NTNU

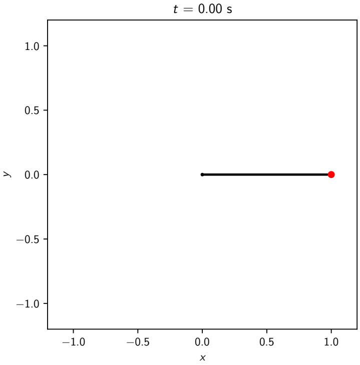
Norwegian University of Science and Technology

SINDy – a survey of methods and their properties

Davide Murari

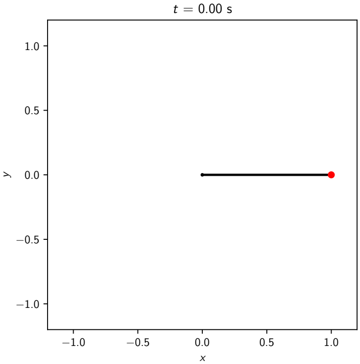
Trial lecture
September 25, 2024

Data-driven discovery of dynamical systems



We want to find the differential equation $\dot{\boldsymbol{x}}(t) = X(\boldsymbol{x}(t))$, $X : \mathbb{R}^2 \to \mathbb{R}^2$, generating the trajectory in the movie.

Data-driven discovery of dynamical systems



We want to find the differential equation $\dot{\boldsymbol{x}}(t) = X(\boldsymbol{x}(t))$, $X : \mathbb{R}^2 \to \mathbb{R}^2$, generating the trajectory in the movie.

Outline of the procedure

We define

$$\begin{cases} \dot{x} = \sum_{i=1}^{N_x} \lambda_i f_i(x,y) \\ \dot{y} = \sum_{j=1}^{N_y} \mu_j g_j(x,y), \end{cases} \tag{1}$$

for a set of functions $f_i, g_j : \mathbb{R}^2 \to \mathbb{R}$, and look for a *good* set of coefficients λ_i, μ_j making (1) an accurate approximation of $\dot{\boldsymbol{x}}(t) = X(\boldsymbol{x}(t))$.

Sparse Identification of Nonlinear Dynamics (SINDy)

Motivation behind SINDy

The right-hand side of most differential equations is made of the sum of a few functions, so the coefficients λ_i, μ_j in the linear combination should be, in large part, set to zero.

Sparse Identification of Nonlinear Dynamics (SINDy)

Motivation behind SINDy

The right-hand side of most differential equations is made of the sum of a few functions, so the coefficients λ_i, μ_j in the linear combination should be, in large part, set to zero.

Some examples:

- Simple pendulum: $\dot{x} = y$, $\dot{y} = -g/L\sin(x)$,
- Lorenz: $\dot{x} = \sigma(y - x)$, $\dot{y} = x(\rho - z) - y$, $\dot{z} = xy - \beta z$,
- Free rigid body:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & x_3/I_3 & -x_2/I_2 \\ -x_3/I_3 & 0 & x_1/I_1 \\ x_2/I_2 & -x_1/I_1 & 0 \end{bmatrix} \boldsymbol{x}.$$

Sparse Identification of Nonlinear Dynamics (SINDy)

The algorithm to approximate $X : \mathbb{R}^d \to \mathbb{R}^d$

1. Build the data and derivative matrices

$$U = \begin{bmatrix} \boldsymbol{x}(t_1) & \cdots & \boldsymbol{x}(t_m) \end{bmatrix}^\top, \; U_p = \begin{bmatrix} \dot{\boldsymbol{x}}(t_1) & \cdots & \dot{\boldsymbol{x}}(t_m) \end{bmatrix}^\top \in \mathbb{R}^{m \times d}.$$

Sparse Identification of Nonlinear Dynamics (SINDy)

The algorithm to approximate $X : \mathbb{R}^d \to \mathbb{R}^d$

1. Build the data and derivative matrices

$$U = \begin{bmatrix} \boldsymbol{x}(t_1) & \cdots & \boldsymbol{x}(t_m) \end{bmatrix}^\top, \quad U_p = \begin{bmatrix} \dot{\boldsymbol{x}}(t_1) & \cdots & \dot{\boldsymbol{x}}(t_m) \end{bmatrix}^\top \in \mathbb{R}^{m \times d}.$$

2. Choose $f_1, \ldots, f_N : \mathbb{R}^d \to \mathbb{R}$ that are likely to appear in X, and define the matrix $\Theta(U) \in \mathbb{R}^{m \times N}$ with entries

$$\Theta(U)_{i,j} = f_j(\boldsymbol{x}(t_i)), \quad i = 1, \ldots, m, \; j = 1, \ldots, N.$$

Sparse Identification of Nonlinear Dynamics (SINDy)

The algorithm to approximate $X : \mathbb{R}^d \to \mathbb{R}^d$

1. Build the data and derivative matrices

$$U = \begin{bmatrix} \boldsymbol{x}(t_1) & \cdots & \boldsymbol{x}(t_m) \end{bmatrix}^\top, \; U_p = \begin{bmatrix} \dot{\boldsymbol{x}}(t_1) & \cdots & \dot{\boldsymbol{x}}(t_m) \end{bmatrix}^\top \in \mathbb{R}^{m \times d}.$$

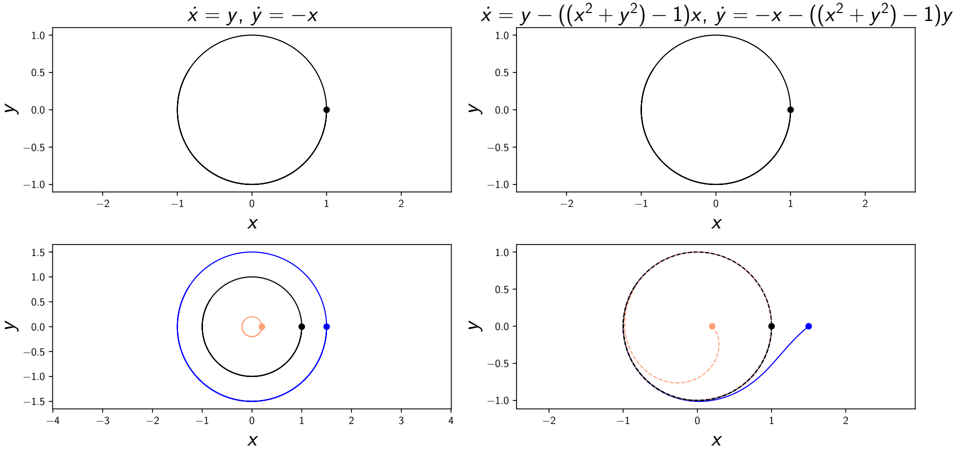
2. Choose $f_1, \ldots, f_N : \mathbb{R}^d \to \mathbb{R}$ that are likely to appear in X, and define the matrix $\Theta(U) \in \mathbb{R}^{m \times N}$ with entries

$$\Theta(U)_{i,j} = f_j(\boldsymbol{x}(t_i)), \; i = 1, \ldots, m, \; j = 1, \ldots, N.$$

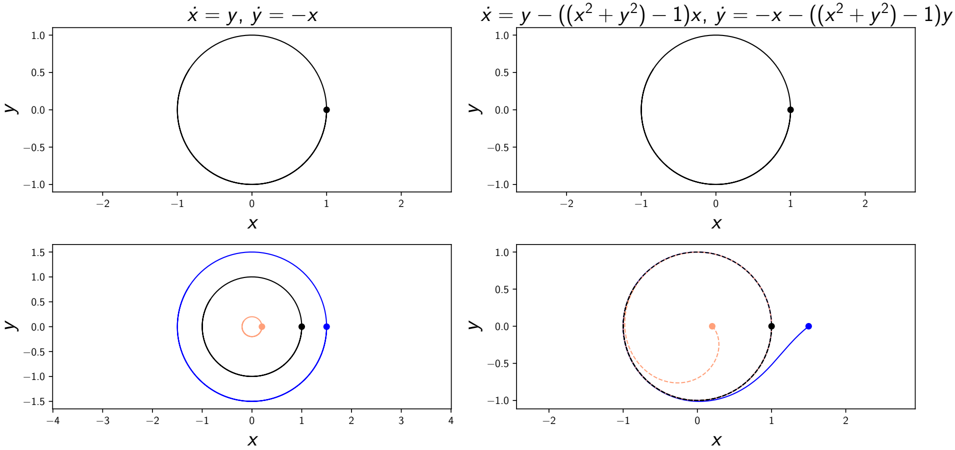
3. Solve

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \left\| U_p - \Theta(U)\Sigma \right\|_F^2 + \lambda \left\| \mathrm{vec}(\Sigma) \right\|_1, \; \lambda > 0.$$

Building the data matrix



Building the data matrix



- The data matrix $U \in \mathbb{R}^{m \times d}$ collects the snapshots of some observed trajectories at different time instants $t_1, \dots, t_m$. Typically $d \ll m$.

Building the derivative matrix

- We generally do not know the exact values of $\dot{\boldsymbol{x}}(t_i)$, i.e., of $X(\boldsymbol{x}(t_i))$, so we need to approximate them to assemble $U_p \in \mathbb{R}^{m\times d}$.

Building the derivative matrix

- We generally do not know the exact values of $\dot{\boldsymbol{x}}(t_i)$, i.e., of $X(\boldsymbol{x}(t_i))$, so we need to approximate them to assemble $U_p \in \mathbb{R}^{m\times d}$.

- Approximating the derivatives is a delicate step that could amplify the noise present in the trajectory data.

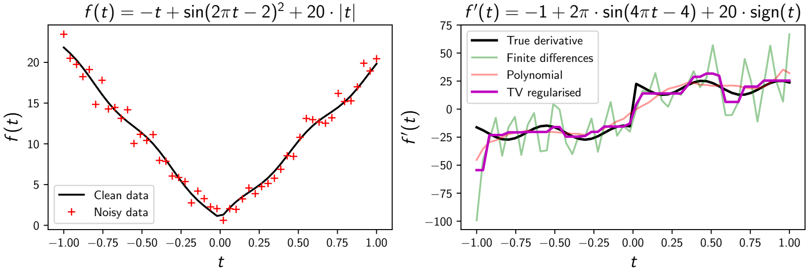


Figure: Results obtained with the `PySINDy`[1] library.

[1] Alan A Kaptanoglu et al. "PySINDy: A comprehensive Python package for robust sparse system identification". In: *arXiv preprint arXiv:2111.08481* (2021).

Total Variation Regularised Derivative

- Let $[t_1, t_m] \ni t \mapsto x(t) \in \mathbb{R}$ be a signal with derivative $u(t)$.
- Consider a vector $\boldsymbol{s} \in \mathbb{R}^m$ made of noisy entries $s_i = x(t_i) + \delta_i$.

Total Variation Regularised Derivative

- Let $[t_1, t_m] \ni t \mapsto x(t) \in \mathbb{R}$ be a signal with derivative $u(t)$.
- Consider a vector $\boldsymbol{s} \in \mathbb{R}^m$ made of noisy entries $s_i = x(t_i) + \delta_i$.
- The TV regularised derivative based on $\boldsymbol{s} \in \mathbb{R}^m$ is defined as

$$\underset{\boldsymbol{u}\in\mathbb{R}^m}{\arg\min}\, F(\boldsymbol{u}) := \frac{1}{2}\left\|A\boldsymbol{u} - (\boldsymbol{s} - s_1)\right\|_2^2 + \alpha \left\|D\boldsymbol{u}\right\|_1 .$$

The matrix A contains quadrature weights, so

$$(A\boldsymbol{u})_i \approx \int_{t_1}^{t_i} u(t)\mathrm{d}t,$$

while D is a finite differences matrix of the first order, so

$$(D\boldsymbol{u})_i \approx \dot{u}(t_i).$$

Building a library of candidate functions

- Many dynamical systems are well approximated by polynomial differential equations.
- Multivariate polynomials are usually the first reasonable set of functions one can test in the dictionary of candidate functions.

Building a library of candidate functions

- Many dynamical systems are well approximated by polynomial differential equations.
- Multivariate polynomials are usually the first reasonable set of functions one can test in the dictionary of candidate functions.
- For example, if we consider polynomials up to degree 2 for a system in $\mathbb{R}^2$, we would have

$$\Theta(U) = \begin{bmatrix} 1 & x(t_1) & y(t_1) & x(t_1)^2 & x(t_1)y(t_1) & y(t_1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x(t_m) & y(t_m) & x(t_m)^2 & x(t_m)y(t_m) & y(t_m)^2 \end{bmatrix} \in \mathbb{R}^{m\times 6}.$$

Building a library of candidate functions

- Many dynamical systems are well approximated by polynomial differential equations.
- Multivariate polynomials are usually the first reasonable set of functions one can test in the dictionary of candidate functions.
- For example, if we consider polynomials up to degree 2 for a system in $\mathbb{R}^2$, we would have

$$\Theta(U) = \begin{bmatrix} 1 & x(t_1) & y(t_1) & x(t_1)^2 & x(t_1)y(t_1) & y(t_1)^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x(t_m) & y(t_m) & x(t_m)^2 & x(t_m)y(t_m) & y(t_m)^2 \end{bmatrix} \in \mathbb{R}^{m\times 6}.$$

- Another common set of functions are trigonometric functions, for example in the pendulum $\ddot{x} = -g/L\sin(x)$. Thus, one can augment the polynomial dictionary with functions like $\sin(kx)$, $k \in \mathbb{Z}$.

Least squares with sparsity promotion

- We now need to find how to linearly combine the columns of $\Theta(U)$ to recover U_p, with a sparse set of coefficients.

Least squares with sparsity promotion

- We now need to find how to linearly combine the columns of $\Theta(U)$ to recover U_p, with a sparse set of coefficients.
- A first strategy to do so is ℓ^1 regularisation, leading to the (convex) unconstrained minimisation problem

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \left\| U_p - \Theta(U)\Sigma \right\|_F^2 + \lambda \left\| \mathrm{vec}(\Sigma) \right\|_1 , \ \ \lambda > 0,$$

or, equivalently, to the inequality-constrained problem

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \left\| U_p - \Theta(U)\Sigma \right\|_F^2 , \ \ \text{s.t.} \ \ \left\| \mathrm{vec}(\Sigma) \right\|_1 < \mathrm{tol}.$$

Least squares with sparsity promotion

- We now need to find how to linearly combine the columns of $\Theta(U)$ to recover U_p, with a sparse set of coefficients.
- A first strategy to do so is ℓ^1 regularisation, leading to the (convex) unconstrained minimisation problem

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \left\| U_p - \Theta(U)\Sigma \right\|_F^2 + \lambda \left\| \mathrm{vec}(\Sigma) \right\|_1 , \quad \lambda > 0,$$

  or, equivalently, to the inequality-constrained problem

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \left\| U_p - \Theta(U)\Sigma \right\|_F^2 , \quad \text{s.t.} \quad \left\| \mathrm{vec}(\Sigma) \right\|_1 < \mathrm{tol}.$$

- This method can be expensive, especially for high-dimensional datasets.

The sequential thresholded least squares method

- The alternative approach recommended in the original paper[2] is the *Sequential Thresholded Least Squares method* (STLS).

[2] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.

The sequential thresholded least squares method

- The alternative approach recommended in the original paper[2] is the *Sequential Thresholded Least Squares method* (STLS).

Sequential thresholded least squares method

1. Solve the least squares problem

$$\Sigma^0 := \underset{\Sigma \in \mathbb{R}^{N \times d}}{\arg\min} \, \| U_p - \Theta(U)\Sigma \|_F^2 \, .$$

The sequential thresholded least squares method

- The alternative approach recommended in the original paper[2] is the *Sequential Thresholded Least Squares method* (STLS).

Sequential thresholded least squares method

1. Solve the least squares problem

$$\Sigma^0 := \operatorname*{arg\,min}_{\Sigma \in \mathbb{R}^{N \times d}} \| U_p - \Theta(U)\Sigma \|_F^2 .$$

2. For $k = 1, \ldots, K$ solve the constrained least squares problem

$$\Sigma^k := \operatorname*{arg\,min}_{\Sigma \in \mathbb{R}^{N \times d}} \| U_p - \Theta(U)\Sigma \|_F^2$$
$$\text{s.t. } \Sigma_{i,j} = 0 \text{ whenever } \left| \Sigma_{i,j}^{k-1} \right| < \lambda.$$

[2] Brunton, Proctor, and Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems".

STLS convergence properties[3]

$$F(\Sigma) = \|U_p - \Theta(U)\Sigma\|_F^2 + \lambda^2 \, \|\mathrm{vec}(\Sigma)\|_0 \,. \qquad (2)$$

[3] Linan Zhang and Hayden Schaeffer. “On the convergence of the SINDy algorithm”. In: *Multiscale Modeling & Simulation* 17.3 (2019), pp. 948–972.

STLS convergence properties[3]

$$F(\Sigma) = \|U_p - \Theta(U)\Sigma\|_F^2 + \lambda^2 \, \|\mathrm{vec}(\Sigma)\|_0 \, . \tag{2}$$

Convergence theorem

Suppose that $\|\Theta(U)\|_2 = 1$.

1. The STLS iterates $\{\Sigma^k\}$ converge to a fixed point in at most $N \cdot d$ steps.

STLS convergence properties[3]

$$F(\Sigma) = \|U_p - \Theta(U)\Sigma\|_F^2 + \lambda^2 \; \|\mathrm{vec}(\Sigma)\|_0 \,. \tag{2}$$

Convergence theorem

Suppose that $\|\Theta(U)\|_2 = 1$.

1. The STLS iterates $\{\Sigma^k\}$ converge to a fixed point in at most $N \cdot d$ steps.
2. A fixed point of the STLS method is a local minimiser of (2).

STLS convergence properties[3]

$$F(\Sigma) = \|U_p - \Theta(U)\Sigma\|_F^2 + \lambda^2 \, \|\mathrm{vec}(\Sigma)\|_0 \, . \qquad (2)$$

Convergence theorem

Suppose that $\|\Theta(U)\|_2 = 1$.

1. The STLS iterates $\{\Sigma^k\}$ converge to a fixed point in at most $N \cdot d$ steps.
2. A fixed point of the STLS method is a local minimiser of (2).
3. A global minimiser of (2) is a fixed point of the scheme.

STLS convergence properties[3]

$$F(\Sigma) = \|U_p - \Theta(U)\Sigma\|_F^2 + \lambda^2 \, \|\mathrm{vec}(\Sigma)\|_0 \,. \tag{2}$$

Convergence theorem

Suppose that $\|\Theta(U)\|_2 = 1$.

1. The STLS iterates $\{\Sigma^k\}$ converge to a fixed point in at most $N \cdot d$ steps.
2. A fixed point of the STLS method is a local minimiser of (2).
3. A global minimiser of (2) is a fixed point of the scheme.
4. The iterates $\{\Sigma^k\}$ strictly decrease (2) unless stationary.

[3] Zhang and Schaeffer, "On the convergence of the SINDy algorithm".

Example: Simple harmonic oscillator

The target equations are

$$\begin{cases} \dot{x}(t) = y(t) \\ \dot{y}(t) = -0.5\,x(t). \end{cases}$$

Result obtained with **LASSO**, fixing $\lambda = 10^{-3}$ and exact derivatives $\dot{\boldsymbol{x}}(t_i)$:

$$\begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{array}\begin{array}{c} \begin{array}{cc} \quad\dot{x}\quad & \quad\dot{y}\quad \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & -0.4996 \\ 0.9991 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array}.$$

Result obtained with **STLS**, fixing $\lambda = 0.05$ and exact derivatives $\dot{\boldsymbol{x}}(t_i)$:

$$\begin{array}{c} \\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{array}\begin{array}{c} \begin{array}{cc} \dot{x} & \quad\dot{y} \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & -0.5 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \end{array}.$$

Example with noisy data

Target differential equations: $\begin{cases} \dot{x} = -0.1x + 2y \\ \dot{y} = -2x - 0.1y \end{cases}.$

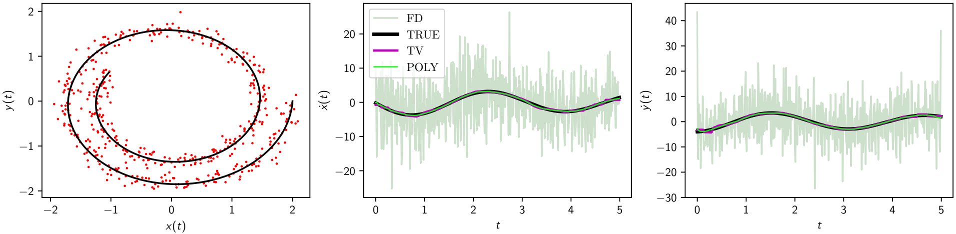


Figure: Gaussian noise with $\sigma = 0.1$. STLS algorithm with $\lambda = 0.05$.

POLY: $\begin{cases} \dot{x} = -0.082x + 1.975y \\ \dot{y} = -1.972x - 0.110y \end{cases}$, TV: $\begin{cases} \dot{x} = -0.092x + 1.974y \\ \dot{y} = -1.981x - 0.107y. \end{cases}$

Constraining the coefficients

- Suppose we know we are dealing with a planar Hamiltonian system of the form
$$\begin{cases} \dot{x} = y \\ \dot{y} = -V'(x), \end{cases} \tag{3}$$
where we do not know the potential energy $V : \mathbb{R} \to \mathbb{R}$.

- Then we could constrain the optimisation problem further, for example saying that there is no term in y in the second equation.

Constraining the coefficients

- Suppose we know we are dealing with a planar Hamiltonian system of the form

$$\begin{cases} \dot{x} = y \\ \dot{y} = -V'(x), \end{cases} \tag{3}$$

  where we do not know the potential energy $V : \mathbb{R} \to \mathbb{R}$.

- Then we could constrain the optimisation problem further, for example saying that there is no term in y in the second equation.

- The same might occur when we know part of the terms on the right-hand side, conservation laws, or symmetries in the equations.

Constraining the coefficients

- Suppose we know we are dealing with a planar Hamiltonian system of the form
$$\begin{cases} \dot{x} = y \\ \dot{y} = -V'(x), \end{cases} \tag{3}$$
where we do not know the potential energy $V : \mathbb{R} \to \mathbb{R}$.
- Then we could constrain the optimisation problem further, for example saying that there is no term in y in the second equation.
- The same might occur when we know part of the terms on the right-hand side, conservation laws, or symmetries in the equations.
- To see how to impose the structure in (3), we first rewrite the SINDy method in vector form.

Vector version of SINDy

- We use the vec operator, which stacks the columns of a matrix into a single column vector:

$$\mathrm{vec}\left(\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_k \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{a}_1^\top & \cdots \boldsymbol{a}_k^\top \end{bmatrix}^\top .$$

Vector version of SINDy

- We use the vec operator, which stacks the columns of a matrix into a single column vector:

$$\mathrm{vec}\left(\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_k \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{a}_1^\top & \cdots \boldsymbol{a}_k^\top \end{bmatrix}^\top.$$

This operator also satisfies $\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$, and hence

$$\mathrm{vec}(\Theta(U)\Sigma) = (I_d \otimes \Theta(U))\,\mathrm{vec}(\Sigma) =: \widetilde{\Theta}(U)\boldsymbol{\sigma} \in \mathbb{R}^{m\cdot d}.$$

More explicitly, $\widetilde{\Theta}(U)$ is of the form

$$\widetilde{\Theta}(U) = \begin{bmatrix} \Theta(U) & 0 & \cdots & 0 \\ 0 & \Theta(U) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Theta(U) \end{bmatrix}.$$

Vector version of SINDy

- We use the vec operator, which stacks the columns of a matrix into a single column vector:

$$\mathrm{vec}\left(\begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_k \end{bmatrix}\right) = \begin{bmatrix} \boldsymbol{a}_1^\top & \cdots \boldsymbol{a}_k^\top \end{bmatrix}^\top .$$

This operator also satisfies $\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$. So,

$$\mathrm{vec}(\Theta(U)\Sigma) = (I_d \otimes \Theta(U))\,\mathrm{vec}(\Sigma) =: \widetilde{\Theta}(U)\boldsymbol{\sigma} \in \mathbb{R}^{m \cdot d}.$$

- Since $\|A\|_F = \|\mathrm{vec}(A)\|_2$, the LASSO formulation can be rewritten as

$$\text{Find } \underset{\boldsymbol{\sigma} \in \mathbb{R}^{N \cdot d}}{\arg\min} \left\| \widetilde{\Theta}(U)\boldsymbol{\sigma} - \boldsymbol{u}_p \right\|_2^2 + \lambda \left\| \boldsymbol{\sigma} \right\|_1 ,$$

where $\boldsymbol{u}_p := \mathrm{vec}(U_p)$.

The constrained STLS algorithm[4]

- With the vector notation, one of the STLS iterates is of the form

$$\boldsymbol{\sigma}^k := \underset{\boldsymbol{\sigma}\in\mathbb{R}^{N\cdot d}}{\arg\min} \left\|\widetilde{\Theta}(U)\boldsymbol{\sigma} - \boldsymbol{u}_p\right\|_2^2$$
$$\text{s.t. } C^k\boldsymbol{\sigma} = \boldsymbol{d}^k, \quad C^k \in \mathbb{R}^{r_k\times N\cdot d},$$

which admits a unique solution if

$$\mathrm{rank}(C^k) = r_k, \text{ and } \mathrm{rank}\left(\begin{bmatrix}\widetilde{\Theta}(U)\\ C^k\end{bmatrix}\right) = N\cdot d.$$

[4] Jean-Christophe Loiseau and Steven L Brunton. “Constrained sparse Galerkin regression”. In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67.

NTNU

Back to planar Hamiltonian systems...

- Say that we want to discover $\ddot{x} = -V'(x)$ with $V(x) = x^2/4$. We can then include prior information as $\widetilde{C}\boldsymbol{\sigma} = \widetilde{\boldsymbol{d}}$ where

$$\widetilde{C} = \begin{array}{c} \begin{array}{cccccc} \multicolumn{3}{c}{\dot{x}} & \multicolumn{3}{c}{\dot{y}} \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ \begin{array}{cccccc} 1 & x & y & 1 & x & y \end{array} \end{array}, \quad \widetilde{\boldsymbol{d}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Back to planar Hamiltonian systems...

- Say that we want to discover $\ddot{x} = -V'(x)$ with $V(x) = x^2/4$. We can then include prior information as $\widetilde{C}\boldsymbol{\sigma} = \widetilde{\boldsymbol{d}}$ where

$$\widetilde{C} = \begin{array}{c} \begin{array}{cccccc} & \dot{x} & & & \dot{y} & \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\ \begin{array}{cccccc} 1 & x & y & 1 & x & y \end{array} \end{array}, \quad \widetilde{\boldsymbol{d}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

- At each step, we can then solve

$$\boldsymbol{\sigma}^k := \underset{\boldsymbol{\sigma} \in \mathbb{R}^{N \cdot d}}{\arg\min} \left\| \widetilde{\Theta}(U)\boldsymbol{\sigma} - \boldsymbol{u}_p \right\|_2^2$$
$$\text{s.t.} \begin{bmatrix} C^k \\ \widetilde{C} \end{bmatrix} \boldsymbol{\sigma} = \begin{bmatrix} \boldsymbol{d}^k \\ \widetilde{\boldsymbol{d}} \end{bmatrix}.$$

Constraining the model in the presence of noise

We now perturb the exact derivatives $\dot{\boldsymbol{x}}(t_i)$ to $\boldsymbol{v}_i = \dot{\boldsymbol{x}}(t_i) + \varepsilon$ with $\varepsilon_k \sim \mathcal{N}(0, \sigma^2)$, $k = 1, \dots, d$, and see how the reconstructed models are.

The target equations are $\dot{x} = y$, $\dot{y} = -0.5x$.

Constraining the model in the presence of noise

We now perturb the exact derivatives $\dot{\boldsymbol{x}}(t_i)$ to $\boldsymbol{v}_i = \dot{\boldsymbol{x}}(t_i) + \varepsilon$ with $\varepsilon_k \sim \mathcal{N}(0, \sigma^2)$, $k = 1, \dots, d$, and see how the reconstructed models are.

The target equations are $\dot{x} = y$, $\dot{y} = -0.5x$.

The orange matrices are obtained with constrained models, while the blue ones are unconstrained:

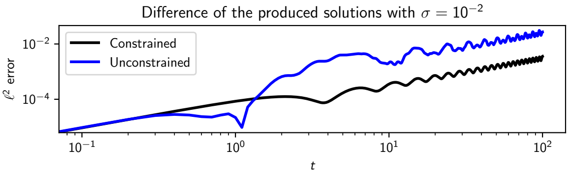
$$
\begin{array}{c}
\\ 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \\ \\
\end{array}
\overset{\sigma = 10^{-3}}{\begin{bmatrix} 0 & 0 \\ 0 & -0.500 \\ 1.000 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\begin{array}{cc}\dot{x} & \dot{y}\end{array}},
\overset{\sigma = 10^{-3}}{\begin{bmatrix} 0 & 0 \\ 0 & -0.500 \\ 1.000 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\begin{array}{cc}\dot{x} & \dot{y}\end{array}},
\overset{\sigma = 10^{-2}}{\begin{bmatrix} 0 & 0 \\ 0 & -0.500 \\ 1.000 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}}_{\begin{array}{cc}\dot{x} & \dot{y}\end{array}},
\overset{\sigma = 10^{-2}}{\begin{bmatrix} 0 & 0.112 \\ 0 & -0.500 \\ 1.000 & 0 \\ 0 & -0.112 \\ 0 & 0 \\ 0 & -0.223 \end{bmatrix}}_{\begin{array}{cc}\dot{x} & \dot{y}\end{array}}.
$$

Analysis of the recovered dynamics

$$\Sigma = \begin{bmatrix} 0 & 0.112 \\ 0 & -0.500 \\ 1.000 & 0 \\ 0 & -0.112 \\ 0 & 0 \\ 0 & -0.223 \end{bmatrix} \begin{matrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{matrix} \implies \begin{cases} \dot{x} = y \\ \dot{y} = -0.5x + c(1 - x^2 - 2y^2), \;\; c \approx 0.112. \end{cases}$$

The additional term vanishes on the energy level set of the initial condition $\boldsymbol{x}_0 = [1, 0]$, which is the ellipse

$$\left\{(x, y) \in \mathbb{R}^2 : \;\; H(x, y) = y^2/2 + x^2/4 = 1/4\right\}.$$



SINDy for discrete dynamical systems

- What if we want to approximate a map $F : \mathbb{R}^d \to \mathbb{R}^d$ defining the discrete dynamics $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k)$?

SINDy for discrete dynamical systems

- What if we want to approximate a map $F : \mathbb{R}^d \to \mathbb{R}^d$ defining the discrete dynamics $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k)$?
- In this case, we do not need the derivative matrix U_p, but we work with the dataset

$$U_l = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_m \end{bmatrix}^\top, \; U_r = \begin{bmatrix} \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_{m+1} \end{bmatrix}^\top \in \mathbb{R}^{m \times d}.$$

SINDy for discrete dynamical systems

- What if we want to approximate a map $F : \mathbb{R}^d \to \mathbb{R}^d$ defining the discrete dynamics $\boldsymbol{x}_{k+1} = F(\boldsymbol{x}_k)$?
- In this case, we do not need the derivative matrix U_p, but we work with the dataset

$$U_l = \begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_m \end{bmatrix}^\top, \; U_r = \begin{bmatrix} \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_{m+1} \end{bmatrix}^\top \in \mathbb{R}^{m \times d}.$$

- We can still apply the same procedure as SINDy for continuous systems, but to these new data matrices:

$$\min_{\Sigma \in \mathbb{R}^{N \times d}} \|U_r - \Theta(U_l)\Sigma\|_F^2 + \lambda \|\mathrm{vec}(\Sigma)\|_1, \; \lambda > 0.$$

SINDy for parametric differential equations

- What we have seen up to now extends to dynamical systems that depend on a parameter $\boldsymbol{\mu} \in \mathbb{R}^p$.
- We can rewrite
$$\dot{\boldsymbol{x}} = X(\boldsymbol{x}, \boldsymbol{\mu})$$
as
$$\begin{cases} \dot{\boldsymbol{x}} = X(\boldsymbol{x}, \boldsymbol{\mu}) \\ \dot{\boldsymbol{\mu}} = 0. \end{cases} \tag{4}$$
- SINDy can then be applied to (4) using the new state variable
$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\mu} \end{bmatrix}.$$

NTNU

SINDy for non-autonomous differential equations

- A similar reasoning applies to explicitly time-dependent differential equations.
- We can rewrite

$$\dot{\boldsymbol{x}} = X(\boldsymbol{x}, t)$$

as

$$\begin{cases} \dot{\boldsymbol{x}} = X(\boldsymbol{x}, t) \\ \dot{t} = 1. \end{cases} \tag{5}$$

- SINDy can then be applied to (5) using the new state variable

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ t \end{bmatrix}.$$

Some limitations and extensions of SINDy

Curse of dimensionality[5]

As the dimension d grows, the set of basis functions one has to consider will grow quickly. For example $\dim(\mathbb{P}_k^d) = \binom{k+d}{d}$, which for $d = 6$ and $k = 5$ is already 462.

[5] Kathleen Champion et al. "Data-driven discovery of coordinates and governing equations". In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451; Brunton, Proctor, and Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems".

Curse of dimensionality[5]

As the dimension d grows, the set of basis functions one has to consider will grow quickly. For example $\dim(\mathbb{P}_k^d) = \binom{k+d}{d}$, which for $d = 6$ and $k = 5$ is already 462.

A common solution to this problem is to start with a truncated SVD:

$$U^\top \approx \Psi_r \Sigma_r V_r^\top \implies \boldsymbol{x} \approx \Psi_r \boldsymbol{a}, \quad \boldsymbol{a} \in \mathbb{R}^r.$$

Then, one can apply the SINDy algorithm in the variable $\boldsymbol{a}$, and $\dot{\boldsymbol{x}}(t) \approx \Psi_r \dot{\boldsymbol{a}}(t)$.

[5] Champion et al., "Data-driven discovery of coordinates and governing equations"; Brunton, Proctor, and Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems".

Approximating the derivatives

The SINDy algorithm depends on having an accurate approximation of the exact derivative matrix U_p.

[6] Hayden Schaeffer and Scott G McCalla. "Sparse model selection via integral terms". In: *Physical Review E* 96.2 (2017), p. 023302.

Approximating the derivatives

The SINDy algorithm depends on having an accurate approximation of the exact derivative matrix U_p.

A solution[6] can be to work with the integral version of the differential equation:

$$\boldsymbol{x}(t) - \boldsymbol{x}(0) = \int_0^t X(\boldsymbol{x}(t))\mathrm{d}t.$$

We can then proceed similarly to the SINDy algorithm and write

$$x_i(t_m) - x_i(0) \approx \sum_{j=1}^{N} \Sigma_{i,j} d_j(t_m), \quad d_j(t_m) \approx \int_0^{t_m} f_j(\boldsymbol{x}(t))\mathrm{d}t.$$

[6] Schaeffer and McCalla, "Sparse model selection via integral terms".

References

- Brunton, Steven L, Joshua L Proctor, and J Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.
- Champion, Kathleen et al. “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451.
- Kaptanoglu, Alan A et al. “PySINDy: A comprehensive Python package for robust sparse system identification”. In: *arXiv preprint arXiv:2111.08481* (2021).
- Loiseau, Jean-Christophe and Steven L Brunton. “Constrained sparse Galerkin regression”. In: *Journal of Fluid Mechanics* 838 (2018), pp. 42–67.
- Schaeffer, Hayden and Scott G McCalla. “Sparse model selection via integral terms”. In: *Physical Review E* 96.2 (2017), p. 023302.
- Zhang, Linan and Hayden Schaeffer. “On the convergence of the SINDy algorithm”. In: *Multiscale Modeling & Simulation* 17.3 (2019), pp. 948–972.

NTNU

THANK YOU FOR THE ATTENTION

Limitation: Knowledge of the terms to include in the dictionary

The quality of the recovered system depends on our knowledge of what basis functions to include in $\Theta(U)$, which can generally not be inferred just based on data.

Limitation: Knowledge of the terms to include in the dictionary

> The quality of the recovered system depends on our knowledge of what basis functions to include in $\Theta(U)$, which can generally not be inferred just based on data.

A solution[1] could be to use general enough parametric models like Neural ODEs

$$\dot{\boldsymbol{x}}(t) = \mathcal{N}_\theta(\boldsymbol{x}(t)), \;\; \theta \in \mathbb{R}^p,$$

to get a first approximation of the right-hand side. We could then do sparse regression over this approximate model to get a more interpretable approximation, as in SINDy.

[1] Christopher Rackauckas et al. "Universal differential equations for scientific machine learning"

Example in higher dimensions

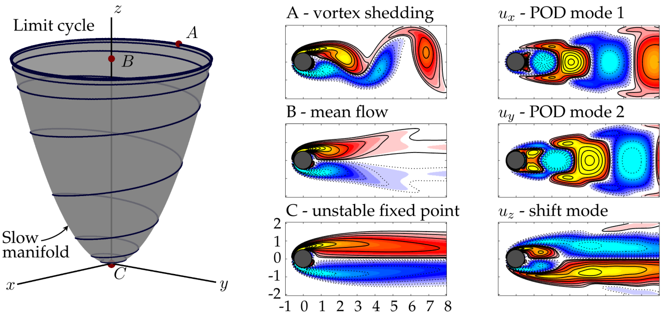
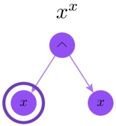


Figure: Low-rank dynamics underlying the periodic vortex shedding behind a circular cylinder at low Reynolds number, $Re = 100$.

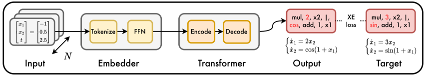
$$\begin{cases} \dot{x} = \mu x - \omega y + Axz \\ \dot{y} = \omega x + \mu y + Ayz \\ \dot{z} = -\lambda(z - x^2 - y^2). \end{cases}$$

Some alternative methods to SINDy

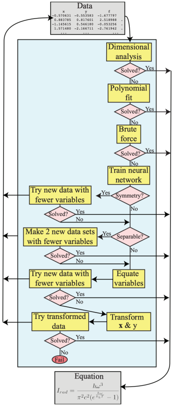
- Symbolic regression with evolutionary algorithms[1].



- Symbolic regression with transformers[2]



- Hybrid approaches, like AI Feynman[3]



[1] Miles Cranmer. “Interpretable machine learning for science with PySR and SymbolicRegression. jl”.

[2] Stéphane d’Ascoli et al. “Odeformer: Symbolic regression of dynamical systems with transformers”.

[3] Silviu-Marian Udrescu and Max Tegmark. “AI Feynman: A physics-inspired method for symbolic regression”.

PDE-FIND[1]

- Similarly to SINDy, we could discover the right-hand side of the PDE

$$\partial_t u(\boldsymbol{x}, t) = \mathcal{N}\left(u, \partial_x u, \partial_{xx} u, \ldots\right), \quad \boldsymbol{x} \in \mathbb{R}^d.$$

[1] Samuel H Rudy et al. "Data-driven discovery of partial differential equations".

PDE-FIND[1]

- Similarly to SINDy, we could discover the right-hand side of the PDE

$$\partial_t u(\boldsymbol{x}, t) = \mathcal{N}\left(u, \partial_x u, \partial_{xx} u, \ldots\right), \ \boldsymbol{x} \in \mathbb{R}^d.$$

- This time, the dataset is a vector $\boldsymbol{u} \in \mathbb{R}^{M \cdot N}$ where

$$\boldsymbol{u} = \mathrm{vec}(U), \ U_{n,m} \approx u(\boldsymbol{x}_n, t_m), \ n = 1, \ldots, N, \ m = 1, \ldots, M,$$

for a spatio-temporal grid $\{(\boldsymbol{x}_n, t_m)\}$ of $\Omega \times [0, T]$, $\Omega \subset \mathbb{R}^d$.

[1] Samuel H Rudy et al. "Data-driven discovery of partial differential equations".

PDE-FIND[1]

- Similarly to SINDy, we could discover the right-hand side of the PDE

$$\partial_t u(\boldsymbol{x}, t) = \mathcal{N}\left(u, \partial_x u, \partial_{xx} u, \ldots\right), \;\; \boldsymbol{x} \in \mathbb{R}^d.$$

- This time, the dataset is a vector $\boldsymbol{u} \in \mathbb{R}^{M \cdot N}$ where

$$\boldsymbol{u} = \mathrm{vec}(U), \;\; U_{n,m} \approx u(\boldsymbol{x}_n, t_m), \;\; n = 1, \ldots, N, \;\; m = 1, \ldots, M,$$

  for a spatio-temporal grid $\{(\boldsymbol{x}_n, t_m)\}$ of $\Omega \times [0, T]$, $\Omega \subset \mathbb{R}^d$.
- The candidate matrix becomes

$$\Theta(U) = \begin{bmatrix} 1 & \boldsymbol{u} & \boldsymbol{u}_x & \boldsymbol{u} \odot \boldsymbol{u}_x & \cdots \end{bmatrix} \in \mathbb{R}^{N \cdot M \times K}$$

  and we have to deal with a sparse regression of the form

$$\min_{\boldsymbol{\sigma} \in \mathbb{R}^K} \|\boldsymbol{u}_t - \Theta(U)\boldsymbol{\sigma}\|_2^2 + \lambda R(\boldsymbol{\sigma}).$$

[1] Samuel H Rudy et al. “Data-driven discovery of partial differential equations”.