Discovering governing equations from data

Part 2. Discovering PDE from data

Rudy et al (2017)

The Method- **PDE-FIND**

- Governing eqn: $$u_t = N(u, u_x, u_{xx}, \ldots, x, t, \mu)$$

- Function library: $$\boldsymbol{\Theta}(\mathbf{U}, \mathbf{Q}) = \begin{bmatrix} 1 & \mathbf{U} & \mathbf{U}^2 & \ldots & \mathbf{Q} & \ldots & \mathbf{U}_x & \mathbf{U}\mathbf{U}_x & \ldots & \mathbf{Q}^2\mathbf{U}^3\mathbf{U}_{xxx} \end{bmatrix}$$

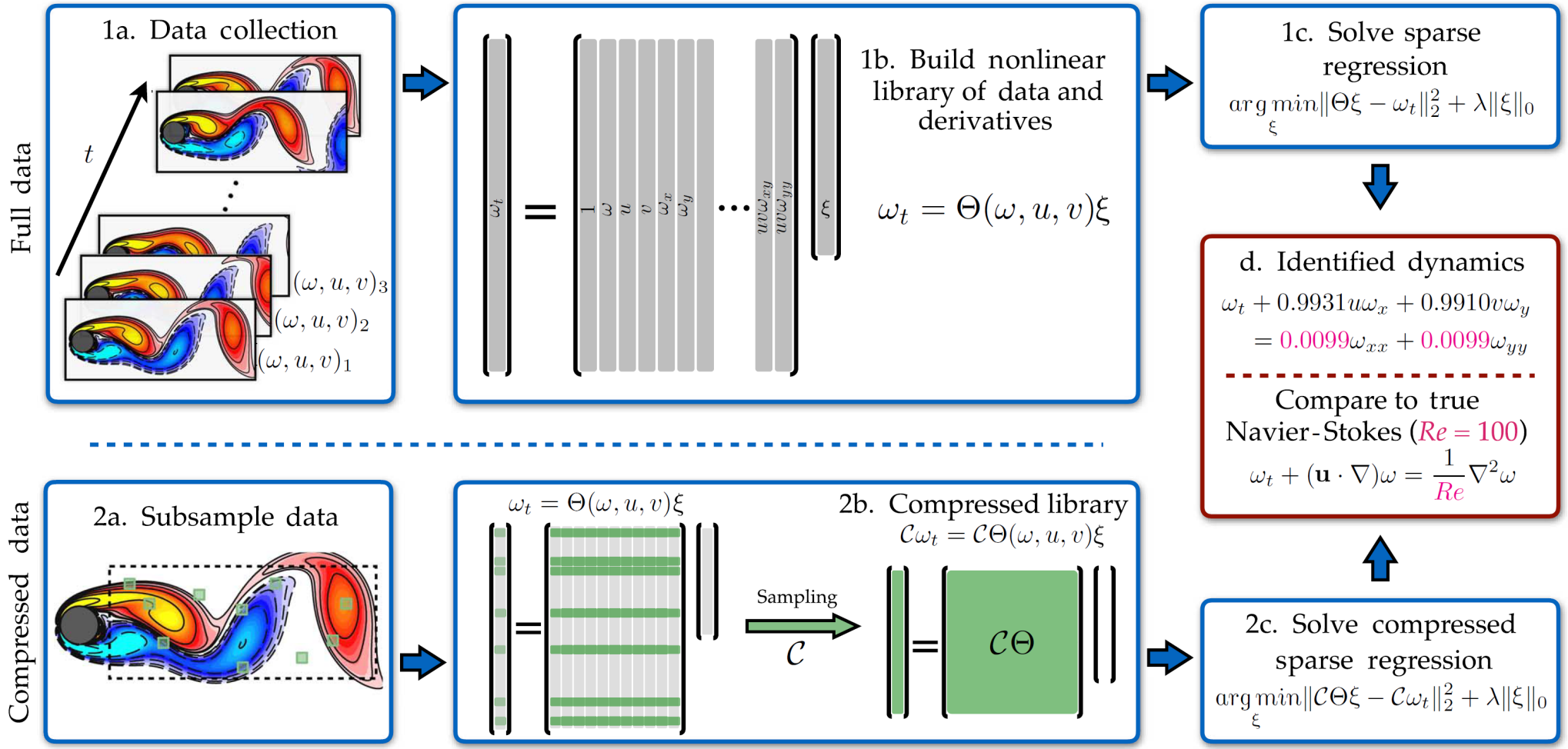
- linear equation representing our PDE for a discretized dataset: $$\mathbf{U}_t = \boldsymbol{\Theta}(\mathbf{U}, \mathbf{Q})\xi$$

Derivatives are taken either using finite differences for clean data, or with polynomial interpolation when noise is added.

$$\begin{bmatrix} u_t(x_0,t_0) \\ u_t(x_1,t_0) \\ u_t(x_2,t_0) \\ \vdots \\ u_t(x_{n-1},t_m) \\ u_t(x_n,t_m) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u(x_0,t_0) & u_x(x_0,t_0) & \ldots & u^5u_{xxx}(x_0,t_0) \\ 1 & u(x_1,t_0) & u_x(x_1,t_0) & \ldots & u^5u_{xxx}(x_1,t_0) \\ 1 & u(x_2,t_0) & u_x(x_2,t_0) & \ldots & u^5u_{xxx}(x_2,t_0) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u(x_{n-1},t_m) & u_x(x_{n-1},t_m) & \ldots & u^5u_{xxx}(x_{n-1},t_m) \\ 1 & u(x_n,t_m) & u_x(x_n,t_m) & \ldots & u^5u_{xxx}(x_n,t_m) \end{bmatrix}}_{\text{Example } \boldsymbol{\Theta} \text{ for real valued function in one spatial dimension}} \begin{bmatrix} \\ \xi \\ \\ \end{bmatrix}$$

- Use an optimization method to find a sparse solution to ξ ⟶ As many zeros as possible in the ξ coefficient vector

Discover PDE from data



E.g., KdV equation $u_t + 6uu_x + u_{xxx} = 0$

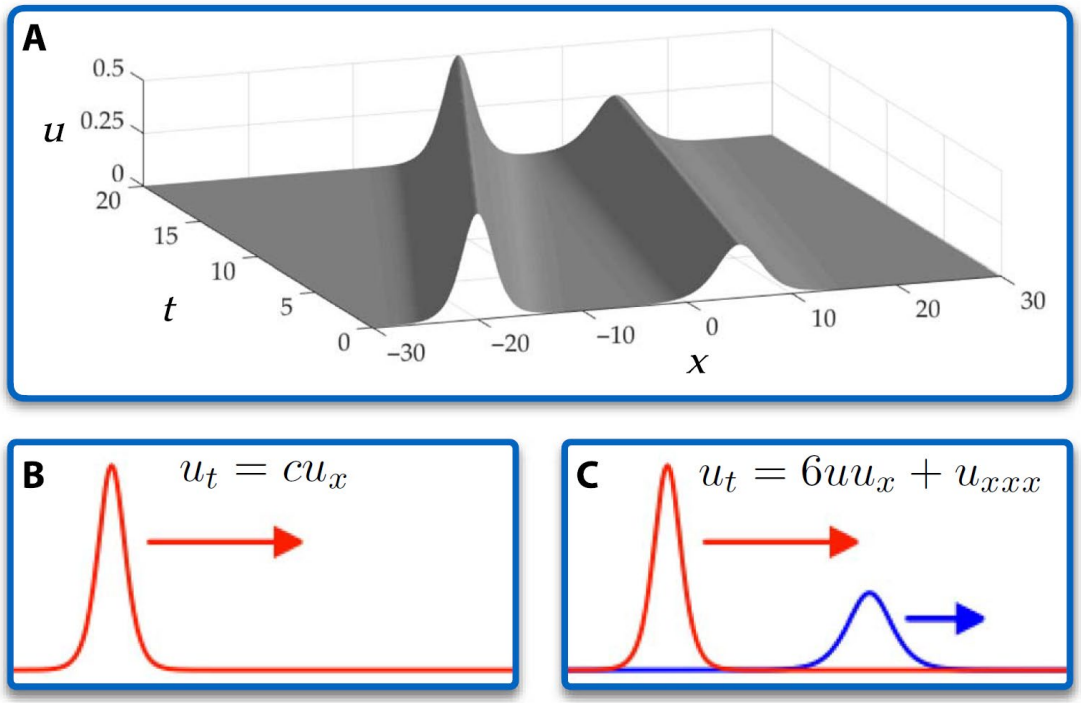


Fig. 3. Inferring nonlinearity via observing solutions at multiple amplitudes. (**A**) Example two-soliton solution to the KdV equation. (**B**) Applying our method to a single soliton solution determines that it solves the standard advection equation. (**C**) Looking at two completely separate solutions reveals nonlinearity.

If one observes a single propagating soliton, it would be indistinguishable from a solution to the one-way wave equation $u_t = cu_x$

PDE-FIND would select the sparsest representation, i.e. $u_t = cu_x$

To get the KdV equation, construct time series data for more than a single initial amplitude.

E.g., Nonlinear Schrodinger equation

table S5. Summary of PDE-FIND for identifying the nonlinear Schrödinger equation.

Correct PDE	$u_t = 0.5iu_{xx} + i\|u\|^2u$
Identified PDE (clean data)	$u_t = 0.500iu_{xx} + 1.000i\|u\|^2u$
Identified PDE (1% noise)	$u_t = 0.479iu_{xx} + 0.982i\|u\|^2u$

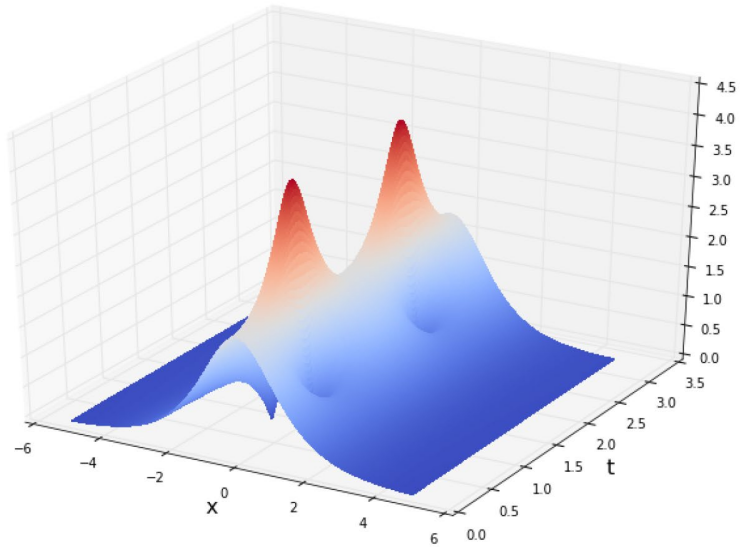


fig. S5. The magnitude of the numerical solution to the nonlinear Schrödinger's equation plotted in space-time.

E.g., Burgers' equation

table S3. Summary of PDE-FIND for identifying Burgers' equation.

Correct PDE	$u_t + uu_x = u_{xx}$
Identified PDE (clean data)	$u_t + 1.001uu_x = 0.100u_{xx}$
Identified PDE (1% noise)	$u_t + 1.010uu_x = 0.103u_{xx}$

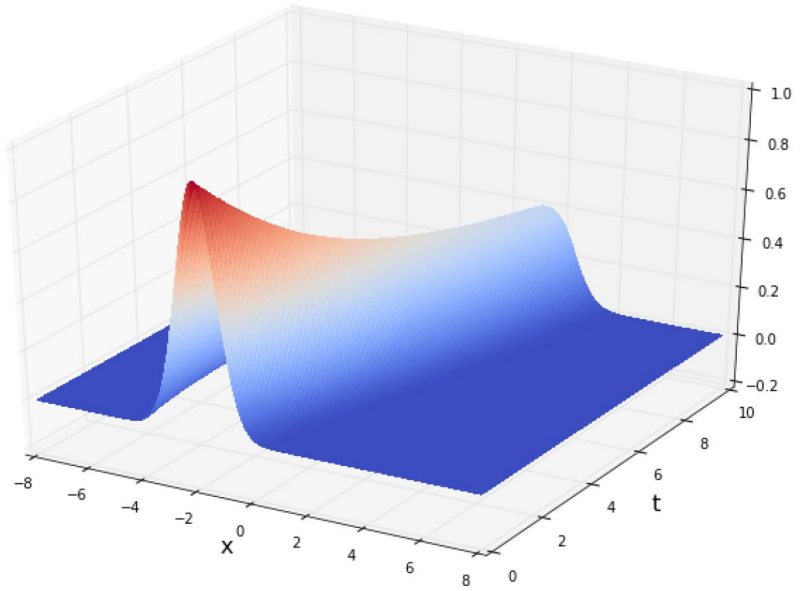


fig. S3. The numerical solution to the Burgers' equation plotted in space-time.

E.g., Reaction-diffusion equation

$$u_t = 0.1\nabla^2 u + \lambda(A)u - \omega(A)v$$
$$v_t = 0.1\nabla^2 v + \omega(A)u + \lambda(A)v$$
$$A = u^2 + v^2, \omega = -\beta A^2, \lambda = 1 - A^2$$

table S7. Summary of PDE-FIND for identifying reaction-diffusion equation.

Correct PDE	$u_t = 0.1u_{xx} + 0.1u_{yy} - uv^2 - u^3 + v^3 + u^2v + u$ $v_t = 0.1v_{xx} + 0.1v_{yy} + v - uv^2 - u^3 - v^3 - u^2v$
Identified PDE (clean data)	$u_t = 0.100u_{xx} + 0.100u_{yy} - 1.000uv^2 - 1.000u^3 + 1.000v^3 + 1.000u^2v + 1.000u$ $v_t = 0.100v_{xx} + 0.100v_{yy} + 1.000v - 1.000uv^2 - 1.000u^3 - 1.000v^3 - 1.000u^2v$
Identified PDE (0.5% noise)	$u_t = 0.095u_{xx} + 0.095u_{yy} - 0.945uv^2 - 0.945u^3 + 1.000v^3 + 1.000u^2v + 0.945u$ $v_t = 0.095v_{xx} + 0.095v_{yy} + 0.946v - 1.000uv^2 - 1.000u^3 - 0.946v^3 - 0.946u^2v$

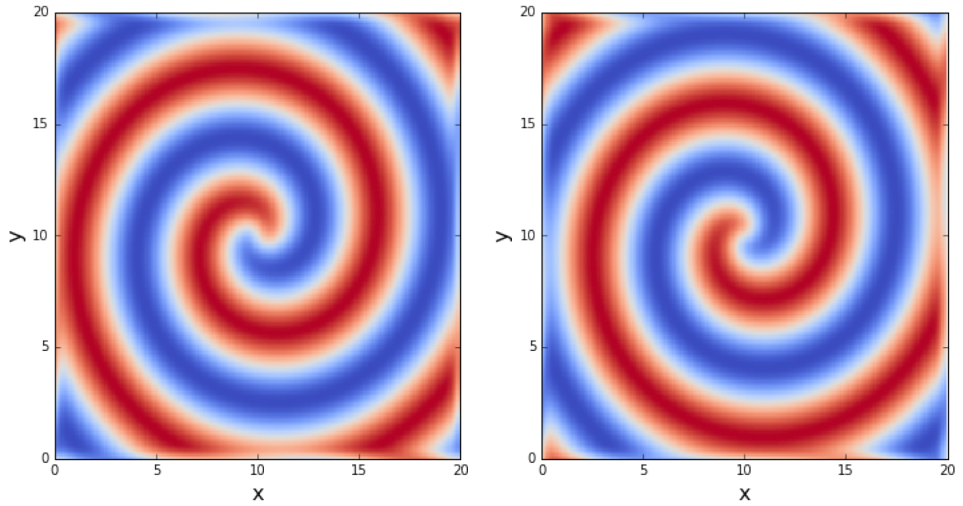


fig. S7. The numerical solution to the reaction-diffusion equation plotted in space-time.

E.g., Navier-Stokes equation

table S8. Summary of PDE-FIND for identifying the Navier-Stokes equation.

Correct PDE	$\omega_t = 0.01\omega_{xx} + 0.01\omega_{yy} - u\omega_x - v\omega_y$
Identified PDE (clean data)	$\omega_t = 0.00988\omega_{xx} + 0.00990\omega_{yy} - 0.990u\omega_x - 0.987v\omega_y$
Identified PDE (1% noise)	$\omega_t = 0.0107\omega_{xx} + 0.0083\omega_{yy} - 0.988u\omega_x - 0.983v\omega_y$

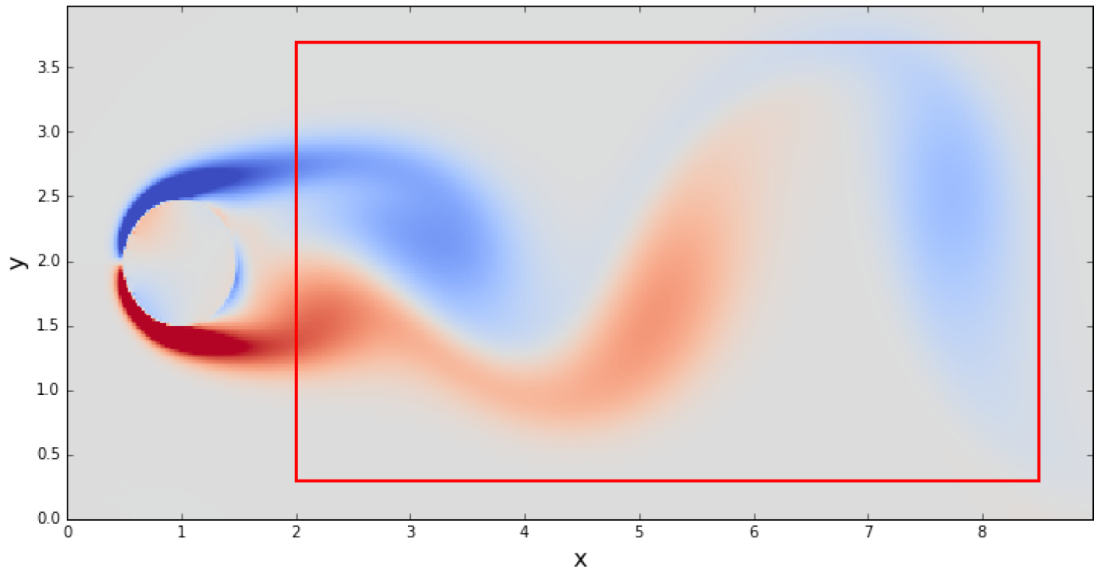
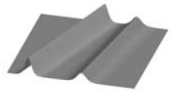
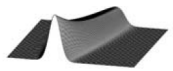
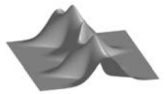
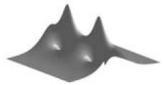
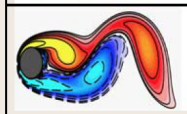


fig. S8. A single snapshot of the vorticity field is illustrated for the fluid flow past a cylinder. The sampling region is outlined in red.

Table 1. Summary of regression results for a wide range of canonical models of mathematical physics. In each example, the correct model structure is identified using PDE-FIND. The spatial and temporal sampling of the numerical simulation data used for the regression is given along with the error produced in the parameters of the model for both no noise and 1% noise. In the reaction-diffusion system, 0.5% noise is used. For Navier-Stokes and reaction-diffusion, the percent of data used in subsampling is also given. NLS, nonlinear Schrödinger; KS, Kuramoto-Sivashinsky.

PDE	Form	Error (no noise, noise)	Discretization
 KdV	$u_t + 6uu_x + u_{xxx} = 0$	$1 \pm 0.2\%, 7 \pm 5\%$	$x \in [-30, 30], n = 512,\ t \in [0, 20], m = 201$
 Burgers	$u_t + uu_x - \epsilon u_{xx} = 0$	$0.15 \pm 0.06\%, 0.8 \pm 0.6\%$	$x \in [-8, 8], n = 256,\ t \in [0, 10], m = 101$
 Schrödinger	$iu_t + \frac{1}{2}u_{xx} - \frac{x^2}{2}u = 0$	$0.25 \pm 0.01\%, 10 \pm 7\%$	$x \in [-7.5, 7.5], n = 512,\ t \in [0, 10], m = 401$
 NLS	$iu_t + \frac{1}{2}u_{xx} + \lvert u\rvert^2 u = 0$	$0.05 \pm 0.01\%, 3 \pm 1\%$	$x \in [-5, 5], n = 512,\ t \in [0, \pi], m = 501$
 KS	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$	$1.3 \pm 1.3\%, 52 \pm 1.4\%$	$x \in [0, 100], n = 1024,\ t \in [0, 100], m = 251$
u  v	$u_t = 0.1\nabla^2 u + \lambda(A)u - \omega(A)v$ $v_t = 0.1\nabla^2 v + \omega(A)u + \lambda(A)v$ $A^2 = u^2 + v^2, \omega = -\beta A^2, \lambda = 1 - A^2$	$0.02 \pm 0.01\%,\ 3.8 \pm 2.4\%$	$x, y \in [-10, 10], n = 256,\ t \in [0, 10], m = 201$ subsample 1.14%
 Navier-Stokes	$\omega_t + (\mathbf{u} \cdot \nabla)\omega = \frac{1}{Re}\nabla^2 \omega$	$1 \pm 0.2\%\ ,\ 7 \pm 6\%$	$x \in [0, 9], n_x = 449,\ y \in [0, 4], n_y = 199,$ $t \in [0, 30], m = 151,$ subsample 2.22%

Codes: https://github.com/snagcliffs/PDE-FIND

Noisy data

Noise Level	Candidate Functions															
	1	u	u^2	u^3	u_x	uu_x	u^2u_x	u^3u_x	u_{xx}	uu_{xx}	u^2u_{xx}	u^3u_{xx}	u_{xxx}	uu_{xxx}	u^2u_{xxx}	u^3u_{xxx}
Clean						−0.986			0.119							
1%						−0.986			0.119							
2%						−0.986			0.118							
3%						−0.986			0.118							
4%						−0.985			0.118							
5%						−0.984			0.117							
6%						−0.983			0.117							
7%						−0.982			0.116							
8%						−0.980			0.115							
9%						−0.978			0.114							
10%				0.168		−1.221			0.076	0.237					−0.090	
11%						−1.127			0.086	0.115					−0.073	
12%	0.001	0.054	−0.239	0.319	−0.064	−0.950	0.190	−0.449	0.035	0.591	−1.179	0.942	−0.006	0.081	−0.300	0.178

fig. S14. Results of PDE-FIND applied to Burgers' equation for varying levels of noise.

Noisy data

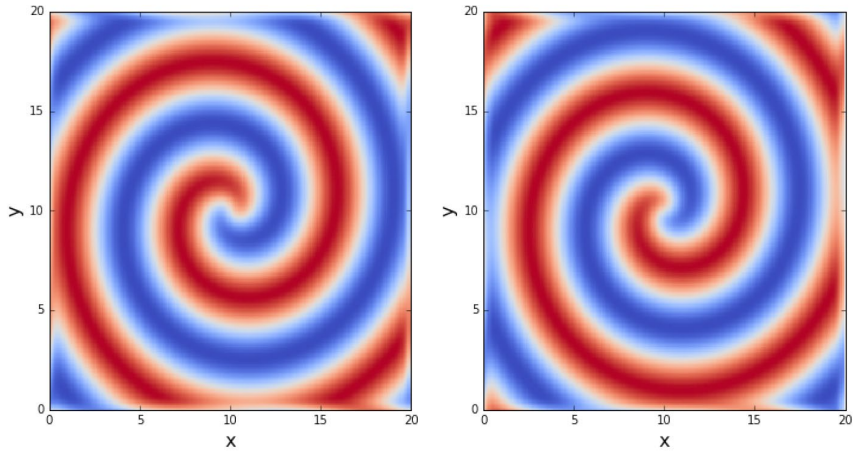


fig. S7. The numerical solution to the reaction-diffusion equation plotted in space-time.

The reaction diffusion equation was much less robust to noise. Even at 1%, PDE-FIND misidentifies the terms in the PDE. The true model, as well as identified models with clean data, 0.5% noise, and 1% noise are given below. Note that coefficients for clean data are inexact but have been rounded to three decimal points.

$$\left.\begin{aligned} u_t &= 0.1u_{xx} + 0.1u_{yy} - uv^2 - u^3 + v^3 + u^2v + u \\ v_t &= 0.1v_{xx} + 0.1v_{yy} - uv^2 - u^3 - v^3 - u^2v + v \end{aligned}\right\} \text{True model}$$

$$\left.\begin{aligned} u_t &= 0.100u_{xx} + 0.100u_{yy} - 1.000uv^2 - 1.000u^3 + 1.000v^3 + 1.000u^2v + 1.000u \\ v_t &= 0.100v_{xx} + 0.100v_{yy} - 1.000uv^2 - 1.000u^3 - 1.000v^3 - 1.000u^2v + 1.000v \end{aligned}\right\} \text{Clean Data}$$

$$\left.\begin{aligned} u_t &= 0.095u_{xx} + 0.095u_{yy} - 0.945uv^2 - 0.945u^3 + 1.000v^3 + 1.000u^2v + 0.945u \\ v_t &= 0.095v_{xx} + 0.095v_{yy} - 1.000uv^2 - 1.000u^3 - 0.946v^3 - 0.946u^2v + 0.946v \end{aligned}\right\} \text{0.5\% Noise}$$

$$\left.\begin{aligned} u_t &= 0.077u_{xx} + 0.077u_{yy} - 0.731uv^2 - 0.732u^3 + 0.810v^3 + 0.810u^2v + 0.776u + 0.169v \\ v_t &= 0.081v_{xx} + 0.079v_{yy} - 0.832uv^2 - 0.832u^3 - 0.772v^3 - 0.772u^2v + 0.815v - 0.149u \end{aligned}\right\} \text{1\% Noise}$$

Limited data

Average parameter error as percentage of true value

Spatial Points: n	Temporal Points: m = 256	128	64	32	16
512	0.113	0.153	0.896	2.446	
256	0.777	0.509	0.238		
128	3.417	3.140		1.161	
64	13.72				
32					

table S9. Accuracy of PDE-FIND on Burger's equation with various grid sizes. Red table entries denote a misidentification of the sparisty pattern either due to the inclusion of extra terms or missing one of the two terms in Burgers' equation. Blue entries show average parameter error as percent of true value. Measurements on all grids were taken from numerical solution on fine grid to ensure error in the method is intrinsic to PDE-FIND and not the numerical solution of Burgers' equation.

Summary (both ODE & PDE discovery)

1. Collect **data** (discretized)
2. Calculate derivatives
3. Use the calculated derivatives to construct a **function library**
4. Represent our PDE for a discretized dataset with a **linear equation**
5. **Sparse regression**: Use an optimization method to find the sparsest choices of functions that describes the dynamics

Comparison

	PINN	SINDy, PDE-FIND
Application	Physics-informed interpolation, inverse problems (obtain unmeasurable quantities)	Discovering equations
Model	NN (x)	Linear model
Parameters to find	w, b	ξ
Loss function	Data loss + Equation loss	Least square + regularization term
Optimization method	Gradient descent, Adam…etc	Sparse regression: LASSO, STLS…etc
Information needed	Equation (but can have free parameters)	Choice of a function library
Extrapolation power	:)	:)

References

Equation discovery

Discovering governing equations from data by sparse identification of nonlinear dynamical systems SL Brunton, JL Proctor, JN Kutz Sign in
Proceedings of the national academy of sciences 113 (15), 3932-3937
1479 2016

Data-driven discovery of partial differential equations SH Rudy, SL Brunton, JL Proctor, JN Kutz Sign in
Science Advances 3 (4), e1602614
644 2017

PINN

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations M Raissi, P Perdikaris, GE Karniadakis Sign in
Journal of Computational Physics 378, 686-707
2085 * 2019

Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations
M Raissi, A Yazdani, GE Karniadakis Sign in
Science 367 (6481), 1026-1030
383 * 2020