DeepMoD

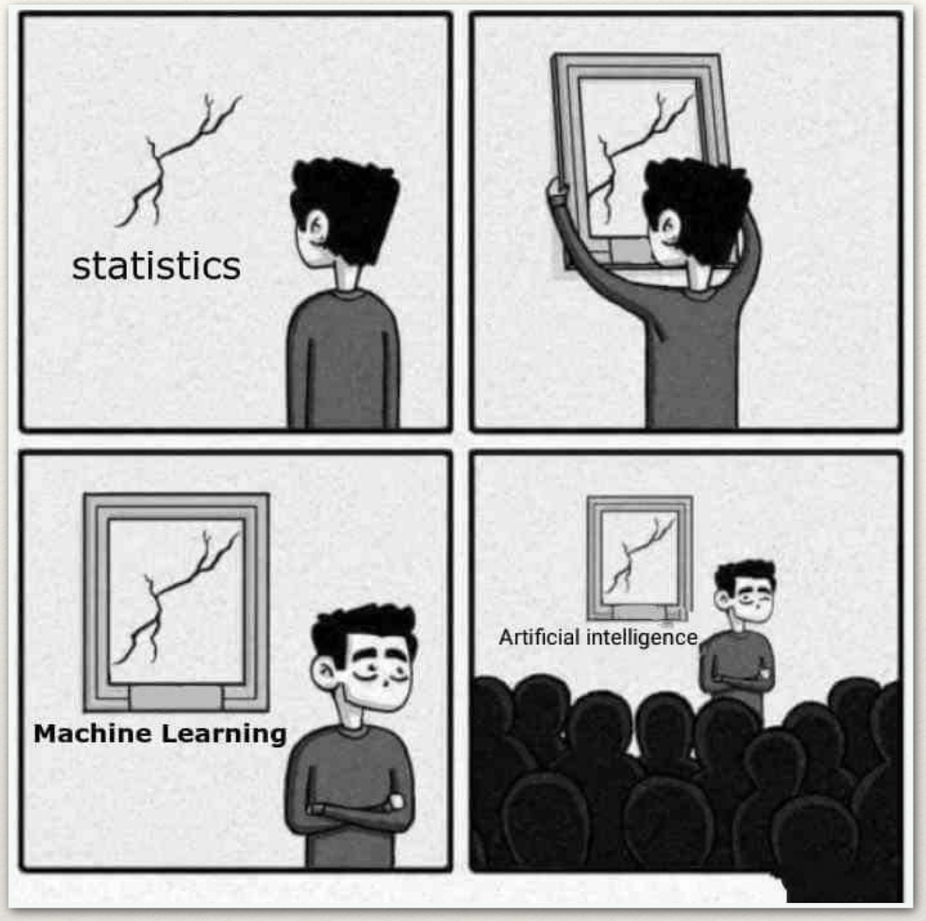
Model discovery with neural networks

Gert-Jan Both, Remy Kusters

CRI Research, Université Paris Descartes, Paris, France

@GJ_Both
@RemyKusters

AI, ML and statistics



@sandserif

Main criticisms …

ML provides uninterpretable solutions

Statistical ML vs. Symbolic ML

Goal:

Develop tools to discover **interpretable** models for quantitative science

Approach:

Combine the **statistical power of neural networks** with the **symbolic power of regression**

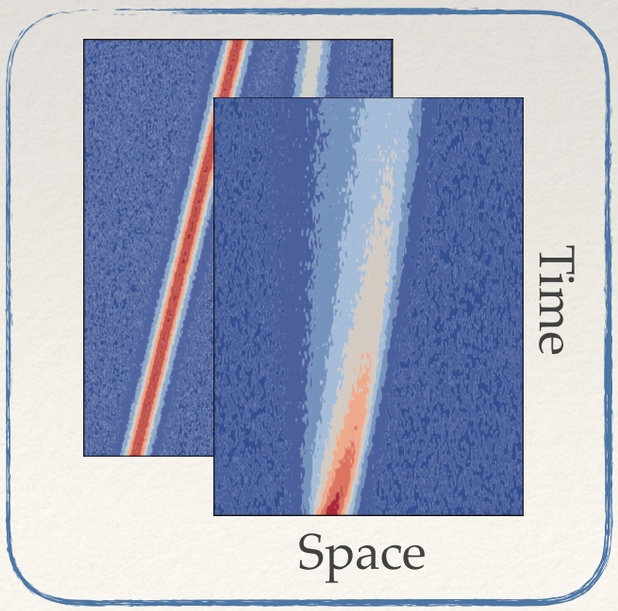
Goal:

Develop tools to discover **interpretable** models for quantitative science

Approach:

Combine the **statistical power of neural networks** with the **symbolic power of regression**

Noisy Data



Differential equation

$$u_t(x,t) = \mathcal{F}(u, u_x, uu_x, u_{xx}, ...)$$

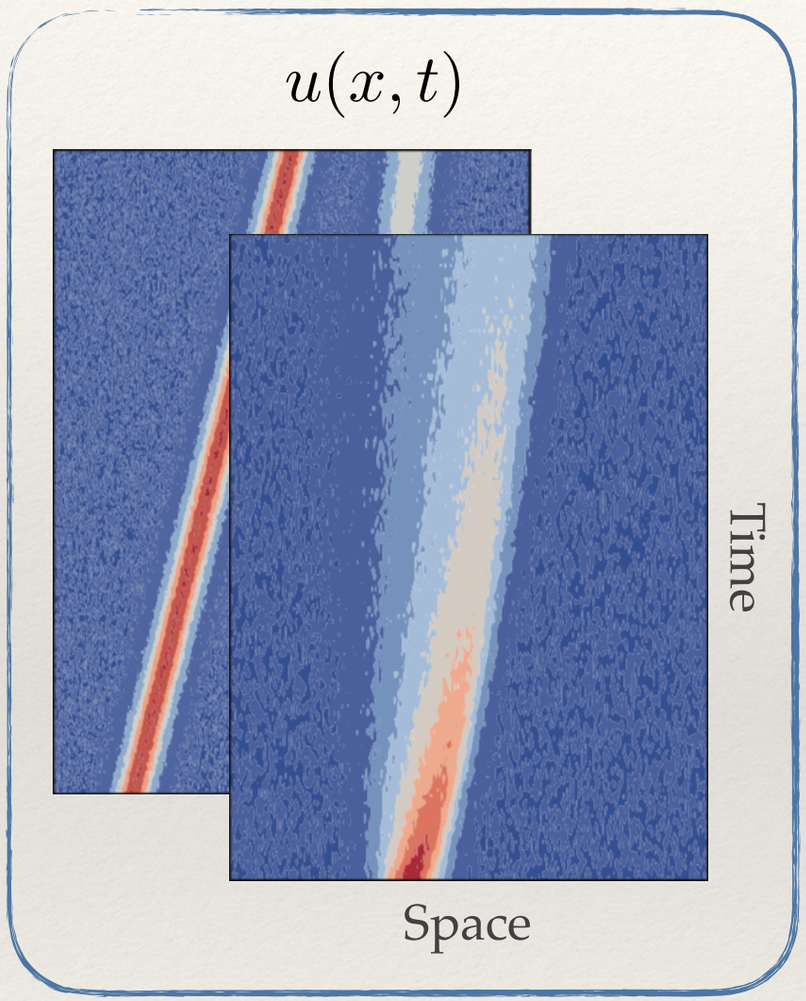
Underlying distribution

$$P(x,t)$$

DeepMoD

Model discovery with neural networks

Noisy Data



Analytic solution

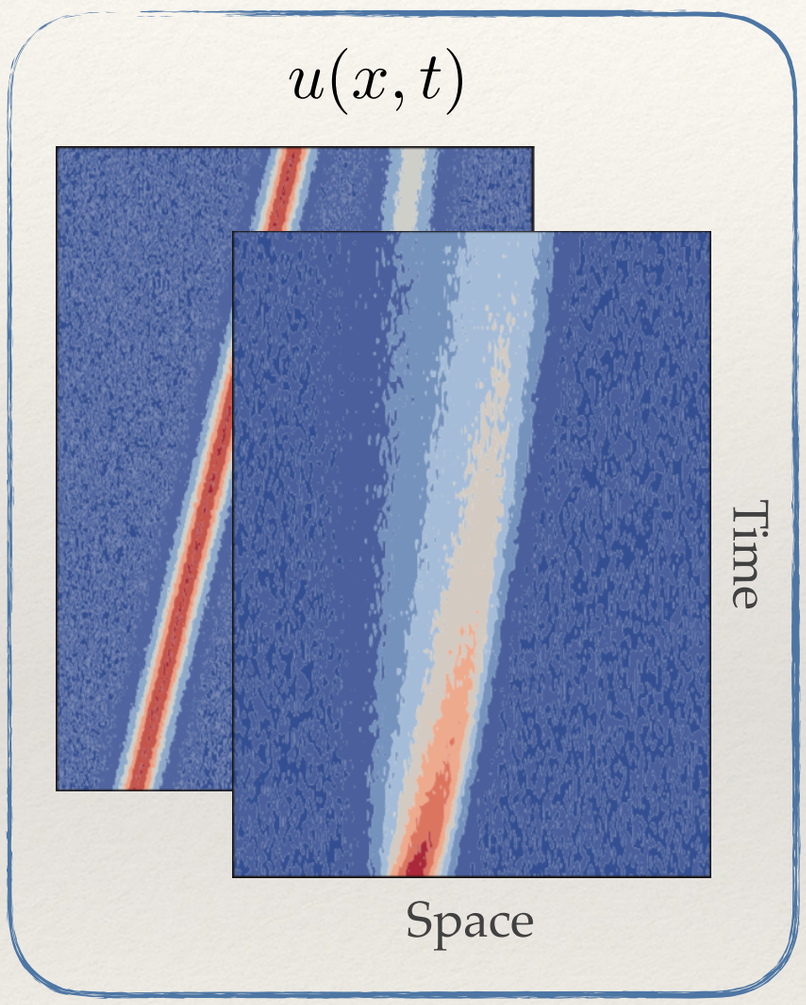
$$u_t(x,t) = \mathcal{F}(u, u_x, uu_x, u_{xx}, ...)$$

Code available:
github.com/PhIMaL

DeepMoD

Model discovery with neural networks

Noisy Data



Solution

$$u_t(x,t) = \Theta\vec{\xi}$$

$$\begin{bmatrix} \\ \\ \end{bmatrix} = \begin{bmatrix} u \\ u_x \\ uu_x \\ \vdots \\ u^2u_{xx} \end{bmatrix} \begin{bmatrix} \\ \\ \end{bmatrix}$$

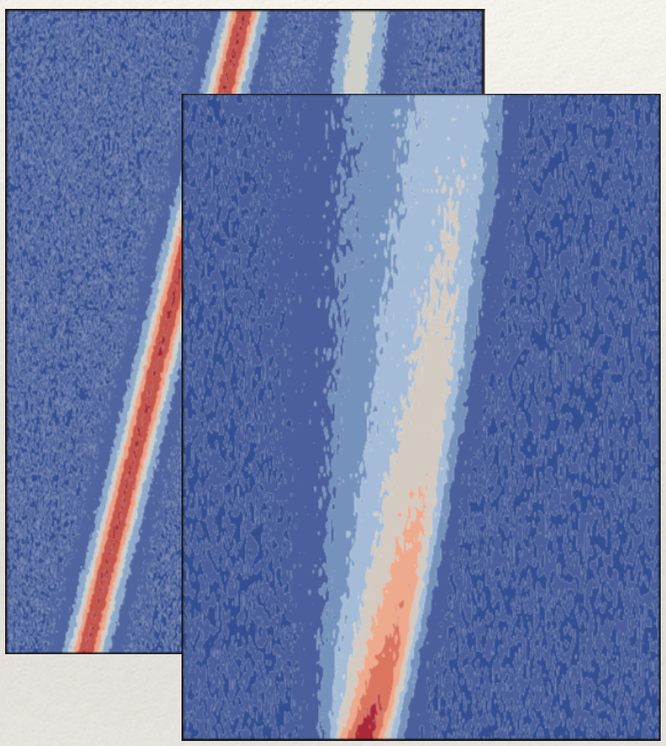
Code available:
github.com/PhIMaL

DeepMoD

Model discovery with neural networks

Noisy Data

$u(x,t)$



Regression

$$u_t(x,t) = \Theta\vec{\xi}$$

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} u & u_x & uu_x & \cdots & u^2u_{xx} \end{bmatrix}$$

Solution

$$u_t(x,t) = \Theta\vec{\xi}$$

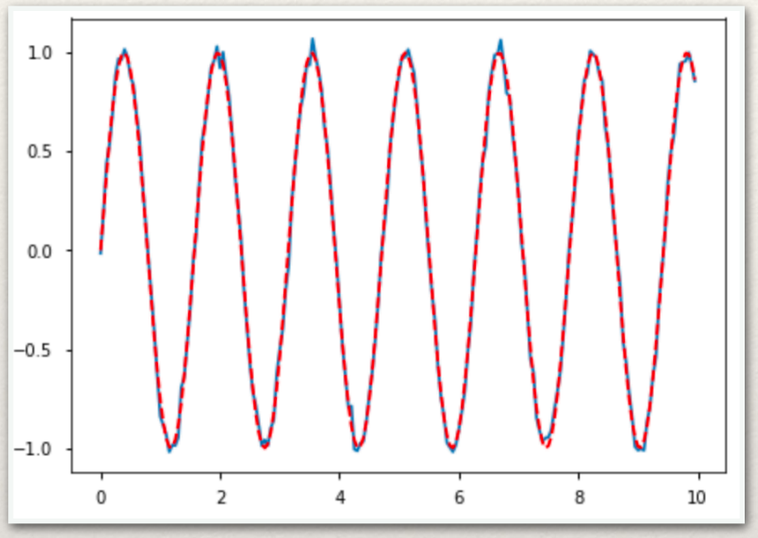
$$\xi^* = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ -0.1 \\ 0 \end{bmatrix} \begin{matrix} - u \\ - u_x \\ - uu_x \\ \\ \vdots \\ - u_{xx} \\ - u^2u_{xx} \end{matrix}$$

Code available:
github.com/PhIMaL

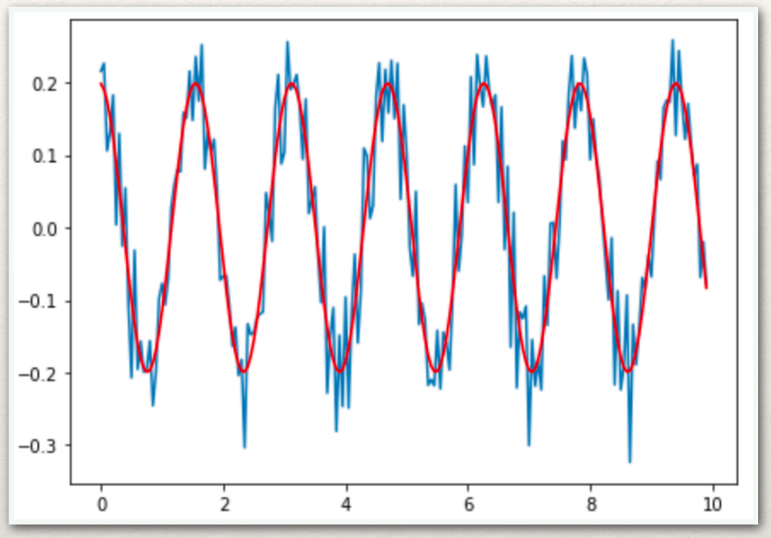
Why does it fail with noise?

5% white noise …

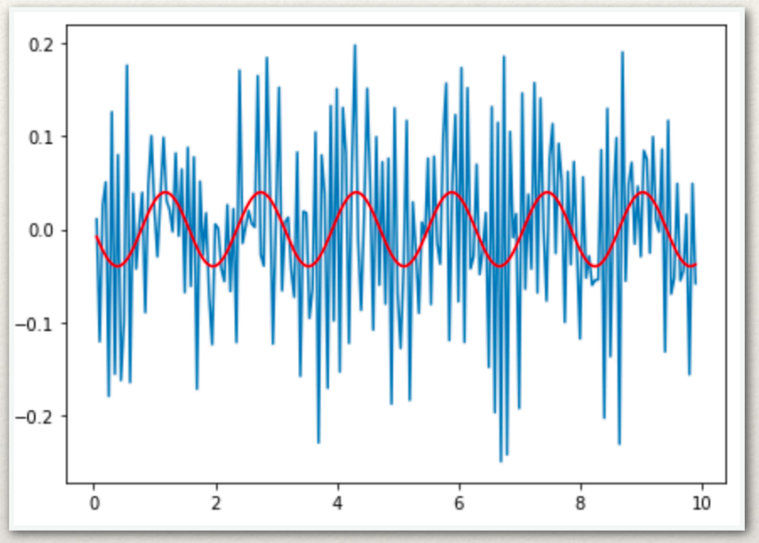
$u(x)$



$u_x(x)$



$u_{xx}(x)$

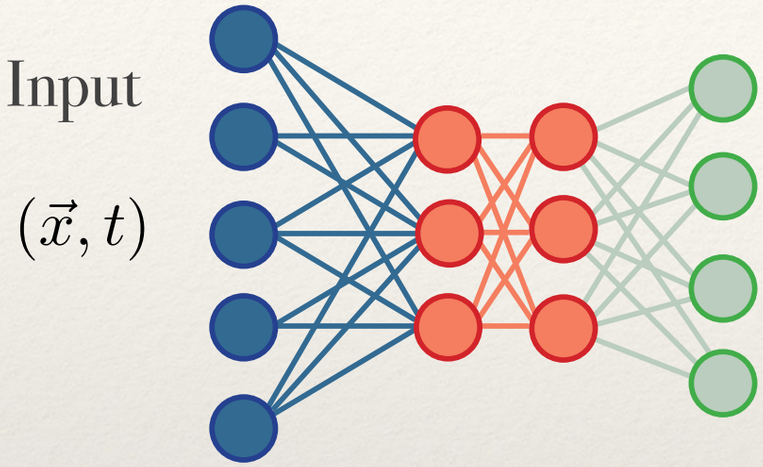


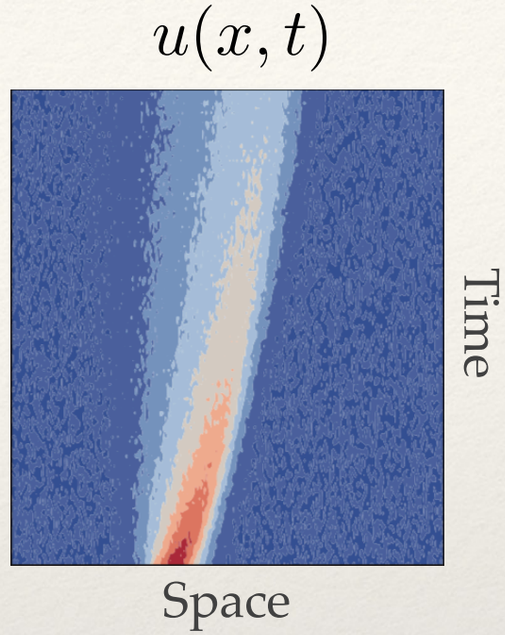
$$\Theta = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ u & u_x & u_x u & u_{xx} & u_{xx} u & u_{xx} u_x & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

Numerically calculating library impossible …

Our approach …

The Neural Network





Construct the library w.r.t. **the inferred solution**

$$\Theta = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \hat{u} & \hat{u}_x & \hat{u}_x\hat{u} & \hat{u}_{xx} & \hat{u}_{xx}\hat{u} & \hat{u}_{xx}\hat{u}_x & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

Include the **regression task** within the neural network

How do we train a neural network

Optimise loss function:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{Reg} + \mathcal{L}_{L_1}$$

Approximate the data as good as possible ...

Learn the mapping
$(\vec{x}, t) \rightarrow \vec{u}$

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} |u(\{x,t\}_i) - \hat{u}_i|^2$$

How do we train a neural network

Optimise loss function:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{Reg} + \mathcal{L}_{L_1}$$

Approximate the data as good as possible ...

Learn the mapping
$(\vec{x}, t) \rightarrow \vec{u}$

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{i=1}^{N} |u(\{x,t\}_i) - \hat{u}_i|^2$$

Discover the differential equation

Regression
(Seek the PDE)

$$\mathcal{L}_{Reg} = \frac{1}{N} \sum_{i=1}^{N} |\Theta_{ij}\xi_j - \partial_t \hat{u}_i|^2$$

How do we train a neural network

Optimise loss function:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{Reg} + \mathcal{L}_{L_1}$$

Approximate the data as good as possible ...

Learn the mapping
$(\vec{x}, t) \rightarrow \vec{u}$

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N} |u(\{x,t\}_i) - \hat{u}_i|^2$$

Discover the differential equation

Regression
(Seek the PDE)

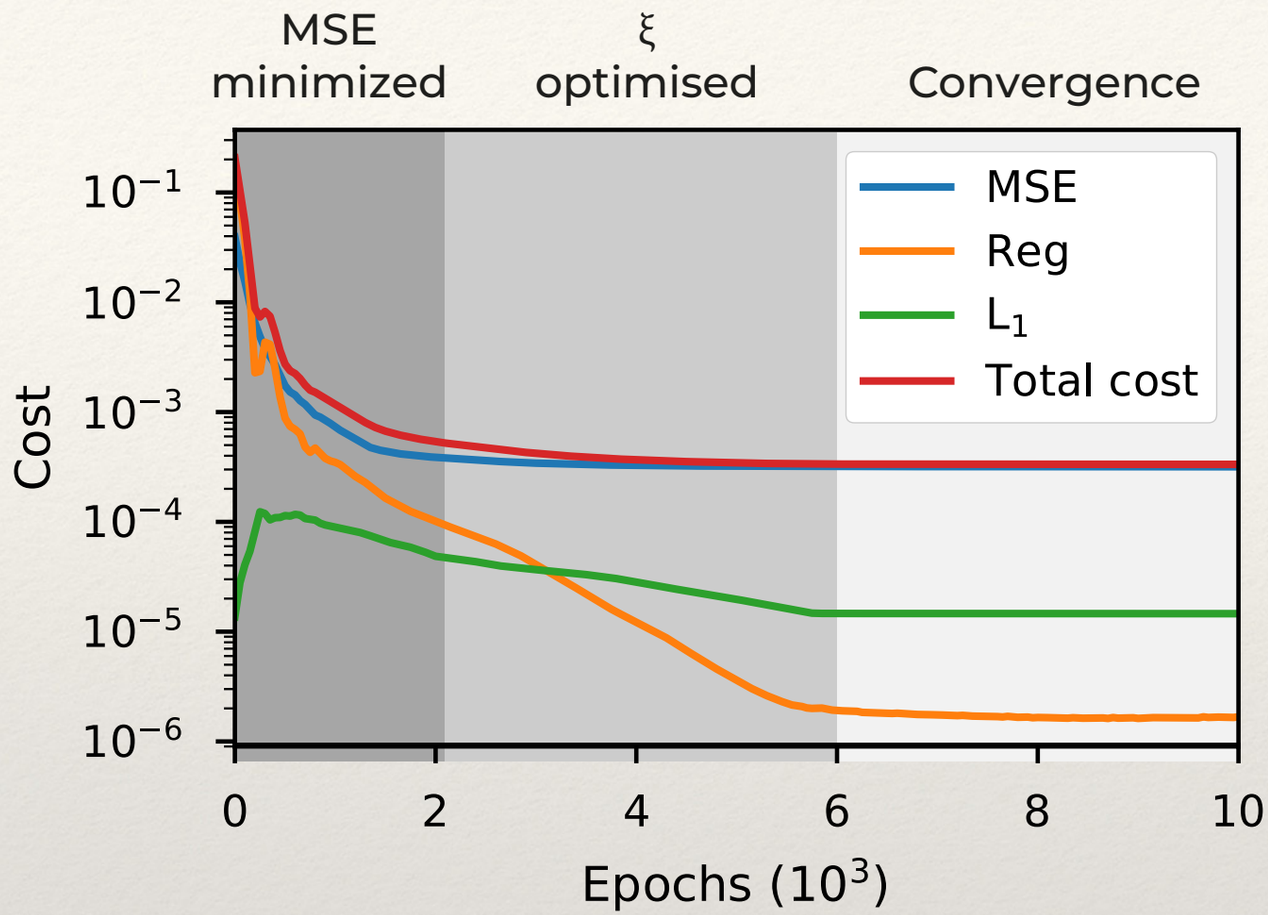
$$\mathcal{L}_{Reg} = \frac{1}{N}\sum_{i=1}^{N} |\Theta_{ij}\xi_j - \partial_t \hat{u}_i|^2$$

Promoting sparsity

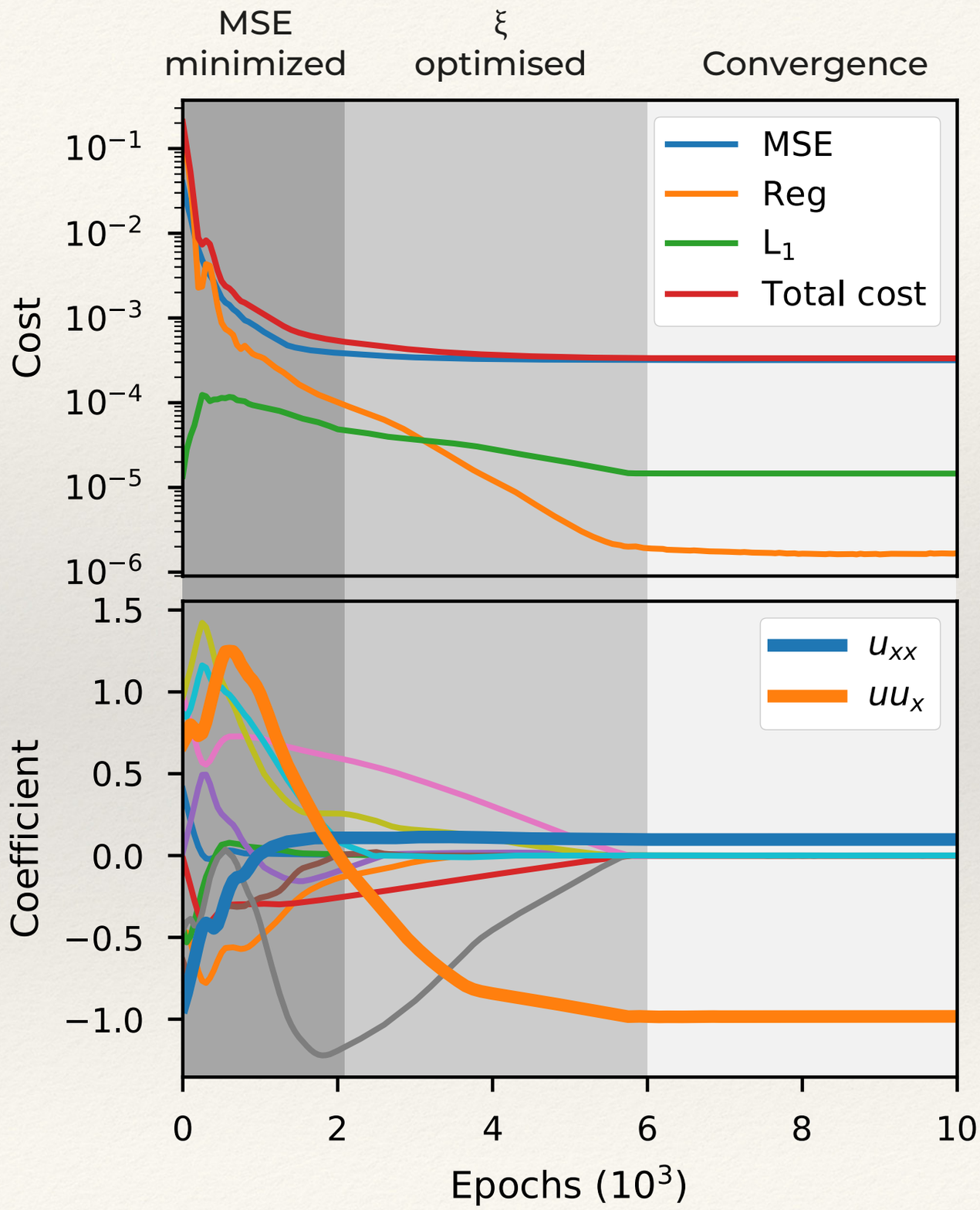
L1 Penalty

$$\mathcal{L}_{L_1} = \lambda \sum_{i=2}^{M} |\xi^i|$$

Training the neural network



Training the coefficient vector



$$u_t(x,t) = \Theta\vec{\xi}$$

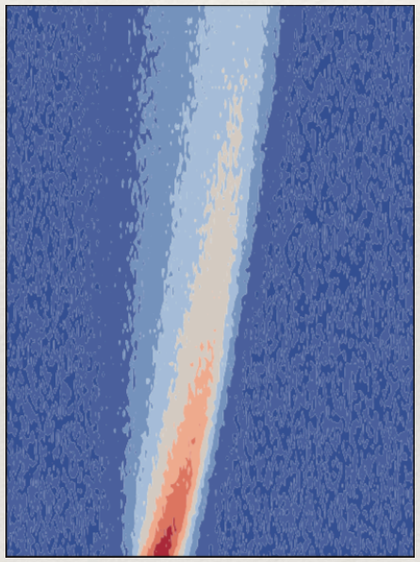
$$\xi^* = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ -0.1 \\ 0 \end{bmatrix} \begin{matrix} - \; u \\ - \; u_x \\ - \; uu_x \\ \\ \\ - \; u_{xx} \\ - \; u^2 u_{xx} \end{matrix}$$

DeepMoD

Model discovery with neural networks

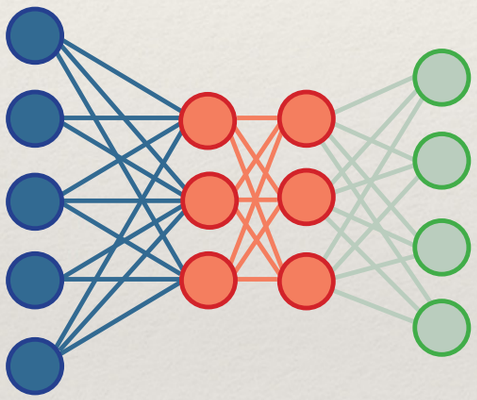
Noisy Data

$u(x,t)$



Neural network

$(\vec{x},t)$ → $\hat{u}$



$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{Reg} + \mathcal{L}_{L_1}$$

Regression

$$u_t(x,t) = \Theta\vec{\xi}$$

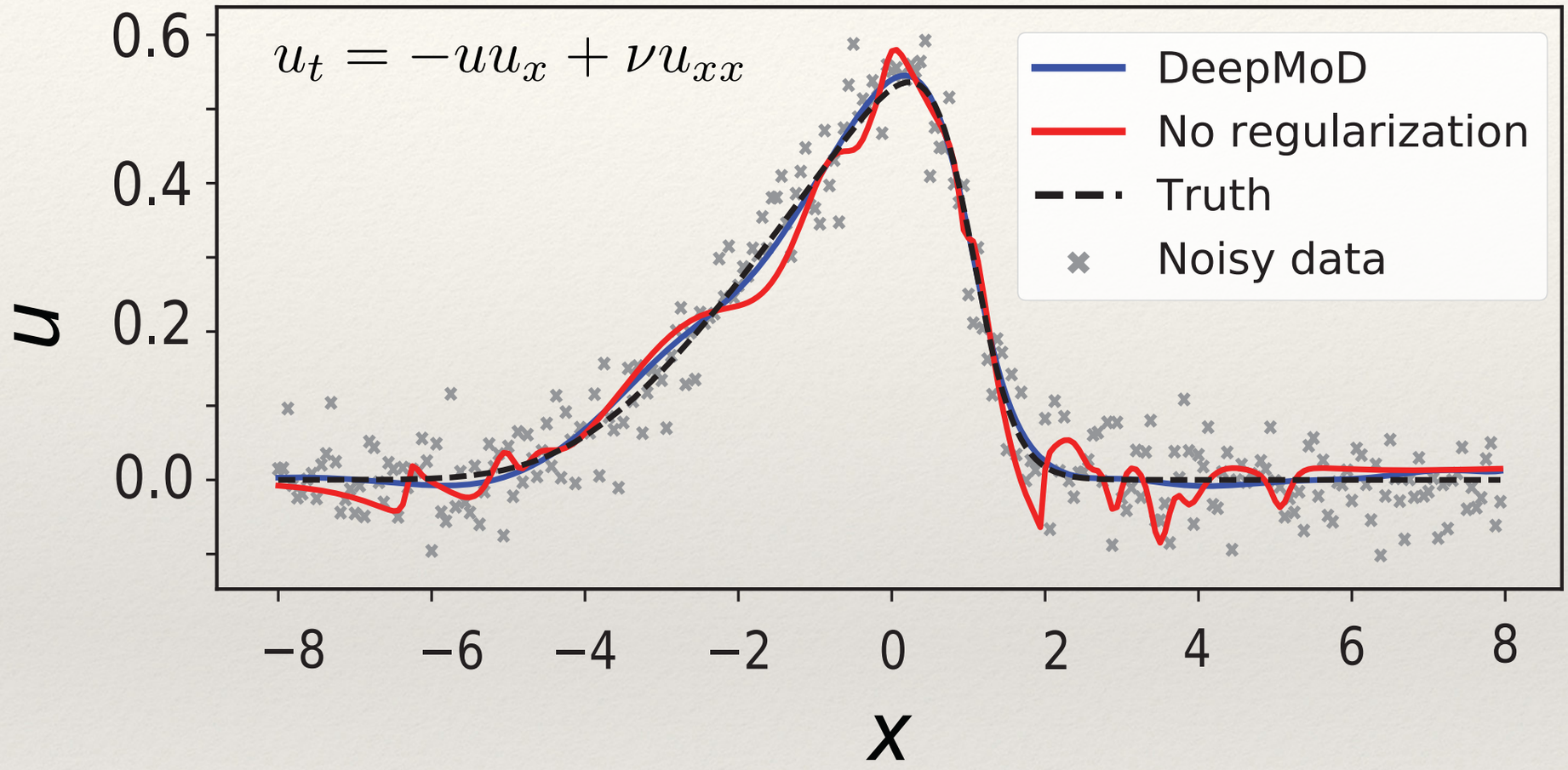
$$\begin{bmatrix} \\ \\ \\ \end{bmatrix} = \begin{bmatrix} u & u_x & uu_x & \cdots & u^2u_{xx} \end{bmatrix}$$

Solution

$$\xi^* = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ -0.1 \\ 0 \end{bmatrix} \begin{matrix} - u \\ - u_x \\ - uu_x \\ \\ \\ - u_{xx} \\ - u^2u_{xx} \end{matrix}$$

Code available:
github.com/PhIMaL

Why regression inside neural network?



Applications in physics

Resilience to Noise/ sampling size

Burgers' equation

$$u_t = -uu_x + \nu u_{xx}$$

Relative error on coefficients

Higher order PDEs

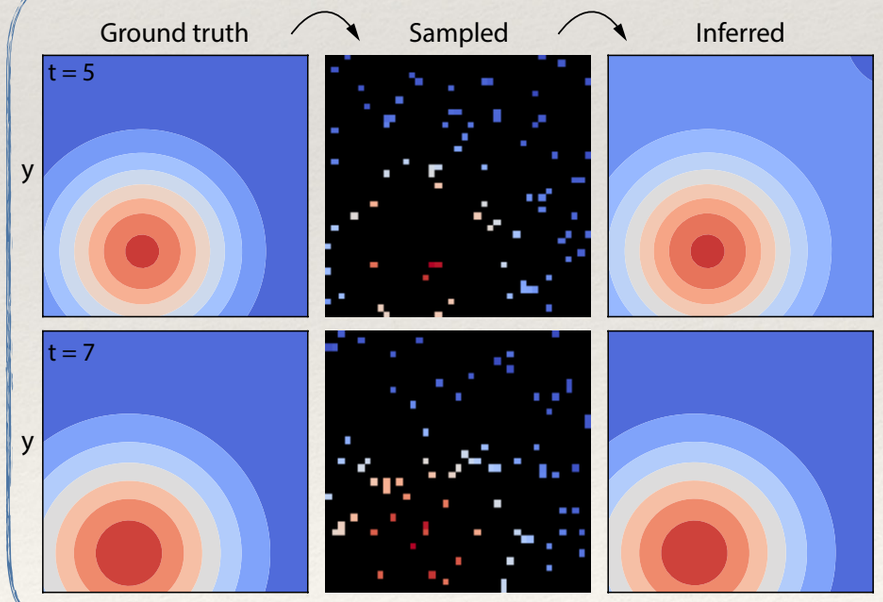
$$u_t = 6uu_x - u_{xxx}$$

Coupled equations

$$u_t = \nabla \cdot (D_i \nabla u - \xi u \nabla w) \quad u \text{ :particle density}$$

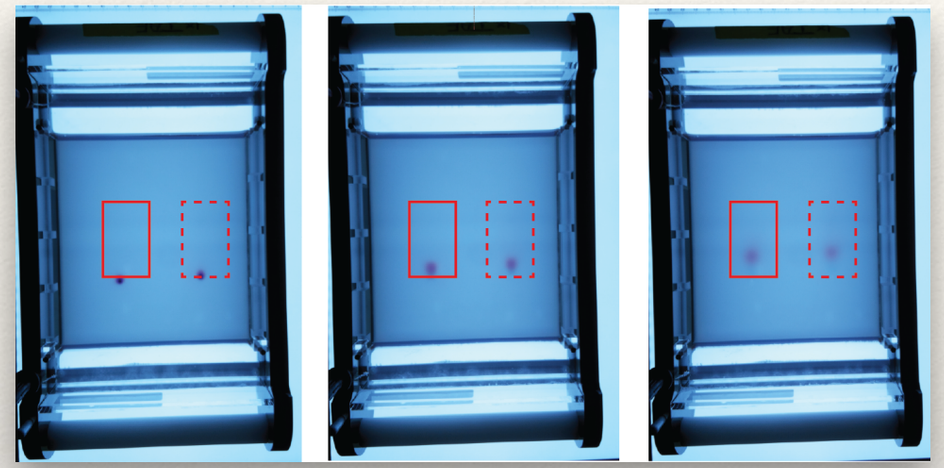
$$w_t = D_w \Delta w - kw + hu \quad w \text{ :attractant density}$$

2D equations

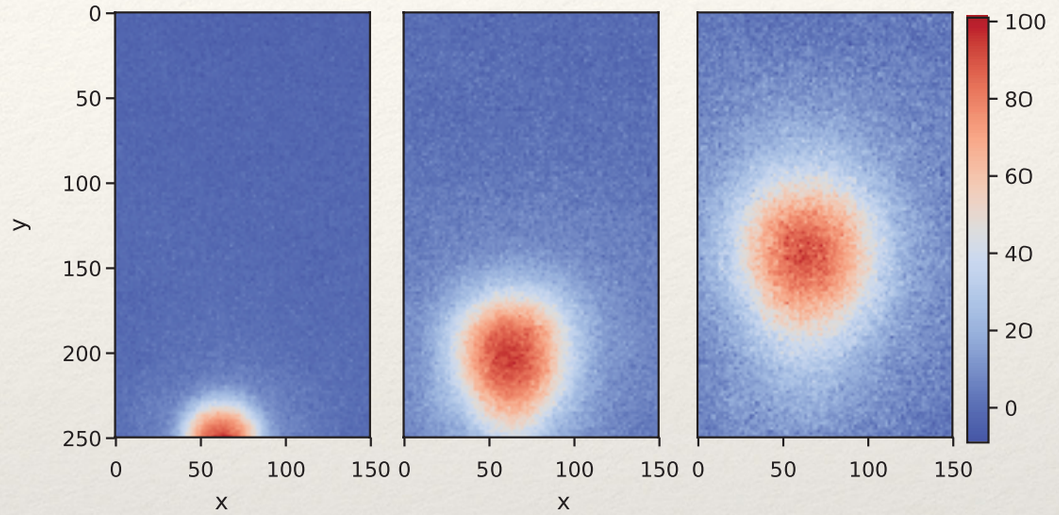


Apply to real data …

Raw images



Density field



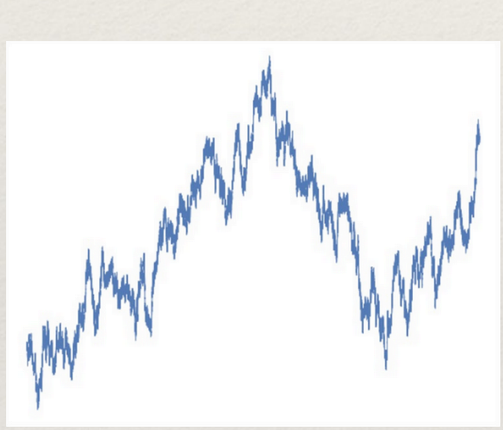
Underlying AD equation

$$u_t = 0.3u_y + 0.01u_{xx} + 0.01u_{yy}$$

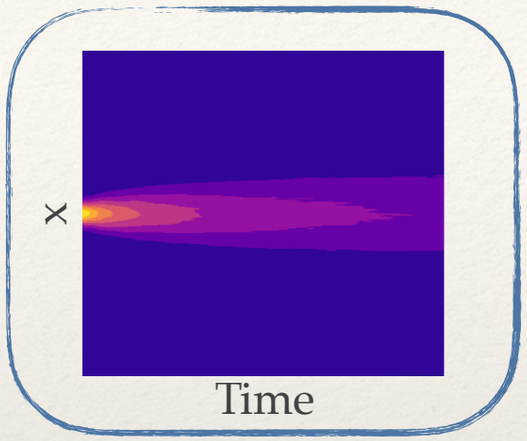
Model discovery from **experimental data**

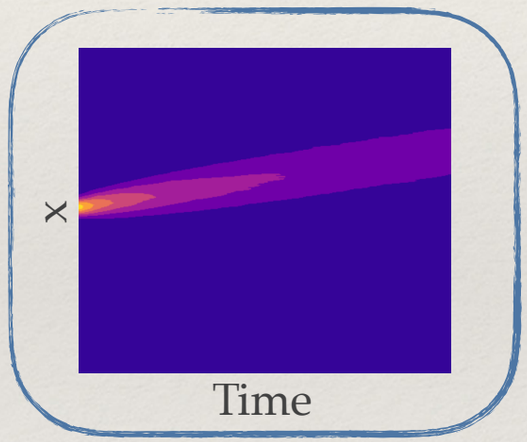
Single Particle Tracking

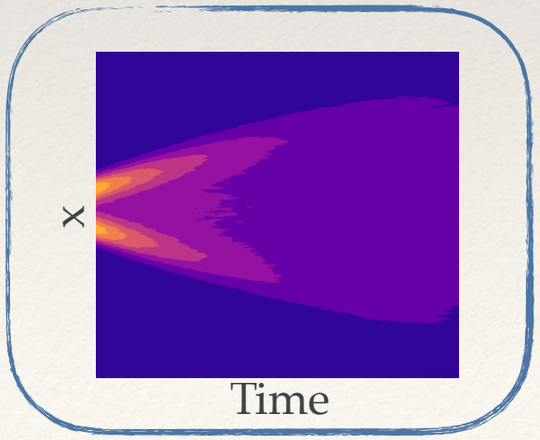
Trajectory



Density estimation







Model discovery

Random walk

$$u_t = Du_{xx}$$

Advected random walk

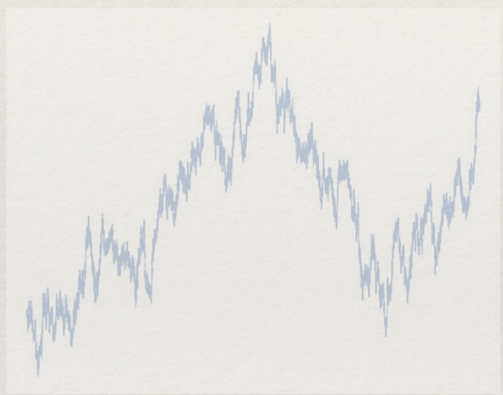
$$u_t = Du_{xx} + v_d u$$

Persistant random walk

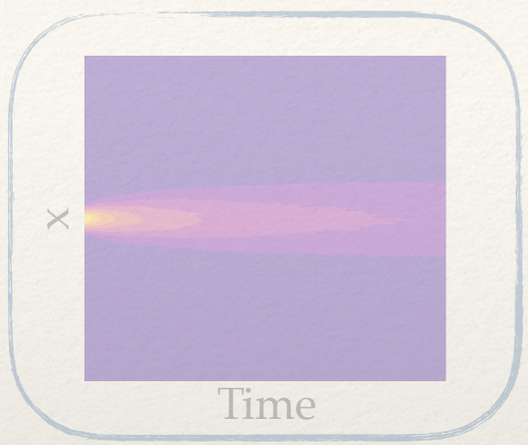
$$u_{tt} + \frac{1}{\tau_c} u_t = \nu^2 u_{xx}$$

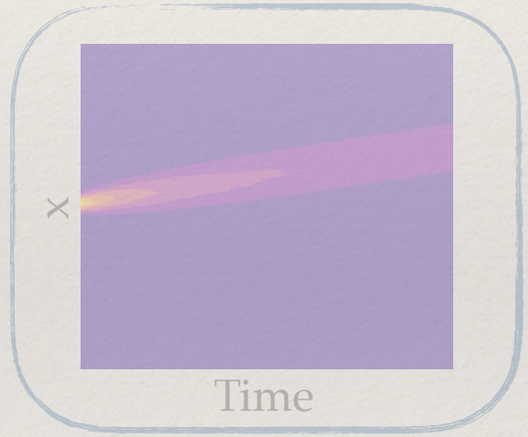
Single Particle Tracking

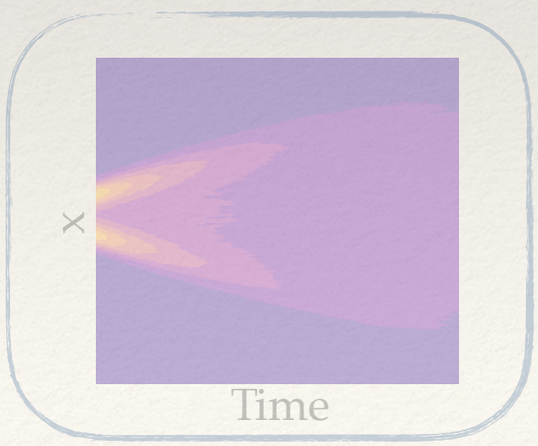
Trajectory



Density estimation







Model discovery

Random walk

$$u_t = Du_{xx}$$

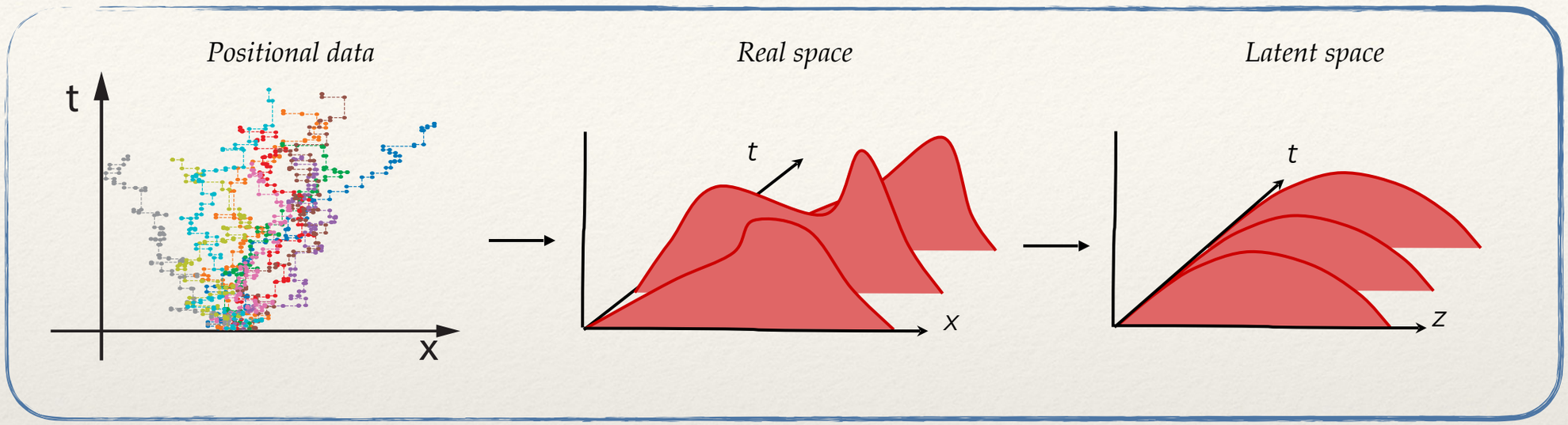
Advected random walk

$$u_t = Du_{xx} + v_d u$$

Persistant random walk

$$u_{tt} + \frac{1}{\tau_c} u_t = \nu^2 u_{xx}$$

Density estimation: Temporal normalizing flows

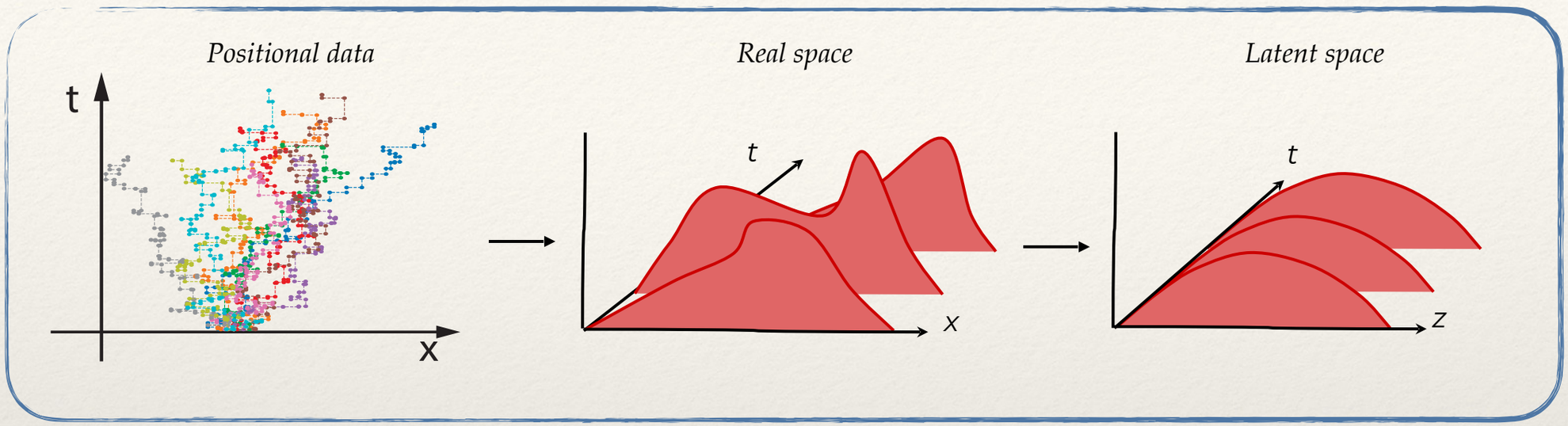


Temporal Normalizing Flows (tNFs) extend NFs with a temporal component to estimate a **time-dependent density evolution**

Code available on Github:
https://github.com/PhIMaL

Paper available on arXiv:
1912.09092

Density estimation: Temporal normalizing flows



Temporal Normalizing Flows (tNFs) extend NFs with a temporal component to estimate a **time-dependent density evolution**

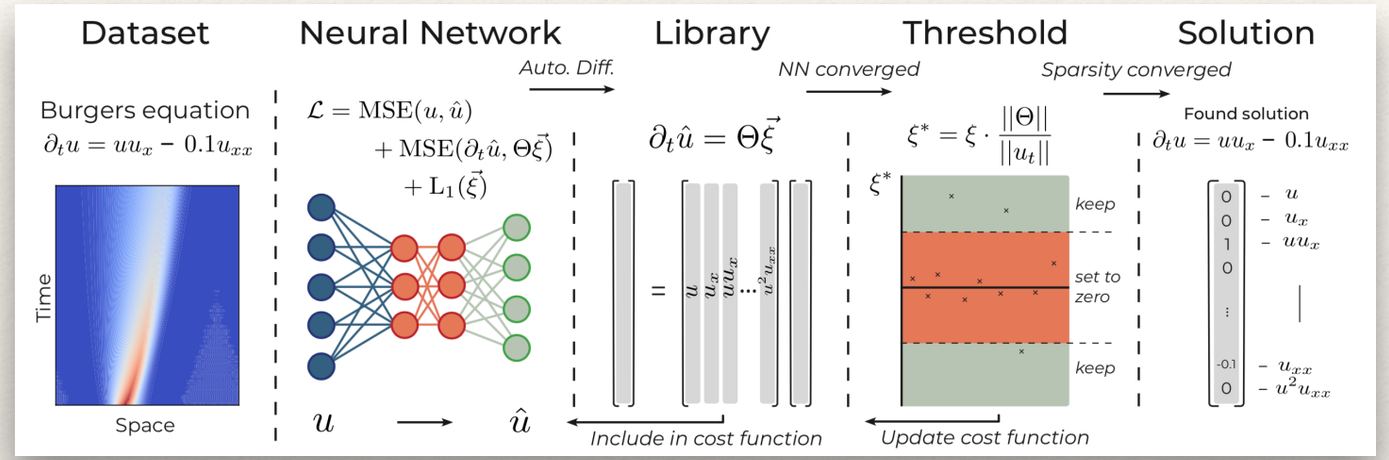
tNFs allow discovery of PDEs with conservation constraints, e.g. Fokker-Planck, ...

Code available on Github:
https://github.com/PhIMaL

Paper available on arXiv:
1912.09092

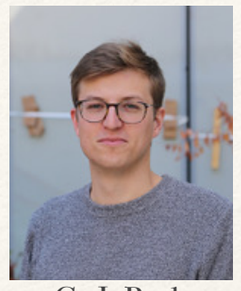
DeepMoD

Model discovery with neural networks

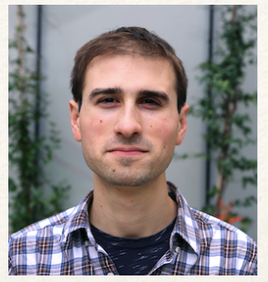


Code available on Github:
https://github.com/PhIMaL

Paper available on arXiv:
1904.08406



G.-J. Both
(PhD)



A, Brandon-Bravo
(Engineer)

C. Le Scao
(Intern)

T. Chrysostomou
(Intern)

@GJ_Both
@RemyKusters

TensorFlow

PyTorch

CRI
Catalysing a fruitful future

Inserm
La science pour la santé
From science to health

Postdoc position available !