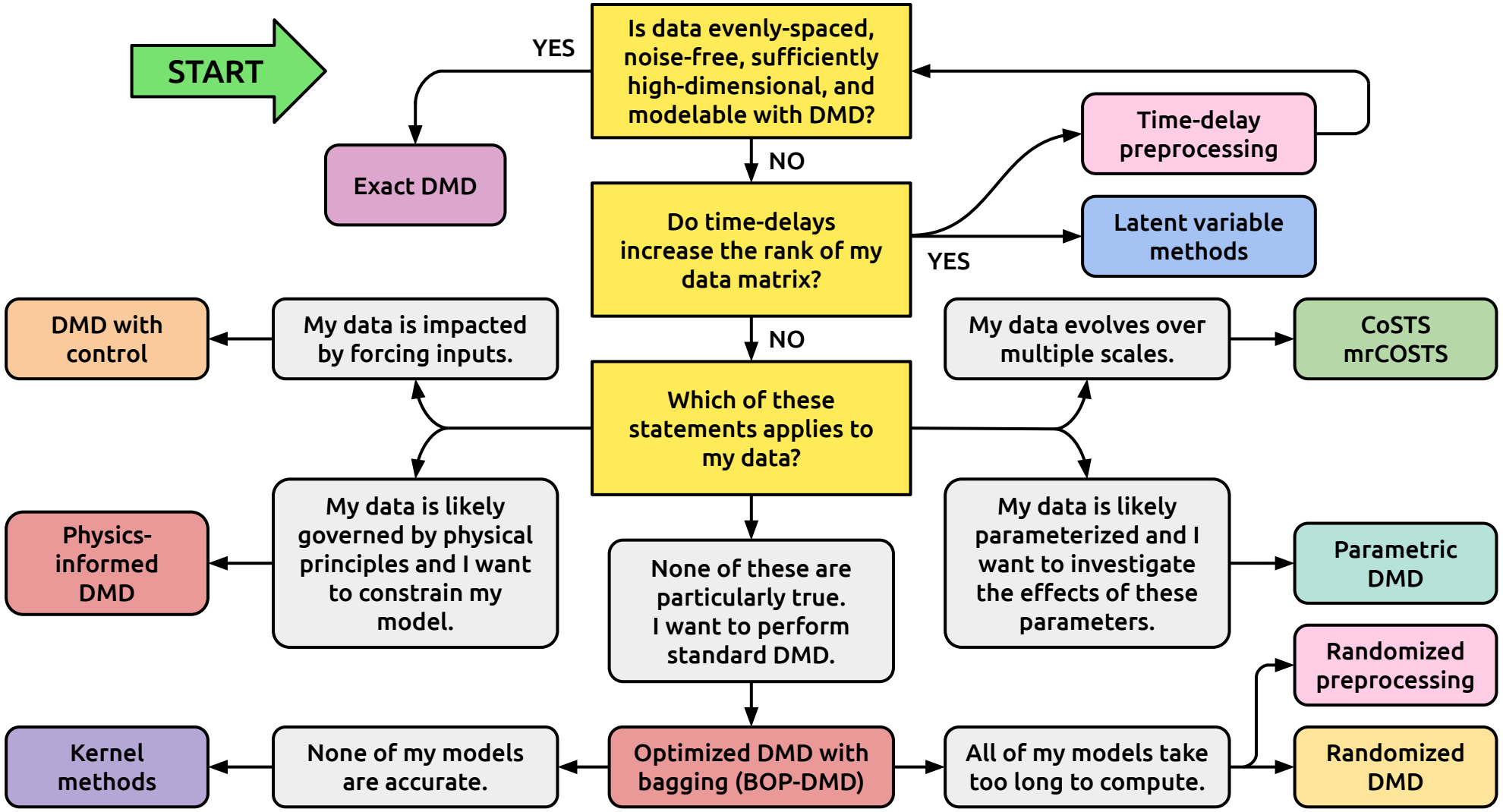
```
def plot_summary(
    dmd,
    *,
    x=None,
    y=None,
    t=None,
    d=1,
    continuous=False,
    snapshots_shape=None,
    index_modes=(0, 1, 2),
    filename=None,
```



Current capabilities

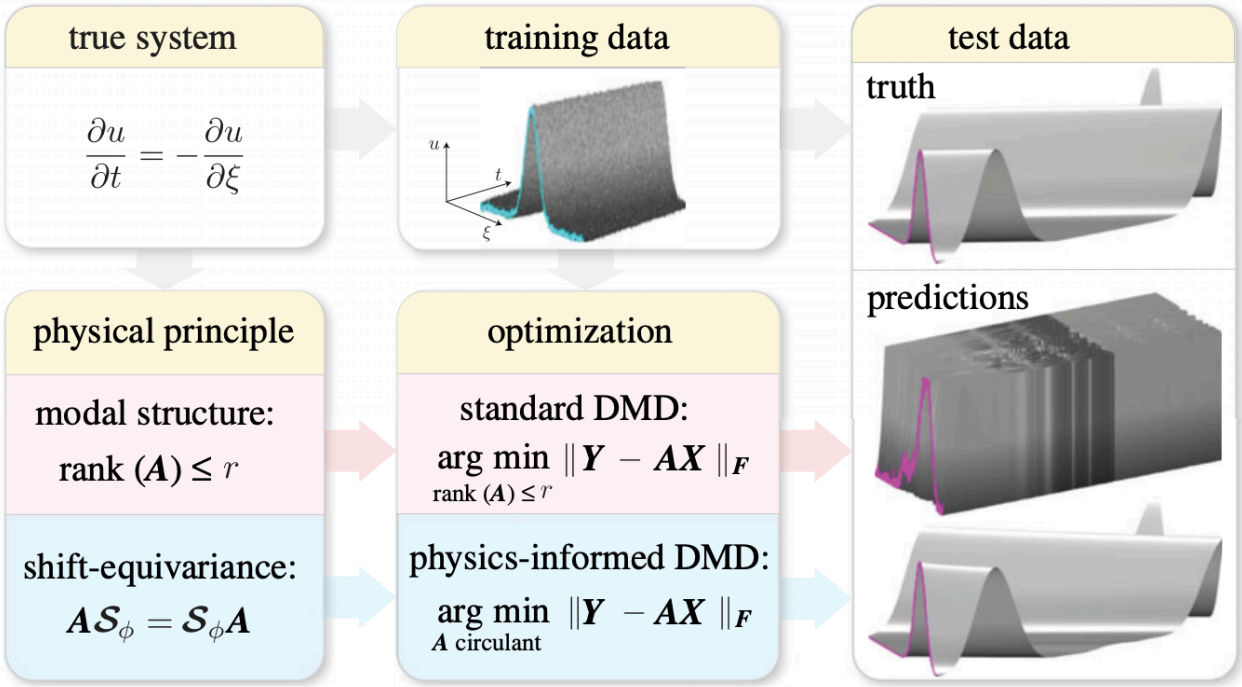


Which method should you use?

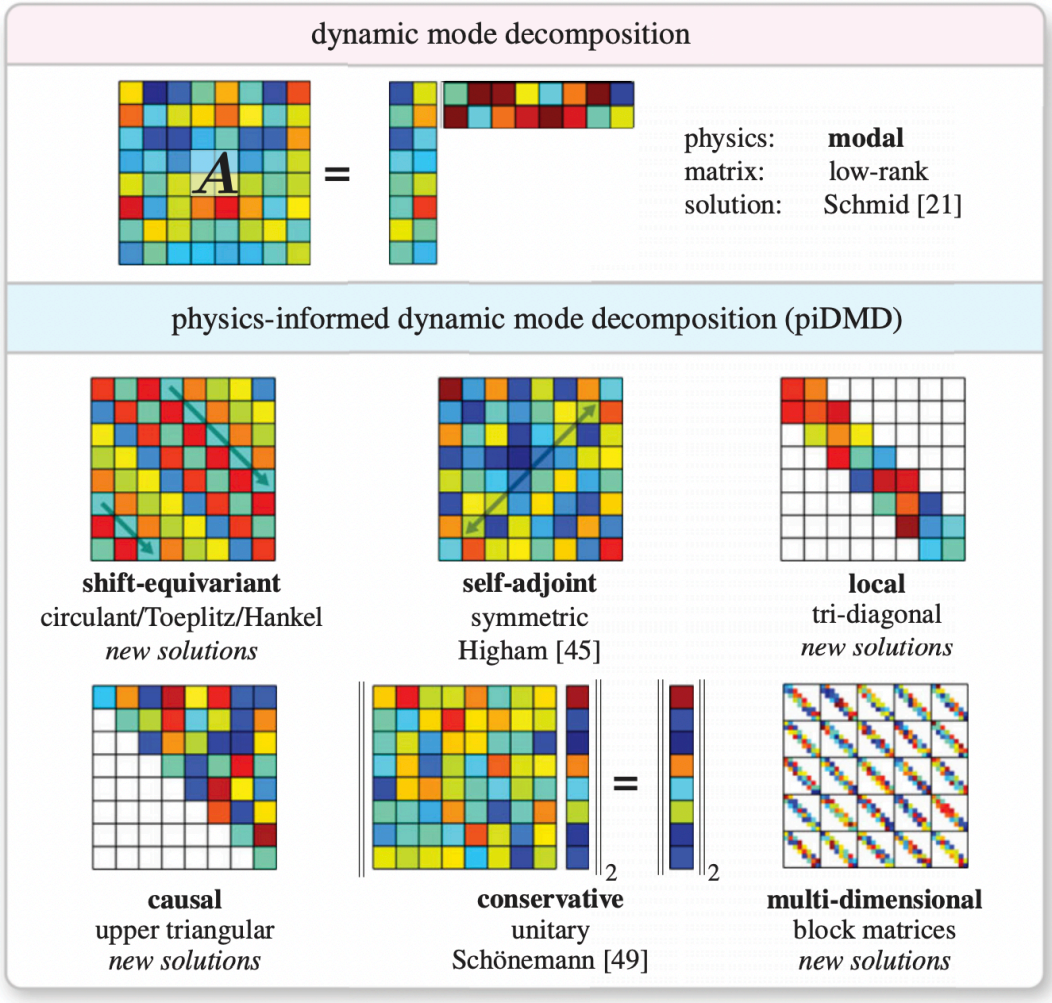


Physics-informed DMD

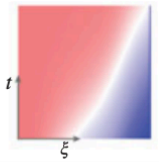
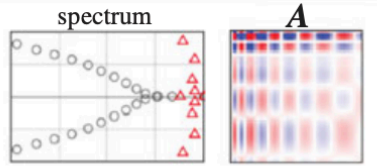
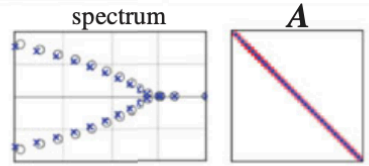
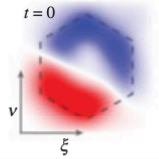
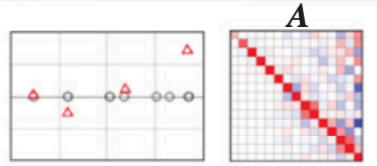
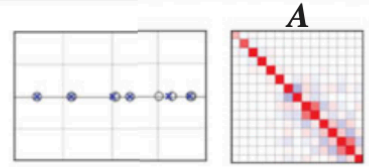
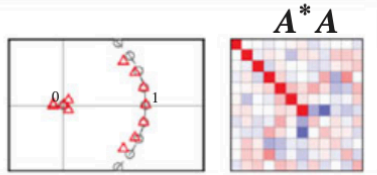
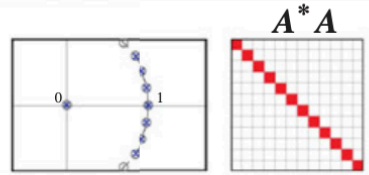
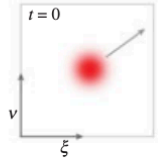
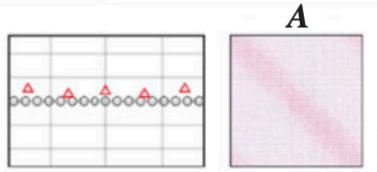
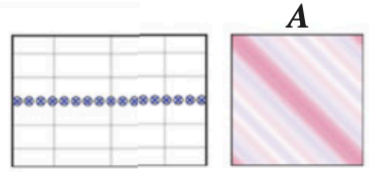
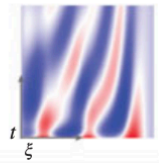
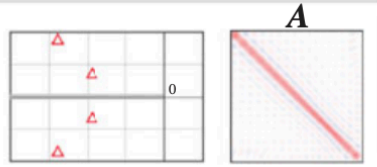
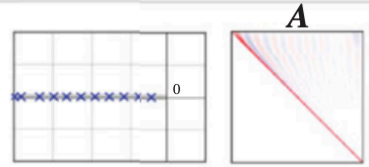
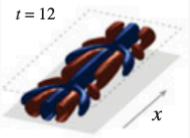
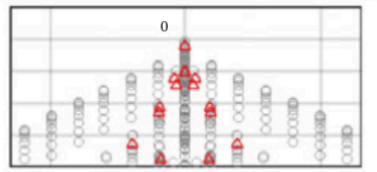
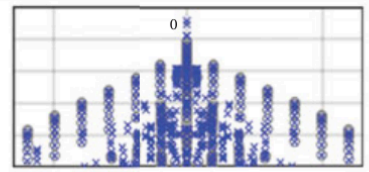
class **PiDMD**(*manifold, manifold_opt=None, compute_A=False, svd_rank=-1, tlsq_rank=0, opt=False*)



Baddoo et al., Proc. R. Soc. A, 2023



Different system models result in different matrix structures you can impose

system	physics (*matrix structure*)	data	standard DMD	physics-informed DMD
advection-diffusion $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial \xi^2} - v(\xi)\frac{\partial u}{\partial \xi}$	local (*tri-diagonal*)			
Schrödinger equation $i\hbar\frac{d}{dt}\lvert\psi(t)\rangle = \hat{H}\lvert\psi(t)\rangle$	self-adjoint (*Hermitian*)			
cylinder flow $\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = \frac{1}{Re}\nabla^2\boldsymbol{u} - \nabla p$	conservative (*unitary*)			
two-dimensional advection $\frac{\partial u}{\partial t} = -\boldsymbol{c}\cdot\nabla u$	shift-equivariant (*block circulant*)			
Volterra integro-differential equation $\frac{\partial u}{\partial t} = \int_{-1}^{\xi} K(\xi, v)\, u\,(v, t)\, dv$	causal (*upper triangular*)			
channel flow $\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = \frac{1}{Re}\nabla^2\boldsymbol{u} - \nabla p$	shift-equivariant (*block circulant*)			

Baddoo et al., 2023

Code documentation

About

Python Dynamic Mode Decomposition

pydmd.github.io/PyDMD/

- DMDBase
- DMD Operator
- DMD
- BOPDMD: Optimized DMD and Bagging, Optimized DMD
- Compressed DMD
- DMD Modes Tuner
- DMD with control
- Kernelized extended DMD
- Forward-backward DMD
- Hankel DMD
- Havok
- Higher-order DMD
- Multi-resolution DMD

- OptDMD
- Parametric DMD
- Plotter
- Physics-informed DMD
- Randomized DMD
- Sparsity-promoting DMD
- Subspace DMD
- VarProDMD: Variable Projection for DMD
- Utilities

Tutorials

Tutorials for the users (almost 20)

Name	Description	PyDMD used classes
Tutorial1 [.ipynb, .py, .html]	Analyzing real, simple data sets with PyDMD	`pydmd.DMD` , `pydmd.BOPDMD`
Tutorial2 [.ipynb, .py, .html]	advanced features of standard DMD	`pydmd.DMD`
Tutorial3 [.ipynb, .py, .html]	multi-resolution DMD for transient phenomena	`pydmd.MrDMD`
Tutorial4 [.ipynb, .py, .html]	compress DMD for computation speedup	`pydmd.CDMD`

...

Tutorial for the developers

Name	Description	PyDMD used classes
Tutorial1 [.ipynb, .py, .html]	implementing a new version of DMD	`pydmd.DMDBase`