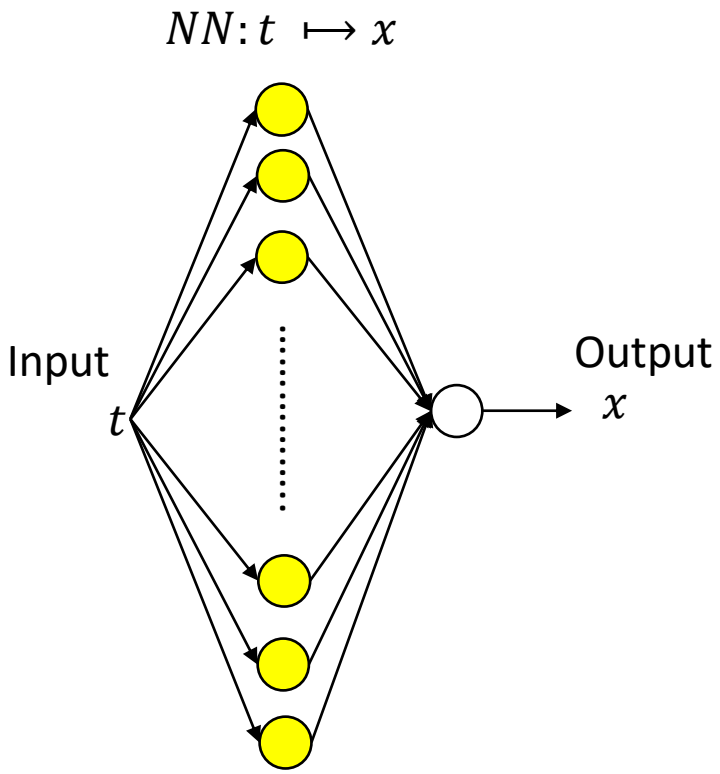
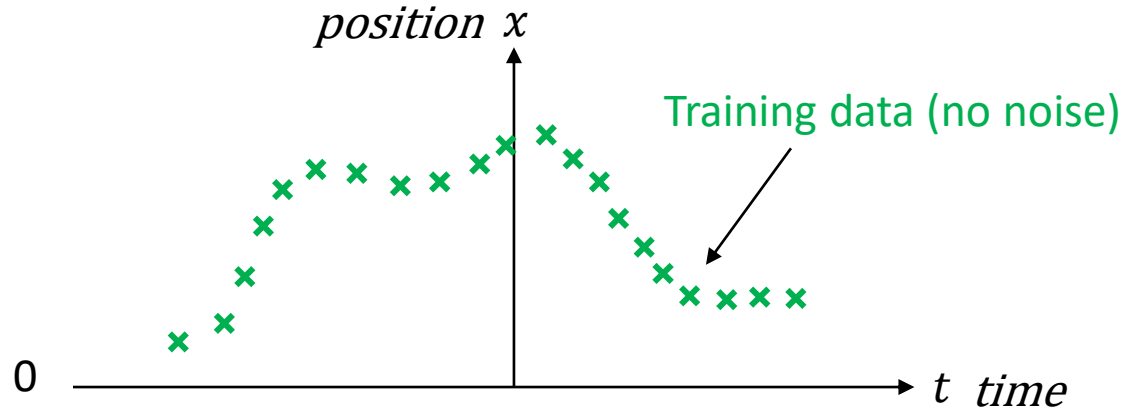
Discovering governing equations from data

Empirical vs Physics-informed fitting

- The data $\{t_d^i\ ,\ x_d^i\}$ is governed by dynamics: $\frac{dx}{dt} = f(x)$
- (1) Empirical vs (2) Physics-informed fitting



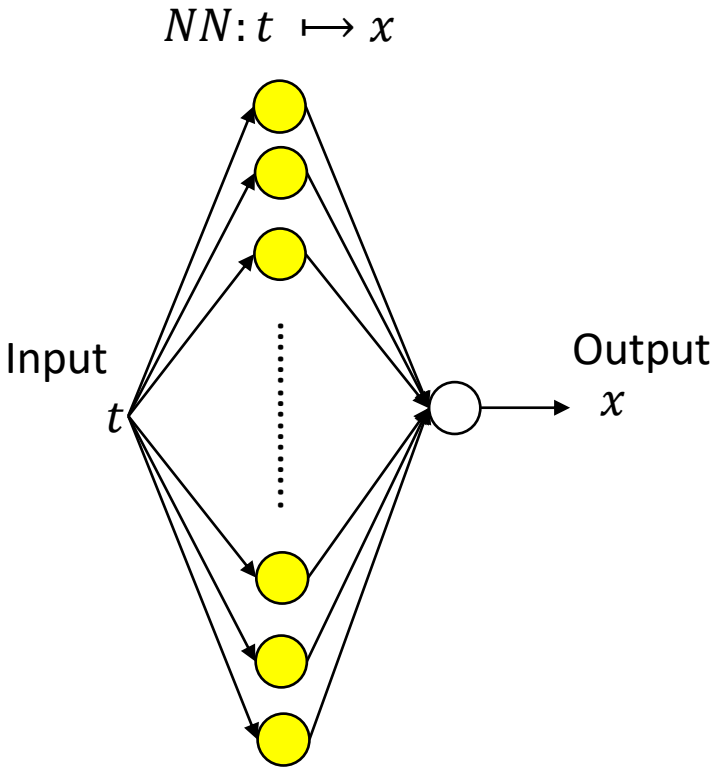


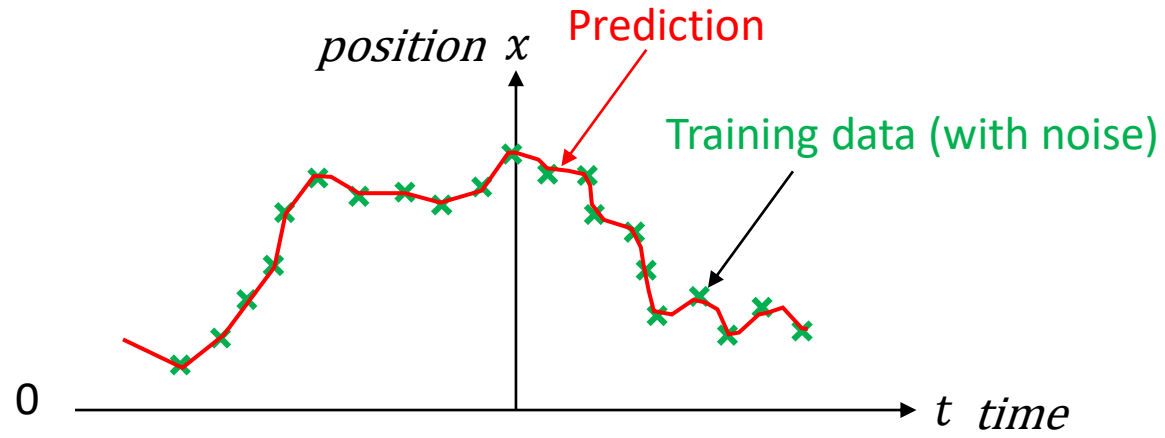
For a given data set $\{t_d^i\ ,\ x_d^i\}$ and an arbitrarily small $\epsilon > 0$
There exist NN prediction $x(t_d^i)$ so that

$$\sum_{i=1}^{n} \left(x(t_d^i) - t_d^i\right)^2 < \epsilon$$

Empirical fitting

- NN can easily overfit noisy data!



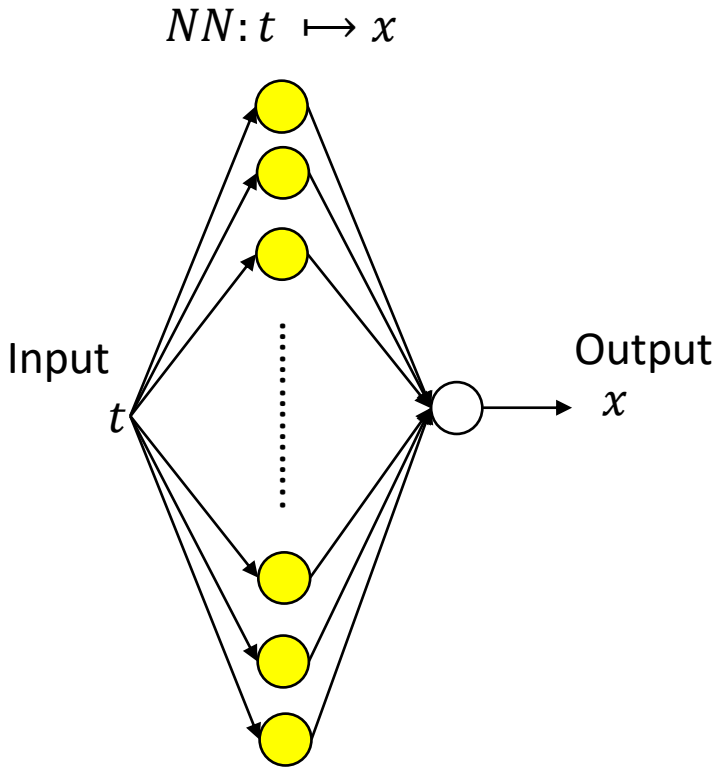


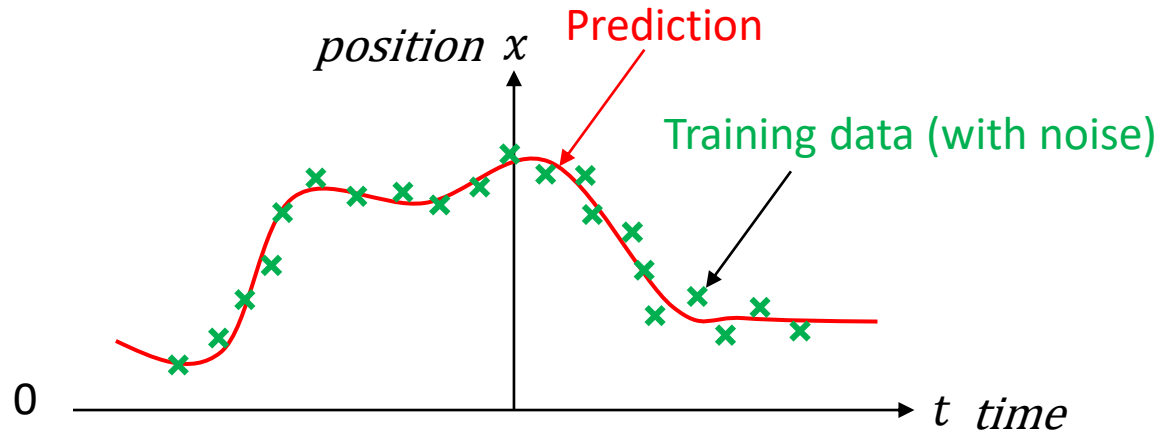
For a given data set $\{t_d^i\ ,\ x_d^i\}$ and an arbitrarily small $\epsilon > 0$
There exist NN prediction $\mathrm{x}\left(t_d^i\right)$ so that

$$\sum_{i=1}^{n}\left(x\left(t_d^i\right) - t_d^i\right)^2 < \epsilon$$

Physics-informed fitting

- NN can easily overfit noisy data!
- We introduced the use of physical laws to regularize the NN prediction



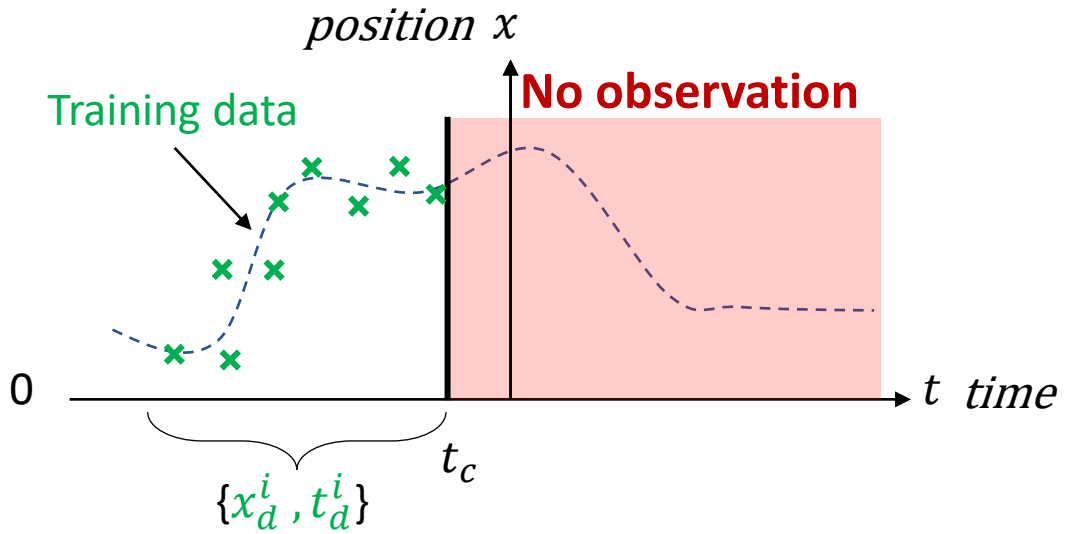


For a given data set $\{t_d^i\ ,\ x_d^i\}$ and an arbitrarily small $\epsilon > 0$
There exist NN prediction $\mathrm{x}\left(t_d^i\right)$ so that

$$\sum_{i=1}^{n}\left(x\left(t_d^i\right) - t_d^i\right)^2 < \epsilon$$

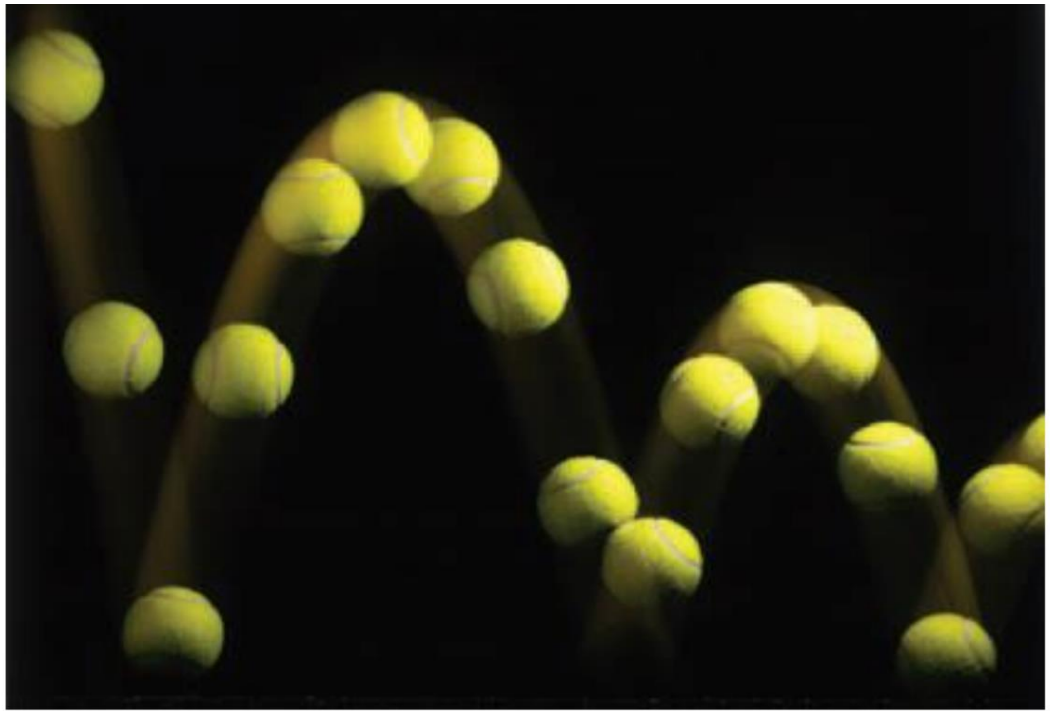
Physics-informed prediction

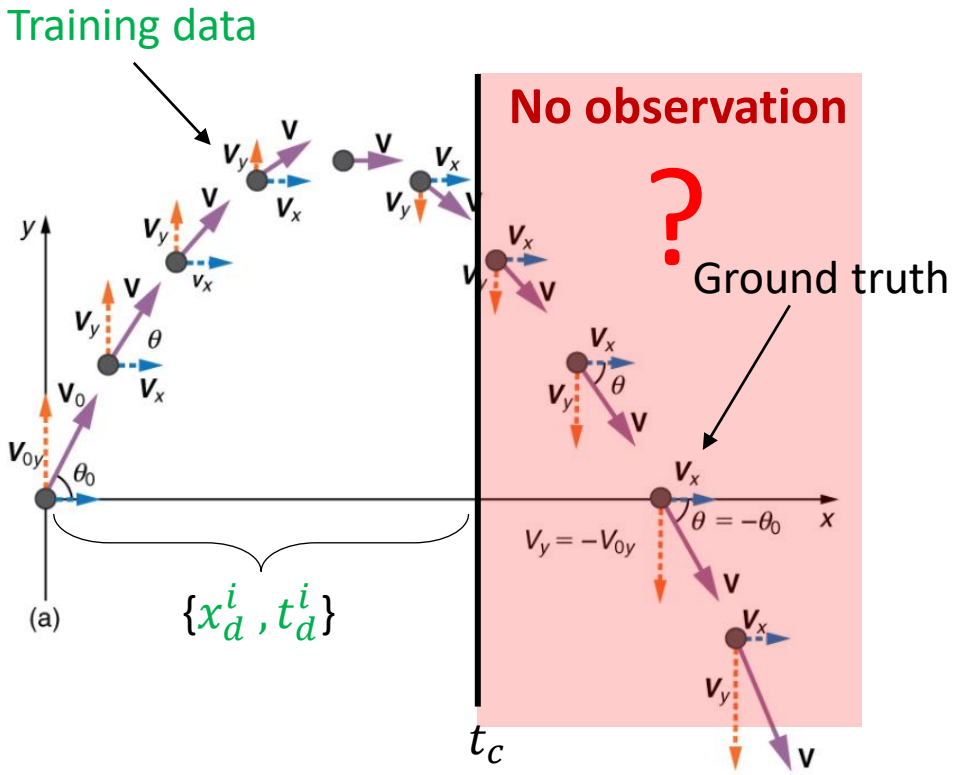
- NN can easily overfit noisy data!
- We introduced the use of physical laws to regularize the NN prediction (**interpolation**)
- PINN prediction can be **extrapolated** to an unseen domain ($t > t_c$)!
- Discovery of **unknown constants/parameter** fields in the physical law
  (e.g. viscosity, pressure, velocity fields…etc)



All these are powerful, but what if we don't have physical laws?

Given training data data $\{t_d^i\ ,\ x_d^i\}$, we want to predict the future ($t > t_c$)

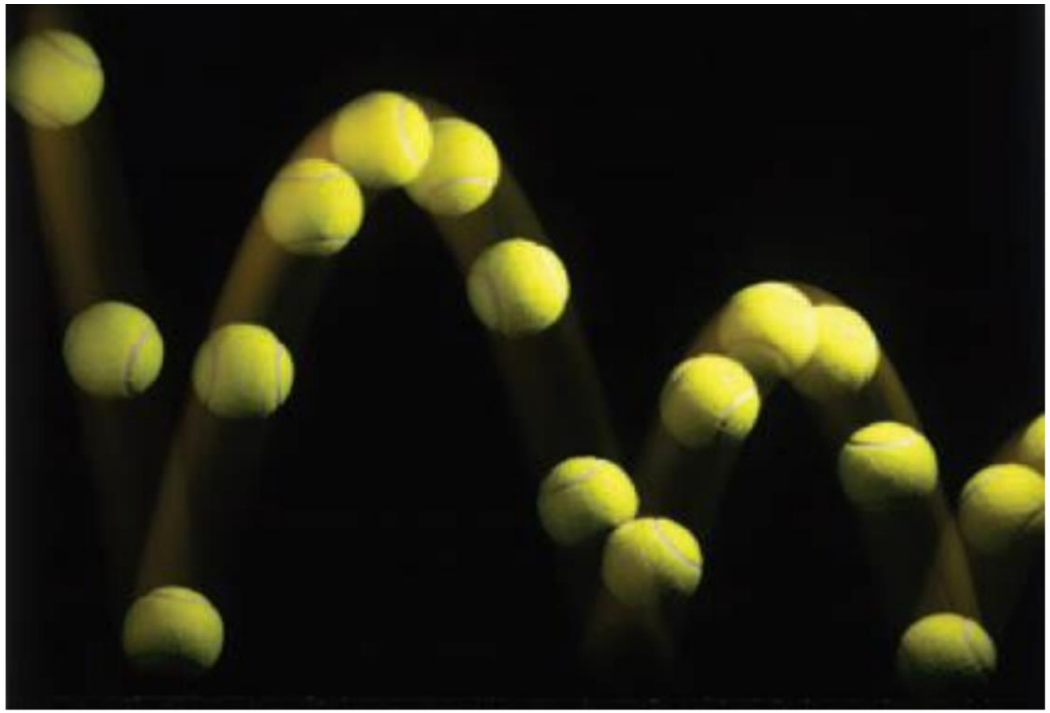


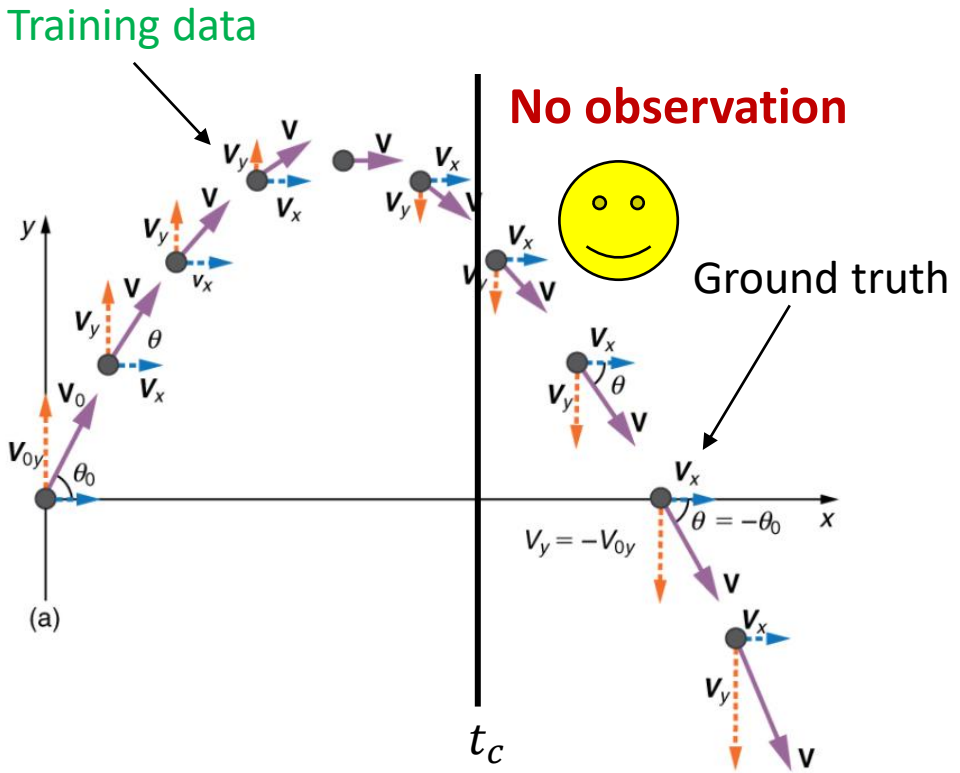


Discover governing equations from data

If our model learn from data that the second derivative of position w.r.t. time is a constant
→ can be **extrapolated** to make prediction for future prediction

$$F = m\frac{d^2x}{dt^2}$$





Discover governing equations from data

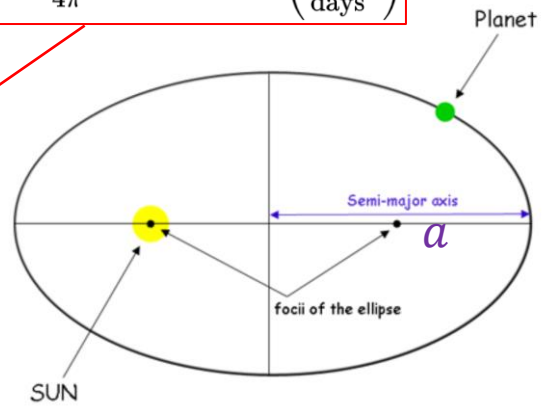
- Newton's second law $\boxed{F=ma}$
- Kepler's third law $\boxed{\frac{a^3}{T^2} = \text{constant}}$

  T is the orbital period
  a is the elliptical semi-major axis

$$\frac{G(M+m)}{4\pi^2} \approx \frac{GM}{4\pi^2} \approx 7.496 \cdot 10^{-6} \left(\frac{\text{AU}^3}{\text{days}^2}\right)$$

Data used by Kepler (1618)

Planet	Mean distance to sun (AU)	Period (days)	$\frac{R^3}{T^2}$ (10^{-6} AU^3/day^2)
Mercury	0.389	87.77	7.64
Venus	0.724	224.70	7.52
Earth	1	365.25	7.50
Mars	1.524	686.95	7.50
Jupiter	5.20	4332.62	7.49
Saturn	9.510	10759.2	7.43

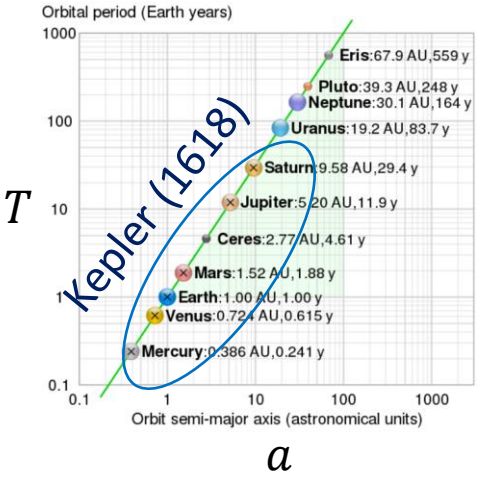


"I first believed I was dreaming... But it is absolutely certain and exact that the ratio which exists between the period times of any two planets is precisely the ratio of the 3/2th power of the mean distance."

— translated from Harmonies of the World by Kepler (1619)

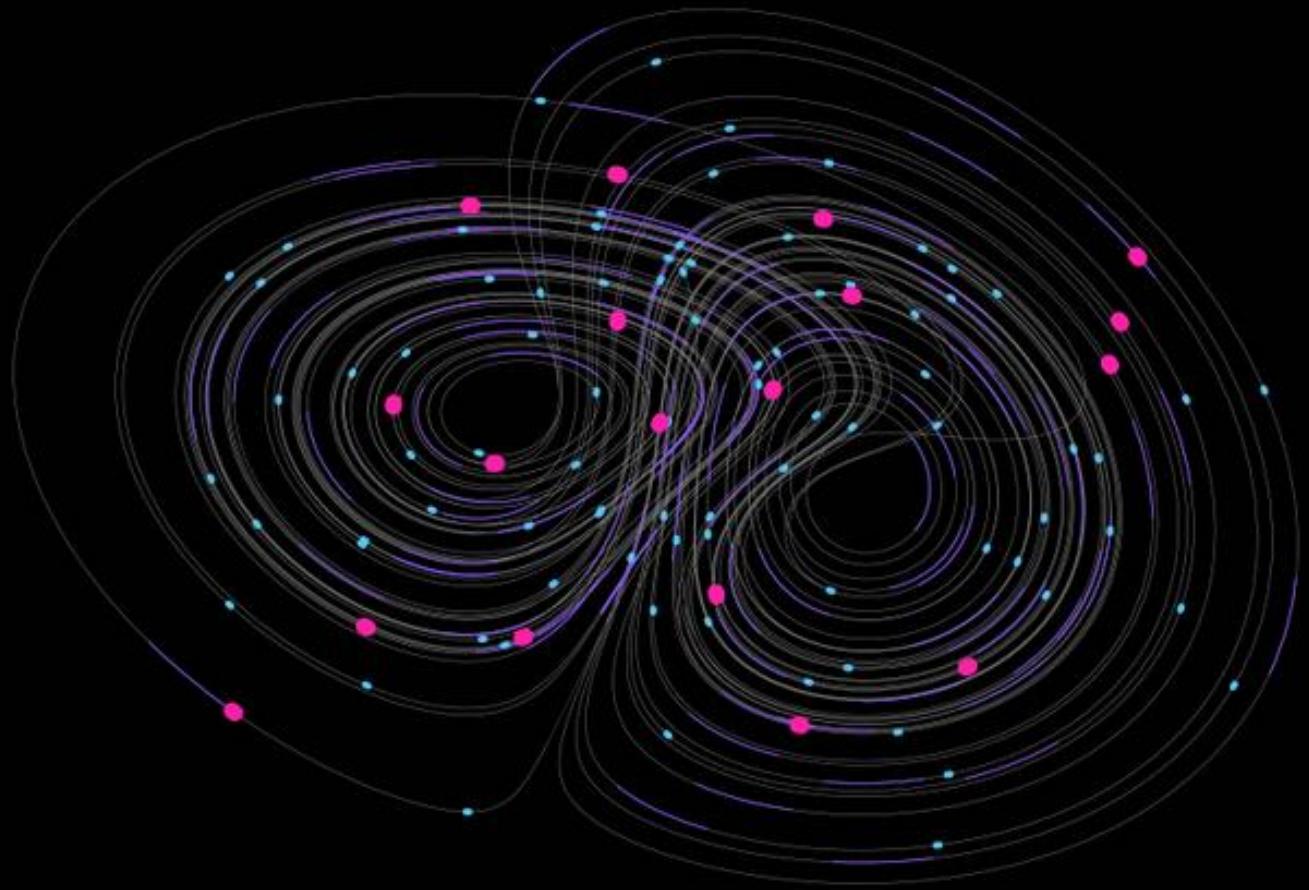
Modern data (Wolfram Alpha Knowledgebase 2018)

Planet	Semi-major axis (AU)	Period (days)	$\frac{R^3}{T^2}$ (10^{-6} AU^3/day^2)
Mercury	0.38710	87.9693	7.496
Venus	0.72333	224.7008	7.496
Earth	1	365.2564	7.496
Mars	1.52366	686.9796	7.495
Jupiter	5.20336	4332.8201	7.504
Saturn	9.53707	10775.599	7.498
Uranus	19.1913	30687.153	7.506
Neptune	30.0690	60190.03	7.504



Discover governing equations from data

Lorenz system (Nonlinear ODE)



Today we will discuss how to find this model from data!

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dx}{dt} = xy - \beta z$$

Brunton et al (2016)

Discover governing equations from data

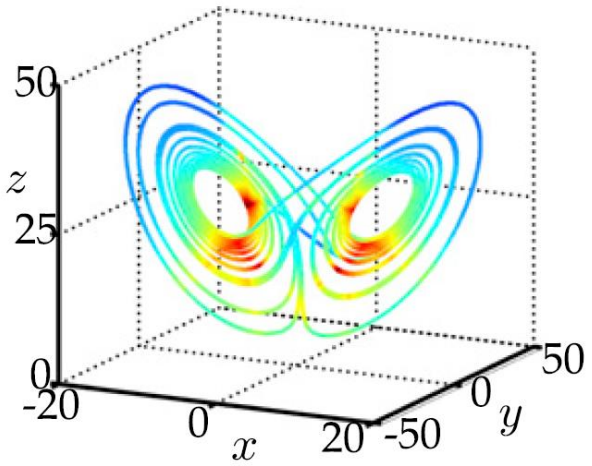
Lorenz system (Nonlinear ODE)

A library of functions

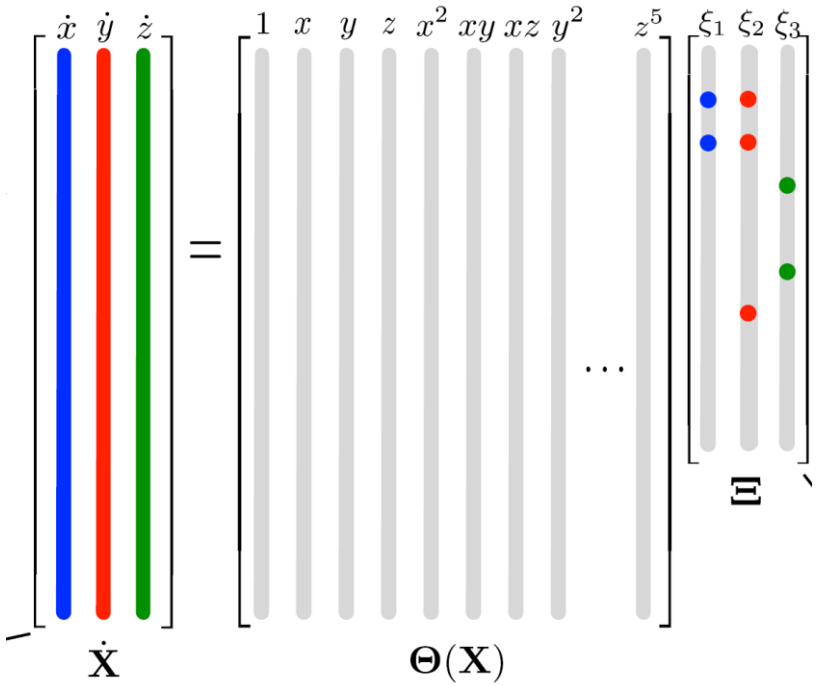
Coefficients

I. True Lorenz System

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= x(\rho-z)-y\\ \dot{z} &= xy-\beta z.\end{aligned}$$



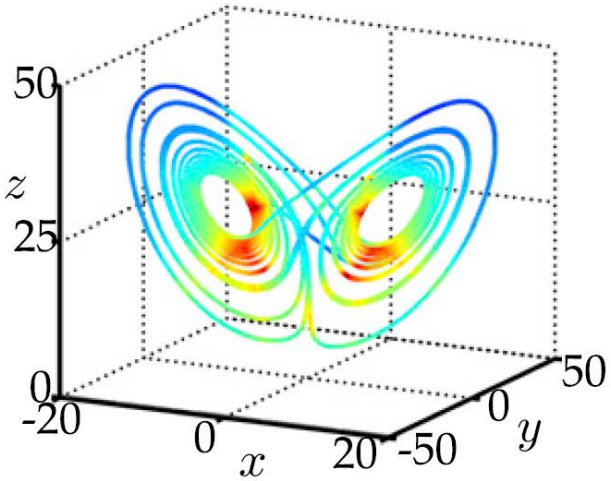
Model in



Model out

III. Identified System

$$\begin{aligned}\dot{x} &= \Theta(\mathbf{x}^T)\xi_1\\ \dot{y} &= \Theta(\mathbf{x}^T)\xi_2\\ \dot{z} &= \Theta(\mathbf{x}^T)\xi_3\end{aligned}$$



Part 1. Discovering ODE from data

Brunton et al (2016)

SINDy- Sparse Identification of Nonlinear Dynamics

- A guided tool for model discovery
- A novel framework to discover governing equations underlying a dynamical system simply from data measurements, leveraging advances in sparsity techniques and machine learning.
- Applications: inferring climate patterns, determining stability of financial markets, predicting and suppressing the spread of disease.

The Goal

given

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

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

want to know this

the dynamic constraints that define the equations of motion of the system

E.g., **Lorenz system** (Nonlinear ODE)

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dx}{dt} = xy - \beta z$$

$$\boldsymbol{x} = [x(t), y(t), z(t)],\ n = 3$$

$$\frac{d\boldsymbol{x}}{dt} = \left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{dz}{dt}\right],\ n = 3$$

$$\boldsymbol{f}(\boldsymbol{x}(t)) = [\sigma(y - x), x(\rho - z) - y, xy - \beta z]$$

The Method

collect together the time-series data...

[implicit:]
choose a measurement coordinate system

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(t_1) \\ \mathbf{x}^T(t_2) \\ \vdots \\ \mathbf{x}^T(t_m) \end{bmatrix} = \overset{\text{state} \rightarrow}{\begin{bmatrix} x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \cdots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \cdots & x_n(t_m) \end{bmatrix}} \Big\downarrow \text{time}$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}^T(t_1) \\ \dot{\mathbf{x}}^T(t_2) \\ \vdots \\ \dot{\mathbf{x}}^T(t_m) \end{bmatrix} = \begin{bmatrix} \dot{x}_1(t_1) & \dot{x}_2(t_1) & \cdots & \dot{x}_n(t_1) \\ \dot{x}_1(t_2) & \dot{x}_2(t_2) & \cdots & \dot{x}_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{x}_1(t_m) & \dot{x}_2(t_m) & \cdots & \dot{x}_n(t_m) \end{bmatrix}.$$

The Method

then, define a basis library...

$$\boldsymbol{\Theta}(\mathbf{X}) = \begin{bmatrix} | & | & | & | & & | & | & | & | & \\ \mathbf{1} & \mathbf{X} & \mathbf{X}^{P_2} & \mathbf{X}^{P_3} & \cdots & \sin(\mathbf{X}) & \cos(\mathbf{X}) & \sin(2\mathbf{X}) & \cos(2\mathbf{X}) & \cdots \\ | & | & | & | & & | & | & | & | & \end{bmatrix}.$$

where

$$\mathbf{X}^{P_2} = \begin{bmatrix} x_1^2(t_1) & x_1(t_1)x_2(t_1) & \cdots & x_2^2(t_1) & x_2(t_1)x_3(t_1) & \cdots & x_n^2(t_1) \\ x_1^2(t_2) & x_1(t_2)x_2(t_2) & \cdots & x_2^2(t_2) & x_2(t_2)x_3(t_2) & \cdots & x_n^2(t_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1^2(t_m) & x_1(t_m)x_2(t_m) & \cdots & x_2^2(t_m) & x_2(t_m)x_3(t_m) & \cdots & x_n^2(t_m) \end{bmatrix}.$$

The Method

then, choose an optimization algorithm to fit...

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi.$$

where $\Xi = [\,\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n\,]$

is a sparse vector of coefficients

Option 1: LASSO $\boldsymbol{\xi} = \underset{\boldsymbol{\xi}'}{\operatorname{argmin}} \|\Theta\boldsymbol{\xi}' - \mathbf{y}\|_2 + \lambda\|\boldsymbol{\xi}'\|_1.$

Option 2: STLS- **Sequential Thresholded Least-Squares**
(fast and accurate)

The Method

What is Sequential Thresholded Least-Squares?

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi.$$

, where $\Xi = [\,\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n\,]$

ξ_k is a sparse vector of coefficients

Hyper parameter: λ

0. Find the least-squares solution Ξ to $\left\|\dot{X} - \Theta\Xi\right\|_2$
1. In each iteration, when the coefficients in $\Xi < \lambda$, set coefficients in Ξ to zero
2. Know which terms in the function library Θ is NOT important. Drop those terms.
3. Recalculate the new least-squares solution to Ξ_{new} with the data $\dot{X}$ and the new function library Θ_{new} (now with less terms)
4. Repeat 1-3 until coefficients in Ξ doesn't change with new iterations.

Computationally efficient, and it rapidly converges to a sparse solution in a small number of iterations!

The Method

What is Sequential Thresholded Least-Squares?

$$\dot{\mathbf{X}} = \Theta(\mathbf{X})\Xi.$$

, where $\Xi = [\,\boldsymbol{\xi}_1\ \boldsymbol{\xi}_2\ \cdots\ \boldsymbol{\xi}_n\,]$

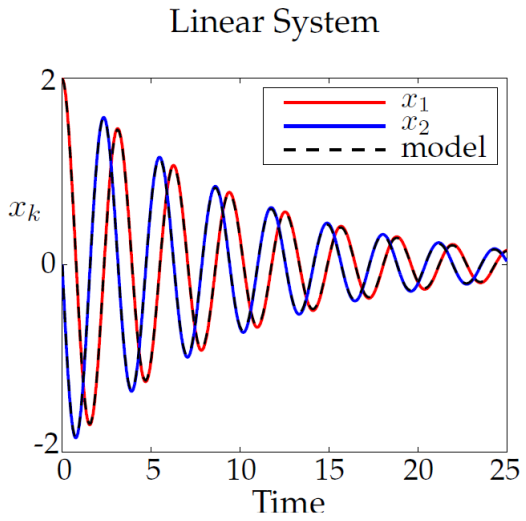
ξ_k is a sparse vector of coefficients

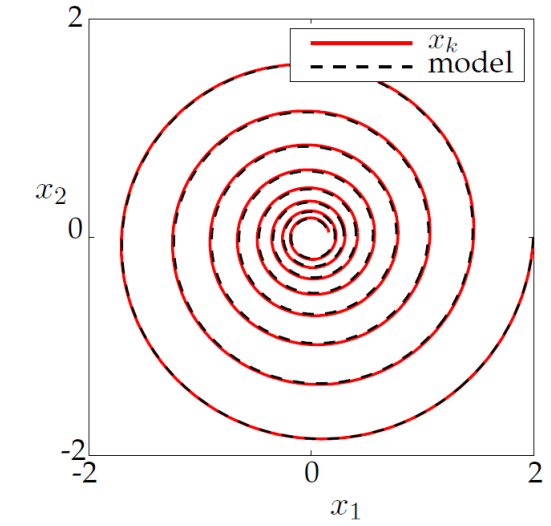
```
%% compute Sparse regression: sequential least squares
Xi = Theta\dXdt;  % initial guess: Least-squares

% lambda is our sparsification knob.
for k=1:10
    smallinds = (abs(Xi)<lambda);   % find small coefficients
    Xi(smallinds)=0;                % and threshold
    for ind = 1:n                   % n is state dimension
        biginds = ~smallinds(:,ind);
        % Regress dynamics onto remaining terms to find sparse Xi
        Xi(biginds,ind) = Theta(:,biginds)\dXdt(:,ind);
    end
end
```

E.g., 2D damped oscillator (ODEs)

- Linear

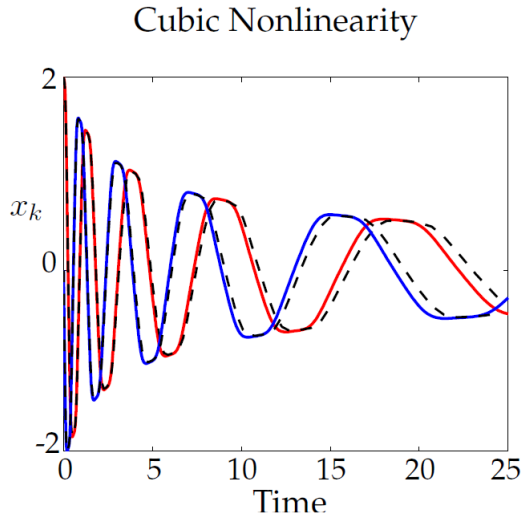
$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1 & 2 \\ -2 & -0.1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

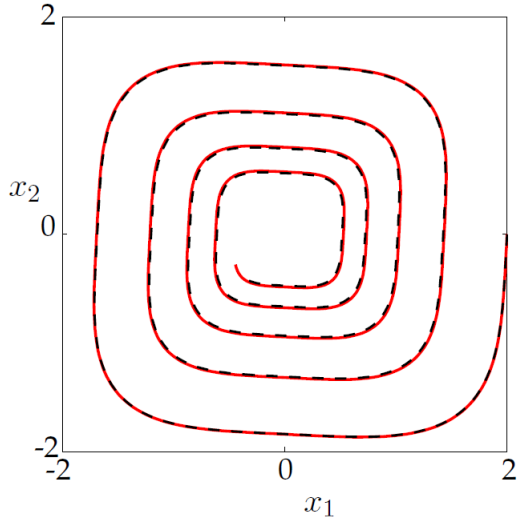




- Non-linear

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.1 & 2 \\ -2 & -0.1 \end{bmatrix}\begin{bmatrix} x^3 \\ y^3 \end{bmatrix}$$





E.g., Lorenz Systems

Training data:

Parameters:

$\sigma = 10, \beta = 8/3, \rho = 28$

Initial condition:

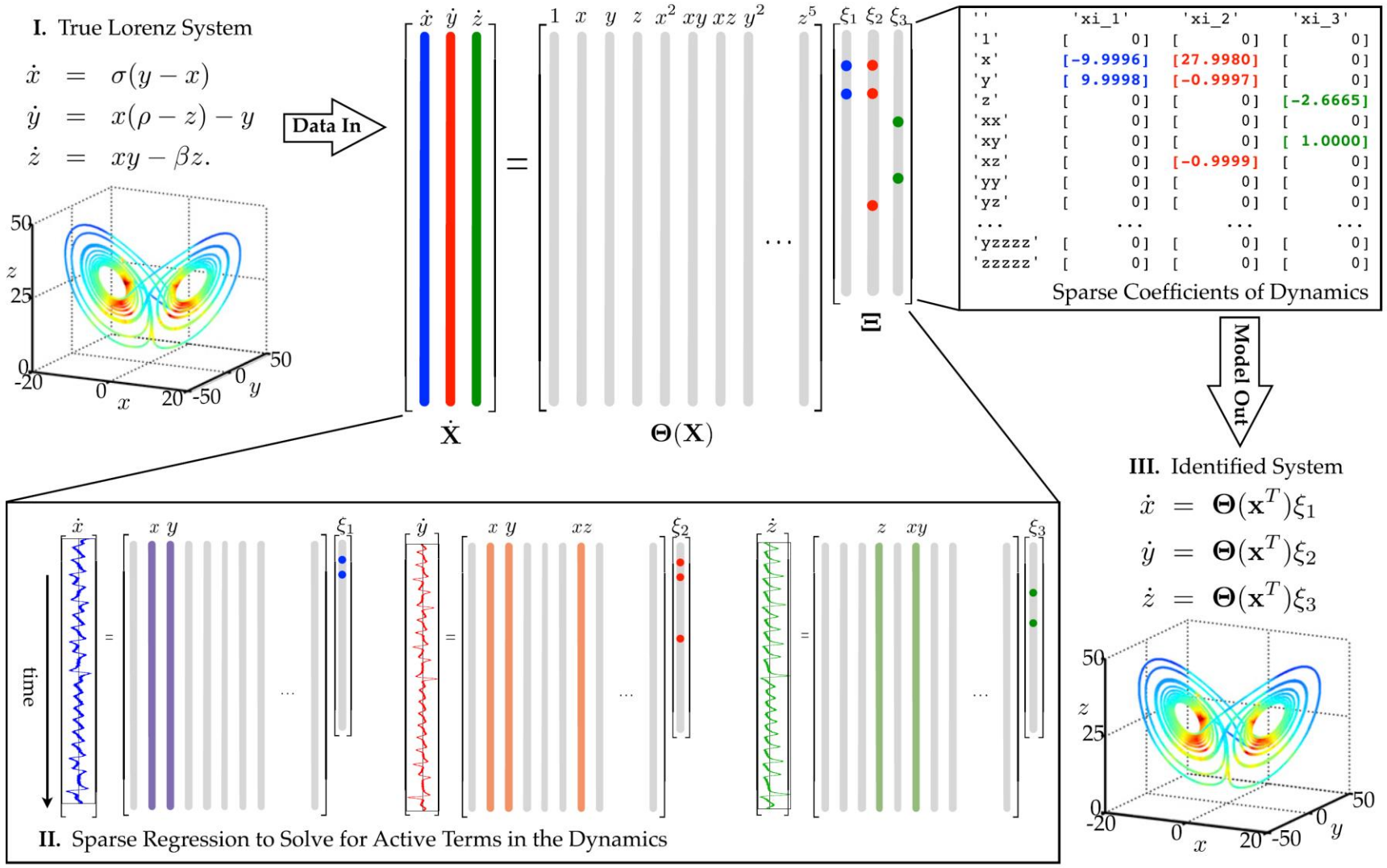
$(x_0, y_0, z_0)^T = (-8, 7, 27)^T$

Time resolution:

Data is collected from $t = 0$ to $t = 100$ with a time-step of $\Delta t = 0.001$

Function library:

polynomials in (x; y; z)
up to fifth order



E.g., Lorenz Systems

Add Gaussian noise $N(0, \eta)$ to data $\dot{X}, X$

Training data:

Parameters:

$\sigma = 10, \beta = 8/3, \rho = 28$

Initial condition:

$(x_0, y_0, z_0)^T = (-8, 7, 27)^T$

Time resolution:

Data is collected from $t = 0$ to $t = 100$ with a time-step of $\Delta t = 0.001$

Function library:

polynomials in (x; y; z)
up to fifth order

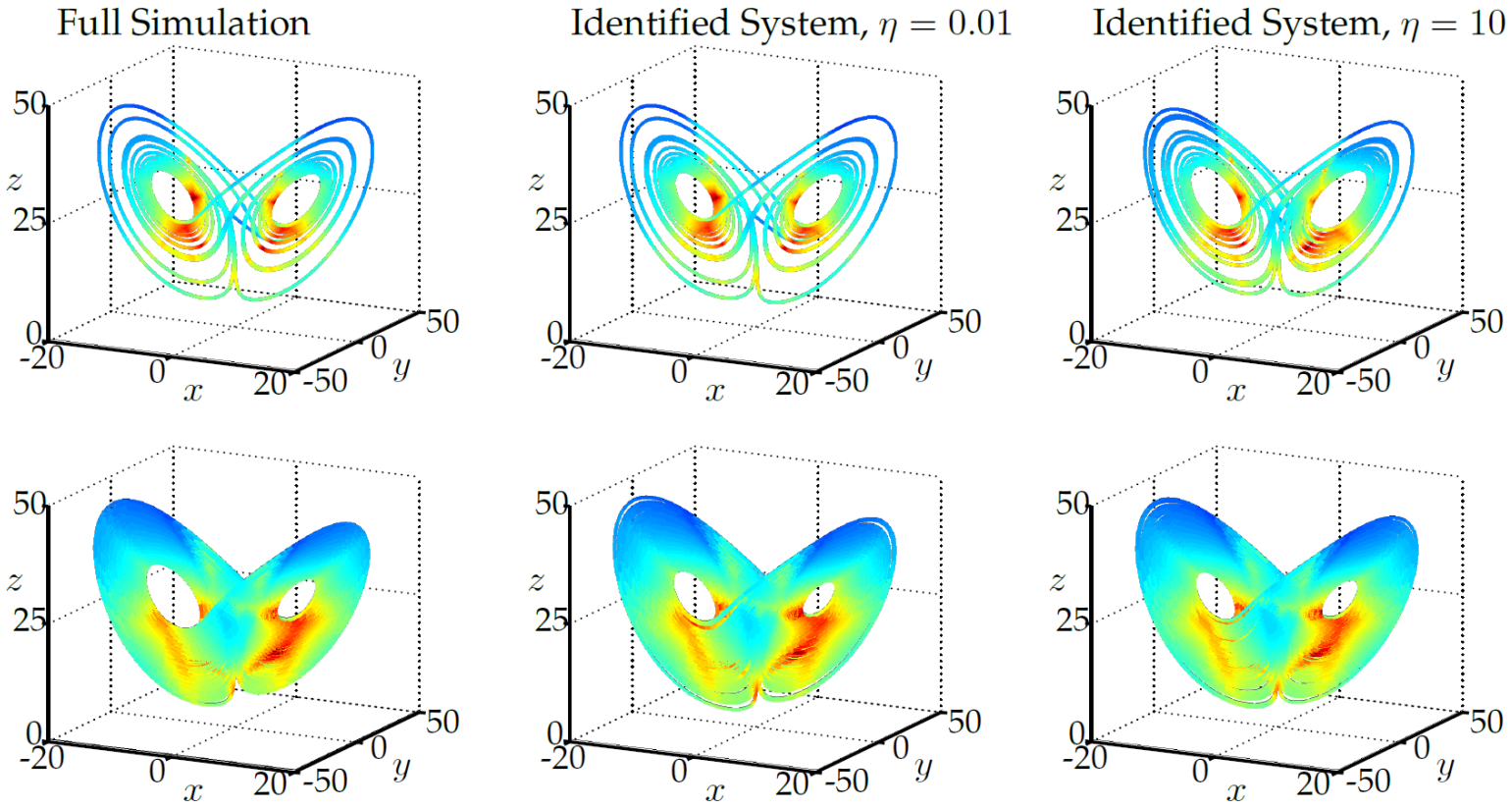


Figure 4: Trajectories of the Lorenz system for short-time integration from $t = 0$ to $t = 20$ (top) and long-time integration from $t = 0$ to $t = 250$ (bottom). The full dynamics (left) are compared with the sparse identified systems (middle, right) for various additive noise, assuming measurements of $\mathbf{x}$ and $\dot{\mathbf{x}}$. The trajectories are colored by Δt, the adaptive Runge-Kutta time step. This color is a proxy for local sensitivity.

E.g., Lorenz Systems

Add Gaussian noise $N(0, \eta)$ to data $\dot{X}, X$

Training data:

Parameters:

$\sigma = 10, \beta = 8/3, \rho = 28$

Initial condition:

$(x_0, y_0, z_0)^T = (-8, 7, 27)^T$

Time resolution:

Data is collected from $t = 0$ to $t = 100$ with a time-step of $\Delta t = 0.001$

Function library:

polynomials in (x; y; z)
up to fifth order

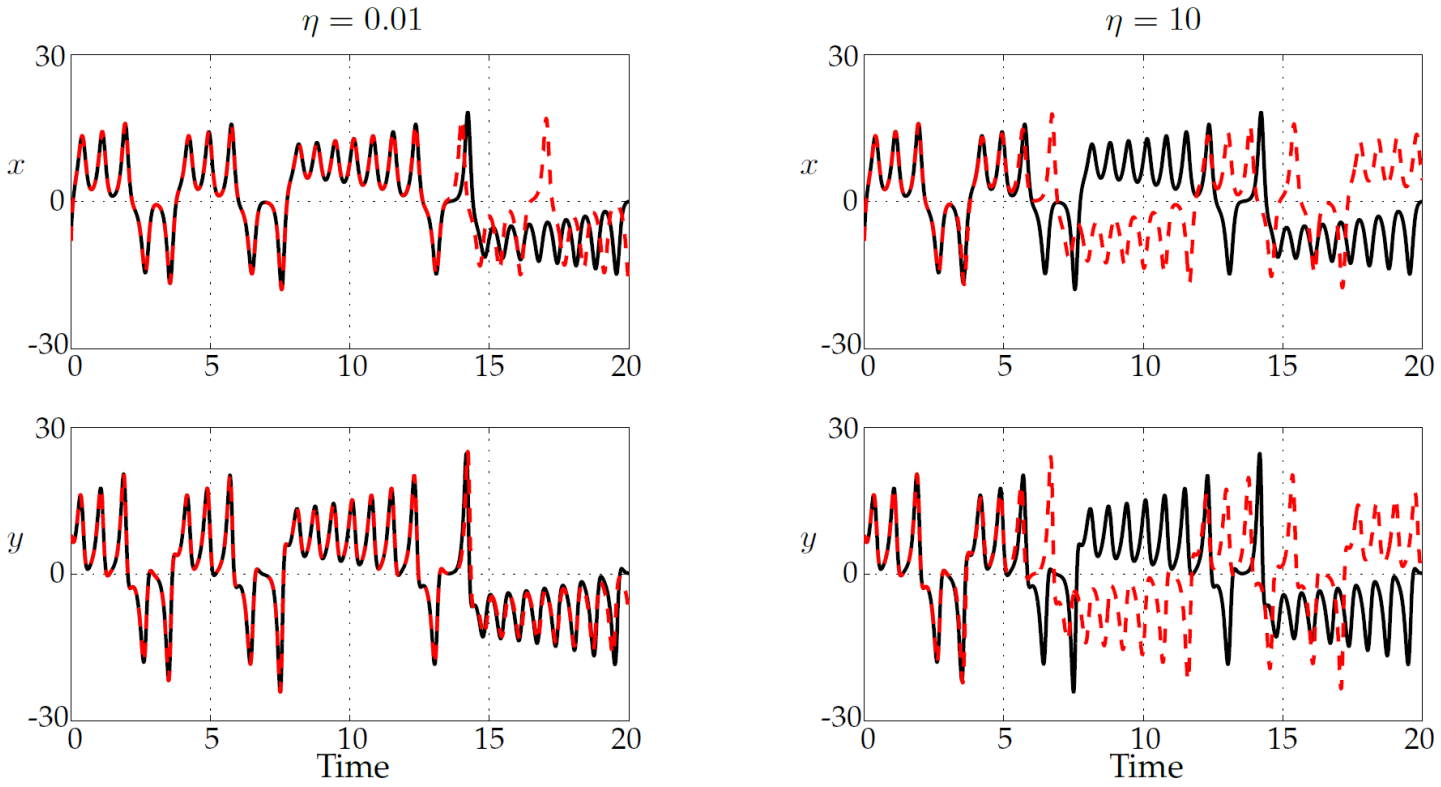


Figure 5: Dynamo view of trajectories of the Lorenz system for the illustrative case where $\mathbf{x}$ and $\dot{\mathbf{x}}$ are measured with noise. The exact system is shown in black (−) and the sparse identified system is shown in the dashed red arrow (−−).

E.g., Lorenz Systems

Add Gaussian noise $N(0, \eta)$ to data X

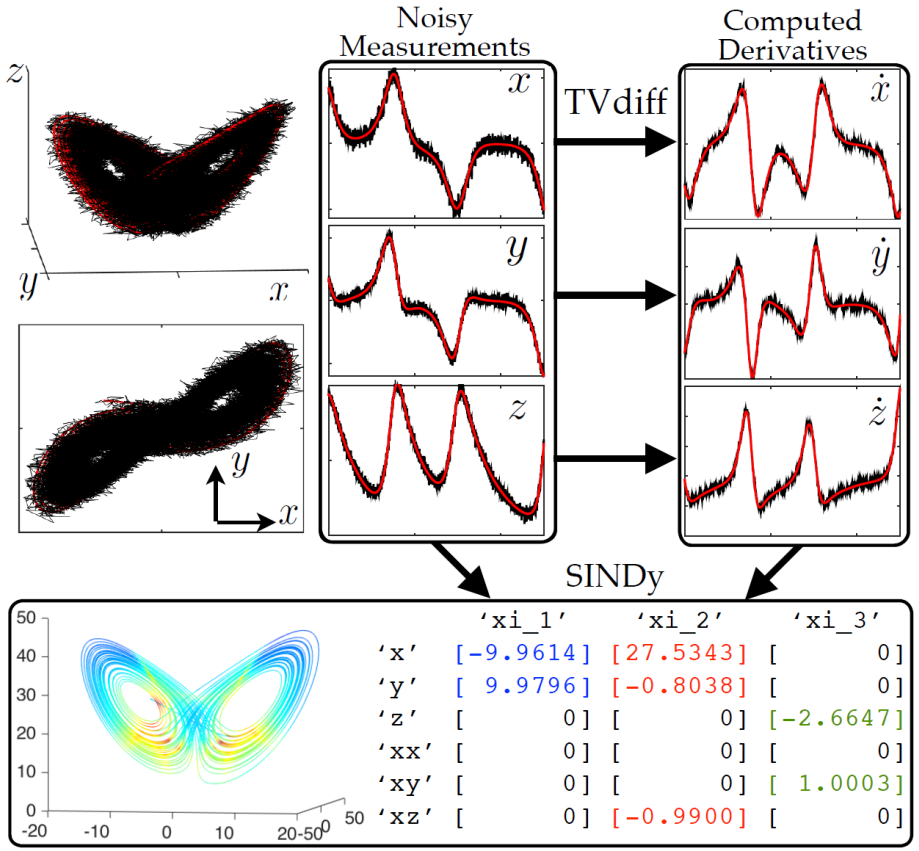


Figure 7: SINDy procedure when only noisy state measurements are available for the Lorenz system. Gaussian noise with $\sigma = 1$ is added to the state, and derivatives are computed using the total variation derivative [32]. The exact system without noise is shown in red, and the noisy measurements and approximated derivatives are shown in black.

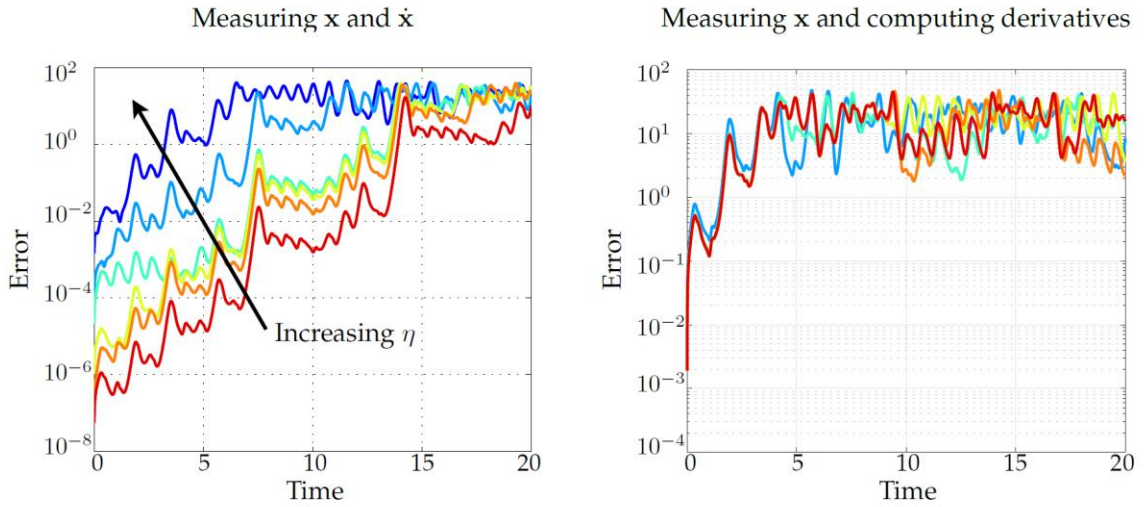


Figure 6: Error vs. time for sparse identified systems generated from data with increasing noise magnitude η. When x and $\dot{x}$ are measured, then noise is added to the derivative (left). This error corresponds to the difference between solid black and dashed red curves in Fig. 5. Sensor noise values are $\eta \in \{0.0001, 0.001, 0.01, 0.1, 1.0, 10.0\}$. When only x is measured, noise is added to the state, and the derivatives $\dot{x}$ are computed using the total variation regularized derivative [32] (right). In this case, the largest noise magnitude $\eta = 10.0$ is omitted, because the approximation fails.

E.g., Flow around a cylinder

2D Navier–Stokes equations

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}),$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}),$$

$$u_x + v_y = 0$$

can be written as

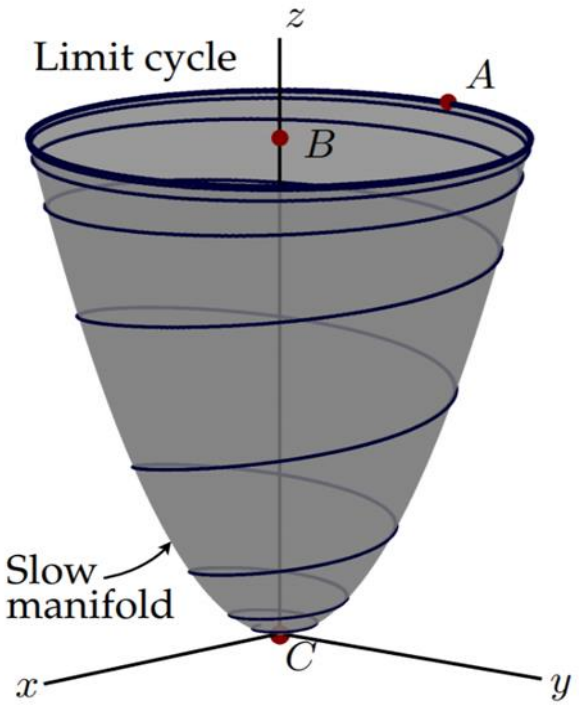
lower-order ODE model

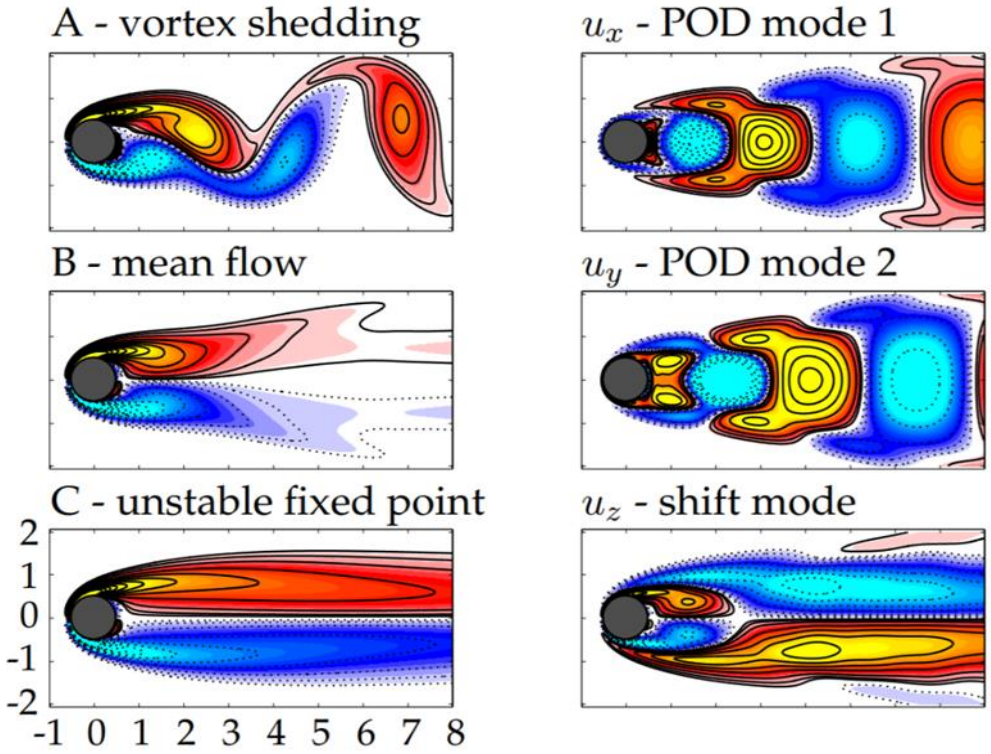
Noack et al (2003)

$$\dot{x} = \mu x - \omega y + Axz,$$

$$\dot{y} = \omega x + \mu y + Ayz,$$

$$\dot{z} = -\lambda\left(z - x^2 - y^2\right)$$





E.g., Flow around a cylinder

2D Navier–Stokes equations

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}),$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}),$$

$$u_x + v_y = 0$$

can be written as

lower-order ODE model

Noack et al (2003)

$$\dot{x} = \mu x - \omega y + Axz,$$

$$\dot{y} = \omega x + \mu y + Ayz,$$

$$\dot{z} = -\lambda(z - x^2 - y^2)$$

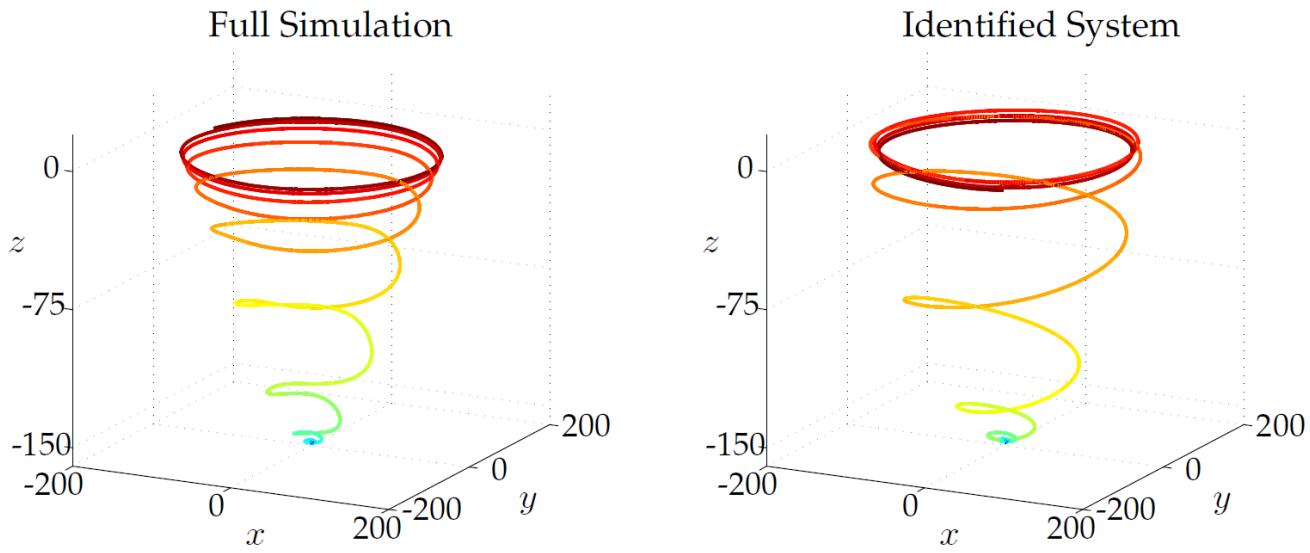


Figure 9: Evolution of the cylinder wake trajectory in reduced coordinates. The full simulation (left) comes from direct numerical simulation of the Navier-Stokes equations, and the identified system (right) captures the dynamics on the slow manifold. Color indicates simulation time.

Limitations of this method?

- Sparse (in time) and noise data?
  (PINN won't suffer as much)
- Partial differential equations?
- What if the important nonlinear terms are not listed in the function library Θ?

E.g. Glycolytic oscillator model

$$\frac{dS_1}{dt} = J_0 - \frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q},$$

$$\frac{dS_2}{dt} = 2\frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5,$$

$$\frac{dS_3}{dt} = k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6),$$

$$\frac{dS_4}{dt} = k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7),$$

$$\frac{dS_5}{dt} = k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5,$$

$$\frac{dS_6}{dt} = -2\frac{k_1 S_1 S_6}{1 + (S_6/K_1)^q} + 2k_3 S_3 (A - S_6) - k_5 S_6,$$

$$\frac{dS_7}{dt} = \psi\kappa (S_4 - S_7) - kS_7.$$

SINDy fails to identify these

''	'S1dot'	'S2dot'	'S3dot'	'S4dot'	'S5dot'	'S6dot'	'S7dot'
'1'	[-780.181]	[1.565e+03]	[0]	[0]	[0]	[-1.565e+03]	[0]
'S1'	[-82.317]	[164.633]	[0]	[0]	[0]	[-164.633]	[0]
'S2'	[-70.328]	[134.657]	[6.00]	[0]	[6.00]	[-140.657]	[0]
'S3'	[-1.578e+04]	[3.156e+04]	[-64.00]	[64.00]	[0]	[-3.143e+04]	[0]
'S4'	[-1.044e+03]	[2.087e+03]	[0]	[-13.00]	[0]	[-2.087e+03]	[1.3000]
'S5'	[1.309e+04]	[-2.618e+04]	[0]	[0]	[0]	[2.618e+04]	[0]
'S6'	[-439.231]	[878.461]	[0]	[0]	[0]	[-879.741]	[0]
'S7'	[3.982e+04]	[-7.964e+04]	[0]	[13.00]	[0]	[7.964e+04]	[-3.1000]
'S1S1'	[18.308]	[-36.615]	[0]	[0]	[0]	[36.615]	[0]
'S1S2'	[253.763]	[-507.527]	[0]	[0]	[0]	[507.526]	[0]
'S1S3'	[3.706e+03]	[-7.412e+03]	[0]	[0]	[0]	[7.412e+03]	[0]
'S1S4'	[-607.006]	[1.214e+03]	[0]	[0]	[0]	[-1.214e+03]	[0]
'S1S5'	[-1.752e+03]	[3.505e+03]	[0]	[0]	[0]	[-3.505e+03]	[0]
'S1S6'	[311.284]	[-622.568]	[0]	[0]	[0]	[622.568]	[0]
'S1S7'	[-9.857e+03]	[1.972e+04]	[0]	[0]	[0]	[-1.972e+04]	[0]
'S2S2'	[231.996]	[-463.993]	[0]	[0]	[0]	[463.993]	[0]
'S2S3'	[1.107e+04]	[-2.213e+04]	[0]	[0]	[0]	[2.213e+04]	[0]
'S2S4'	[-242.407]	[484.813]	[0]	[0]	[0]	[-484.813]	[0]
'S2S5'	[-8.786e+03]	[1.757e+04]	[-6.00]	[0]	[-18.00]	[-1.757e+04]	[0]
'S2S6'	[434.818]	[-869.636]	[0]	[0]	[0]	[869.636]	[0]
'S2S7'	[-2.041e+04]	[4.082e+04]	[0]	[0]	[0]	[-4.082e+04]	[0]
'S3S3'	[-1.086e+03]	[2.171e+03]	[0]	[0]	[0]	[-2.171e+03]	[0]
'S3S4'	[-3.535e+04]	[7.071e+04]	[0]	[0]	[0]	[-7.071e+04]	[0]
'S3S5'	[2.056e+04]	[-4.112e+04]	[0]	[0]	[0]	[4.112e+04]	[0]
'S3S6'	[5.193e+03]	[-1.039e+04]	[16.00]	[-16.00]	[0]	[1.035e+04]	[0]
'S3S7'	[4.116e+03]	[-8.232e+03]	[0]	[0]	[0]	[8.232e+03]	[0]
'S4S4'	[7.587e+03]	[-1.517e+04]	[0]	[0]	[0]	[1.517e+04]	[0]
'S4S5'	[1.342e+04]	[-2.684e+04]	[0]	[-100.00]	[-100.00]	[2.684e+04]	[0]
'S4S6'	[1.267e+03]	[-2.533e+03]	[0]	[0]	[0]	[2.533e+03]	[0]
'S4S7'	[-1.320e+04]	[2.640e+04]	[0]	[0]	[0]	[-2.640e+04]	[0]
'S5S5'	[-7.955e+03]	[1.591e+04]	[0]	[0]	[0]	[-1.591e+04]	[0]
'S5S6'	[-5.507e+03]	[1.101e+04]	[0]	[0]	[0]	[-1.101e+04]	[0]
'S5S7'	[2.174e+04]	[-4.348e+04]	[0]	[0]	[0]	[4.348e+04]	[0]
'S6S6'	[214.779]	[-429.558]	[0]	[0]	[0]	[429.558]	[0]
'S6S7'	[-1.561e+04]	[3.122e+04]	[0]	[0]	[0]	[-3.122e+04]	[0]
'S7S7'	[8.259e+04]	[-1.652e+05]	[0]	[0]	[0]	[1.652e+05]	[0]

Pause and Ponder

- Is $\left\|\dot{X} - \Theta\Xi\right\|_2$ in the SINDy loss function a equation or data loss?