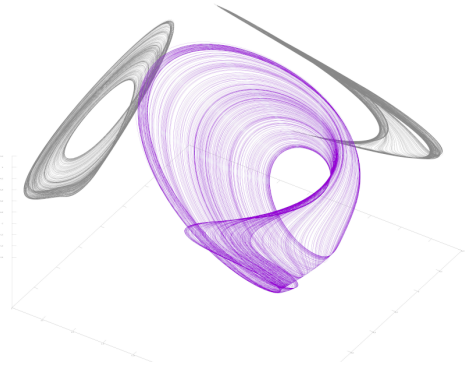
Deep learning for Physics

Lionel MATHELIN
LISN–CNRS

Dataflot / Decipher



M2 MVA 2020-2021

Sommaire

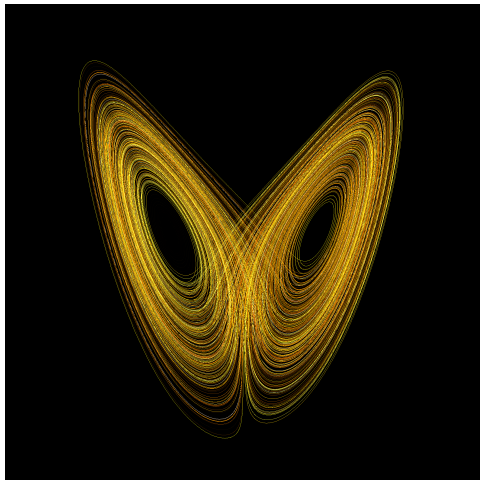
Deep learning for Physics

See how (deep) learning can leverage Physics/applications bottlenecks (online computational time and memory, modeling capability (super regression), etc.).

But also how Physics can help learning (bounds, first principles, speed-up learning, etc.).

A minimally complex system model

Illustration on Lorenz '63 attractor as a proxy good enough for its features.



Simplified mathematical model for atmospheric convection:

$$\dot{x} = \sigma(y - x), \qquad \text{convection rate},$$
$$\dot{y} = x(\rho - z) - y, \qquad \text{horizontal temperature variations},$$
$$\dot{z} = xy - \beta z \qquad \text{vertical temperature variations}.$$

Becomes chaotic for $\rho > \sigma\ (\sigma + \beta + 3) / (\sigma - \beta - 1)$.
Typically: $\sigma = 10$ (Prandtl number), $\beta = 8/3$, $\rho = 28$ (Rayleigh number). Most "esthetic" values among the ones leading to chaos ...
System of nonlinear deterministic ODEs.

One wants to learn to predict from observations:

$$\dot{\boldsymbol{u}} = \boldsymbol{F}(\boldsymbol{u}) \qquad \text{or} \qquad \boldsymbol{u}_{t+\Delta t} = \boldsymbol{F}_{\Delta t}(\boldsymbol{u}_t).$$

Quality of the Information

Get to know your “opponent”!

How much information is necessary to describe the system/dynamics? How big should the training set be? How to design it?

Quality of the Information

Get to know your “opponent”!

How much information is necessary to describe the system/dynamics? How big should the training set be? How to design it?

Different definitions of dimension

The geometry of chaotic attractors can be complex and difficult to describe.
What is the dimension of the attractor of Lorenz’63? In other words, what is the intrinsic dimension of its data manifold?

It is therefore useful to have quantitative characterizations of such geometrical objects.

- Topological dimension $\longrightarrow$ integer,
- Metric dimension $\longrightarrow$ fractional.

Minimum number of independent dynamic variables to define the state of the system (nb degrees of liberty).

Many definitions of dimensions

- **Box Counting Dimension**
  If the attractor exists in a d-dimensional phase space (d is necessarily an integer). Let $N(\epsilon)$ be the number of d-dimensional cubes of edge length ϵ from the grid that are needed to cover the attractor. Then

$$N \propto 1/\epsilon^d, \qquad \Rightarrow \qquad d_0 = \lim_{\epsilon \to 0_+} \frac{\log N(\epsilon)}{\log 1/\epsilon}.$$

Many definitions of dimensions

- **Box Counting Dimension**
  If the attractor exists in a d-dimensional phase space (d is necessarily an integer). Let $N(\epsilon)$ be the number of d-dimensional cubes of edge length ϵ from the grid that are needed to cover the attractor. Then

$$N \propto 1/\epsilon^d, \qquad \Rightarrow \qquad d_0 = \lim_{\epsilon \to 0_+} \frac{\log N(\epsilon)}{\log 1/\epsilon}.$$

- **Rényi dimension**
  Also called the 'generalized dimension', takes into account the frequency with which cubes are visited via weighting them according to their **natural measure** μ, defined as the fraction of the time that the long orbit on the attractor spends in any given ϵ-region C_j of state space.
  Accounts for time spent in cubes.

$$d_q = \lim_{\epsilon \to 0_+} \frac{1}{1-q} \frac{\log \left(\Sigma_j \left(\mu(C_j) \right)^q \right)}{\log(1/\epsilon)}.$$

  Special cases: d_0: 'box counting' or 'capacity dimension', d_1 is 'information dimension', d_2 has been called the 'correlation dimension'.
  $d_2 \leq d_1 \leq d_0$.

Dimension d_2

d_2 has a nice numerical algorithm for its computation (Grassberger & Procaccia) as the slope of the following function when $\epsilon \to 0$: [Also a convenient algo (faster, less noisy with few points)]:

$$d_2 = \lim_{\epsilon \to 0_+} \frac{\log\left(1/N^2 \Sigma_{i \neq j} H\left(\epsilon - \|x_i - x_j\|\right)\right)}{\log \epsilon} \qquad \text{with } H \text{ the Heaviside function.}$$

Numerator is ratio of samples such that their distances is $< \epsilon$.

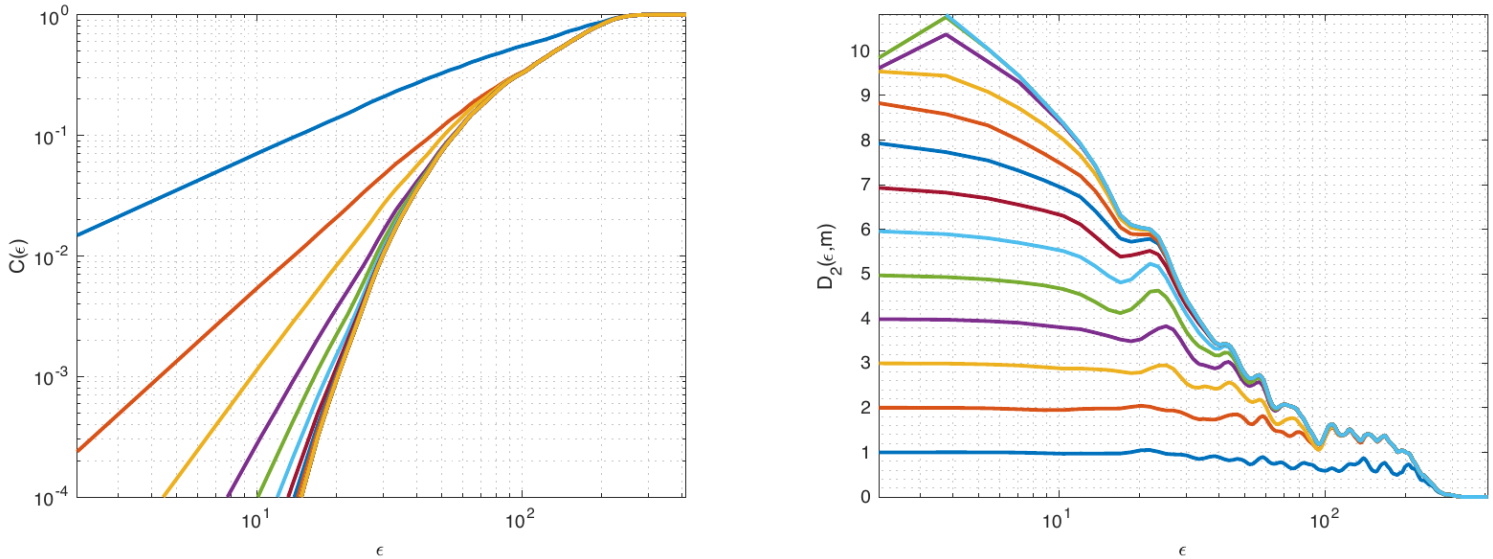


Figure: Illustration of the estimation of the d_2 dimension by varying the threshold ϵ.

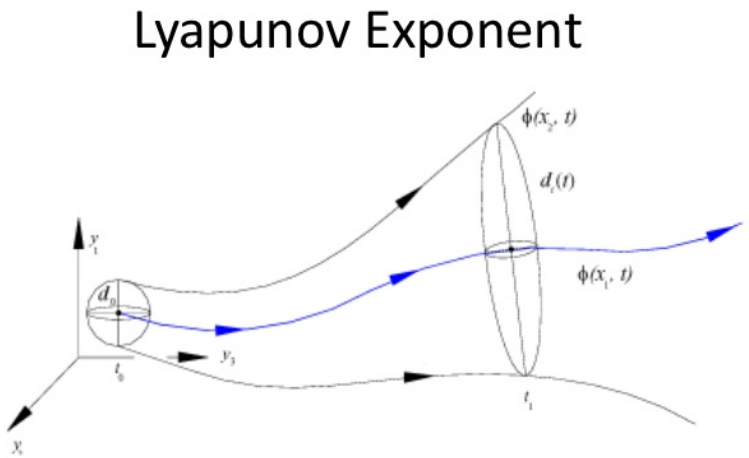
Kaplan-Yorke dimension

It is defined in terms of the **Lyapunov exponents**.

Illustrative example in 1-D

$$
\begin{aligned}
u_n &= f(u_{n-1}), \\
\epsilon_n &= f(u_{n-1}+\epsilon_{n-1}) - f(u_{n-1}), \qquad \left|\frac{\epsilon_n}{\epsilon_0}\right| = \prod_i \left|\frac{\epsilon_i}{\epsilon_{i-1}}\right| = \prod_i \left|f'(\boldsymbol{u}_{i-1})\right|, \\
\epsilon_n &= \exp(\lambda\, n)\,\epsilon_0 \qquad \Rightarrow \quad \lambda = \lim_{n\to\infty} \frac{1}{n}\sum_i^n \ln\left|f'(\boldsymbol{u}_{i-1})\right|.
\end{aligned}
$$

This extends to dynamical systems via singular values of the mean Jacobian of the flow $\phi^t : \boldsymbol{u}_0 \to \phi^t(\boldsymbol{u}_0) = \boldsymbol{u}_t$.



- $|\delta \mathbf{Z}(t)| = e^{\lambda t}|\delta \mathbf{Z}(0)|$

Kaplan-Yorke dimension

For Lorenz'63: $\lambda = (0.90, 0.00, -14.57)$

The 'Lyapunov dimension' d_L or d_{KY} is given by

$$d_{KY} = k + \frac{\lambda_1 + \lambda_2 + \ldots + \lambda_k}{|\lambda_{k+1}|},$$

where k is the maximum value of i such that $\lambda_1 + \ldots + \lambda_i > 0$, $\lambda_1 \geq \lambda_2 \geq \ldots$.

The Kaplan-Yorke conjecture states that $d_1 = d_{KY}$ for 'typical' systems, that is, the information dimension is equal to the Lyapunov dimension.

This relationship is remarkable in that it relates dynamics (Lyapunov exponents) to attractor geometry and natural measure (d_1).

Hausdorff dimension

Let $N(r)$ be the minimal number of open balls of radius r necessary to cover a compact metric space X. If, when $r \to 0$, $N(r)$ grows like $\frac{1}{r^d}$, the space X is said of dimension d.

More precisely, d is the number such that $N(r)r^s \to 0$ if $s > d$, and $N(r)r^s \to \infty$ if $s < d$:

$$\begin{aligned} H^s(X) &= \lim_{r\to 0} \inf_{\mathrm{diam}(A_i)<r} \left\{ \sum_{i=1}^{\infty} \mathrm{diam}(A_i)^s \,\middle|\, X \subseteq \bigcup_{i=1}^{\infty} A_i \right\}, \\ d_H(X) &= \inf\left\{ s \mid H^s(X) = 0 \right\} = \sup\left\{ s \mid H^s(X) = \infty \right\}. \end{aligned}$$

For Lorenz'63: $d_2 = 2.05 \pm 0.01$, $d_H = 2.06 \pm 0.01$.

Considerations on the size of the training set

Ergodic theory

Founding idea is that the long-term statistics of a system are equivalently described in terms of invariant measure (independent of time), μ.

The Poincaré recurrence theorem guarantees that trajectories exiting a region $S \in \Omega$ of non-vanishing measure will return to it infinitely many times.

Let S be an ensemble including a point $\boldsymbol{u}_0$ of the attractor. For an ergodic system, the **mean recurrence time** for a given S is proportional to the measure of S:

$$\left\langle \tau_{S(\boldsymbol{u}_0)} \right\rangle_{\mu_S} \propto \frac{1}{\mu(S)} \sim \epsilon^{-d_A}, \qquad \text{with} \qquad S = B_{\boldsymbol{u}_0}^{d_{\text{ambient}}}(\epsilon).$$

$\longrightarrow$ the mean recurrence time then becomes large when ϵ is small (# **accuracy**) and/or d_A is large: "**Curse of dimensionality**".

$\longrightarrow$ If one wants a uniform learning accuracy of order ϵ (**sampling of the natural measure**), the number of samples to be considered is then $N \sim \epsilon^{-d_A}$.

One could use a less severe bound, e.g., $N \gtrsim (1/\epsilon)^{d/2}$ suggested by Eckmann & Ruelle (1992).

Black-box approach

Structure-based consideration

If the structure was known, need to identify $\{\sigma, \beta, \rho\}$ only $\longrightarrow N \geq 3$ (noiseless)!

Physics/expertise is in the structure, not the data!

White-box approach

A grey-box approach

Size of the training set is partially conditioned by the minimum complexity of

- the physical system at hand ($\sim \epsilon^{-d_A}$),
- the structure of the retained model ($\sim$ # parameters of the network).

Choosing a model structure considerably restricts (the dimension of) the solution space and reduces the required size of the training set.

How to sample? Which observations?

A numerical experiment

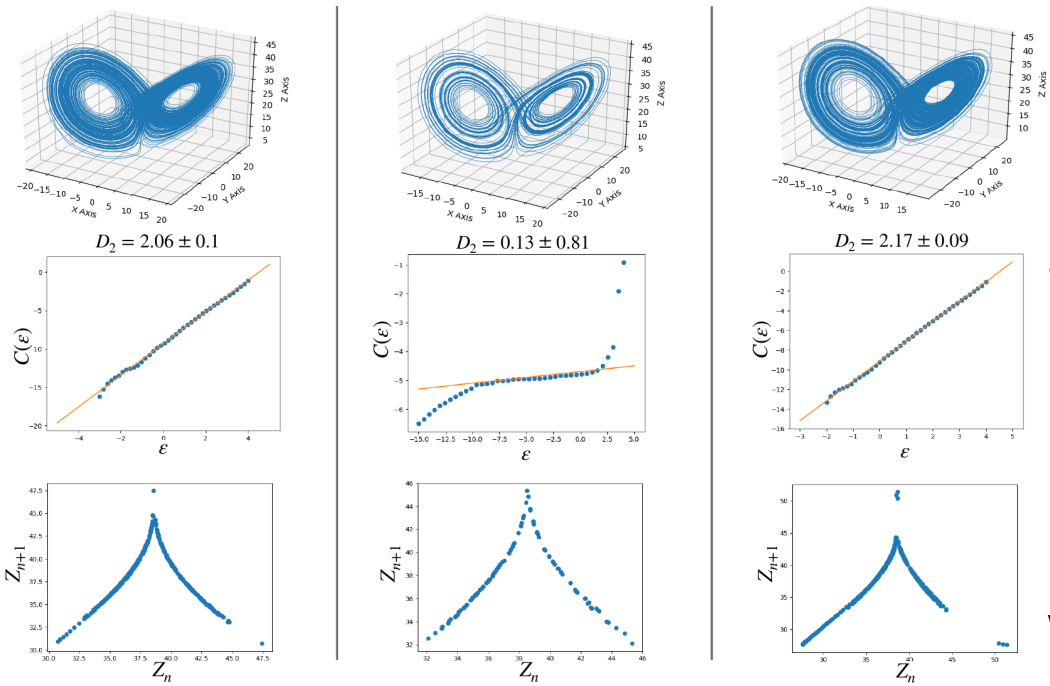
One training set, three different sampling strategies:

- 1 long trajectory on the attractor, $N = 27,000$ points.
- 9 short trajectories on the attractor, $N = 9 \times 3000$.
- 3*3 short trajectories ($N = 3 \times 3 \times 3000$) initialized around the unstable fixed points: $(0, 0, 0)$, $\left(-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho-1\right)$ and $\left(\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho-1\right)$. Perturbations along the three orthogonal directions of the Jacobian.

A numerical experiment

One training set, three different sampling strategies:

- 1 long trajectory on the attractor, $N = 27,000$ points.
- 9 short trajectories on the attractor, $N = 9 \times 3000$.
- 3*3 short trajectories ($N = 3 \times 3 \times 3000$) initialized around the unstable fixed points: $(0, 0, 0)$, $\left(-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho-1\right)$ and $\left(\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho-1\right)$. Perturbations along the three orthogonal directions of the Jacobian.



LSTM: $\Delta t = 0.01$, $\mathcal{J} = \sum_{n=1}^{N} \left\| \boldsymbol{u}_n - \widehat{\boldsymbol{u}}_n \right\|_2^2$, no noise.

Does not generalize

What happened? Any guess?...

Figure: Performance of different sampling strategies.

A numerical experiment

Several viewpoints:

- The predictability of the system is better close to the equilibrium points (low entropy, e.g. Kolmogorov-Sinai entropy).
- The loss function is well suited to learn close to these equilibrium points (non-chaotic dynamic). No need to further regularize the cost landscape [Scheme smooth (equilibrium points) vs rough (attractor)].
- Curriculum learning: first learn $\dot{x} = \alpha\, x$, then the rest.

What to learn?

General considerations on the loss

Crucial to ask the right question (objective) to hope getting a good answer (model), i.e., a model relevant for our application.

General considerations on the loss

Crucial to ask the right question (objective) to hope getting a good answer (model), i.e., a model relevant for our application.

Other problem: ease the learning.
The loss function induces the cost landscape and conditions how easy it is to get a good solution.

Non-chaotic system (near equilibrium points) $\longrightarrow$ the cost landscape $\mathcal{J}$ as a function of the model parameters and/or of the initial condition is smooth.

However, opposite for a chaotic system and the output (loss function) does not exhibit a smooth dependence with respect to inputs $\longrightarrow$ learning via gradient is difficult, high sensitivity.

Workarounds

- Restrict the solution space (parameters to be learned) by penalizing irrelevant subspaces. For instance, by penalizing the deviation from the Jacobian:

$$\mathcal{J} = \sum_{n=1}^{N} \left\| \boldsymbol{u}_n - \widehat{\boldsymbol{u}}_n \right\|_2^2 + \alpha \sum_{n=1}^{N} \left\| \nabla \boldsymbol{u}_n - \nabla \widehat{\boldsymbol{u}}_n \right\|_2^2.$$

Here: $L^2 \longrightarrow H^1$. This can be generalized to higher order losses.
$\longrightarrow$ promotes couplings (subspaces) between parameters and hence reduces the variance of the loss as a function of variations of the parameters $\Rightarrow$ smoother landscape.
Making it practical (no need for $\nabla \boldsymbol{u}$):

$$\mathcal{J} = \sum_{n=1}^{N} \left\| \boldsymbol{u}_n - \widehat{\boldsymbol{u}}_n \right\|_2^2 + \alpha \sum_{n=1}^{N} \left\| \nabla \widehat{\boldsymbol{u}}_n \right\|_2^2.$$

Should fit $\boldsymbol{u}$ without going crazy ($\left\| \nabla \widehat{\boldsymbol{u}}_n \right\|$).

Workarounds

- Restrict the solution space (parameters to be learned) by penalizing irrelevant subspaces. For instance, by penalizing the deviation from the Jacobian:

$$\mathcal{J} = \sum_{n=1}^{N} \left\| \boldsymbol{u}_n - \widehat{\boldsymbol{u}}_n \right\|_2^2 + \alpha \sum_{n=1}^{N} \left\| \nabla \boldsymbol{u}_n - \nabla \widehat{\boldsymbol{u}}_n \right\|_2^2.$$

  Here: $L^2 \longrightarrow H^1$. This can be generalized to higher order losses.
  $\longrightarrow$ promotes couplings (subspaces) between parameters and hence reduces the variance of the loss as a function of variations of the parameters $\Rightarrow$ smoother landscape.
  Making it practical (no need for $\nabla \boldsymbol{u}$):

$$\mathcal{J} = \sum_{n=1}^{N} \left\| \boldsymbol{u}_n - \widehat{\boldsymbol{u}}_n \right\|_2^2 + \alpha \sum_{n=1}^{N} \left\| \nabla \widehat{\boldsymbol{u}}_n \right\|_2^2.$$

  Should fit $\boldsymbol{u}$ without going crazy ($\left\| \nabla \widehat{\boldsymbol{u}}_n \right\|$).

- Penalizing "accelerations". The L^2 loss does not allow to penalize zero-mean accelerations.

Workarounds (2)

- Irregular sampling or more realistic loss (Ordinary Least Squares vs Total Least Squares [SCHEME]): DTW.

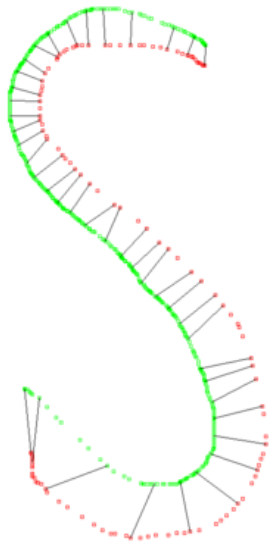


Figure: Sketch of the alignment of two irregularly sampled time-series via DTW.

Dynamic Time Warping (DTW) computes the best possible alignment between two discrete time series of respective length N and M by computing first the $N \times M$ pairwise distance matrix between these points to solve then a dynamic program (DP) using Bellman's recursion with a quadratic cost NM.
$\longrightarrow$ Robust to shifts, dilatation, etc. It is based on optimal transport.

Soft DTW makes this loss differentiable and efficient to optimize (gradient-based technique).

Physics-motivated losses

Physics-agnostic machine learned systems are fragile, for instance wrt noisy, small training datasets.

Example with stability
If one knows the system is stable $\Rightarrow$ to be enforced during learning.

Several definitions of stability:

- Stability in machine learning: robustness wrt perturbations in data or hyperparameters.
- Stability in Physics: given $\boldsymbol{u}_0$, the system remains within a ball of size r $\forall\, t > 0$ around $\boldsymbol{u}_0$.
- Stability: $\forall \epsilon > 0, \exists \delta > 0 \setminus |\boldsymbol{u}_0| < \delta \Rightarrow \left|\boldsymbol{f}^k(\boldsymbol{u}_0)\right| < \epsilon$, $k = 1, 2, \ldots$.

Physics-motivated losses — Stability

Difficult to verify numerically (costly) $\Rightarrow$ Stablity in the sense of Lyapunov.

$\boldsymbol{u}_0 = \mathbf{0}$ is stable in the sense of Lyapunov if it exists a continuous function $\mathcal{V} : \mathbb{R}^d \to \mathbb{R}$ such that:

$$\begin{aligned} \mathcal{V}(\mathbf{0}) = 0, \qquad & \mathcal{V}(\boldsymbol{u}) > 0, \qquad && \forall \boldsymbol{u} \neq \mathbf{0}, \\ \mathcal{V}(\boldsymbol{f}(\boldsymbol{u})) - \mathcal{V}(\boldsymbol{u}) \leq 0, \qquad & && \forall \boldsymbol{u} \neq \mathbf{0}, \quad |\boldsymbol{u}| < r. \end{aligned}$$

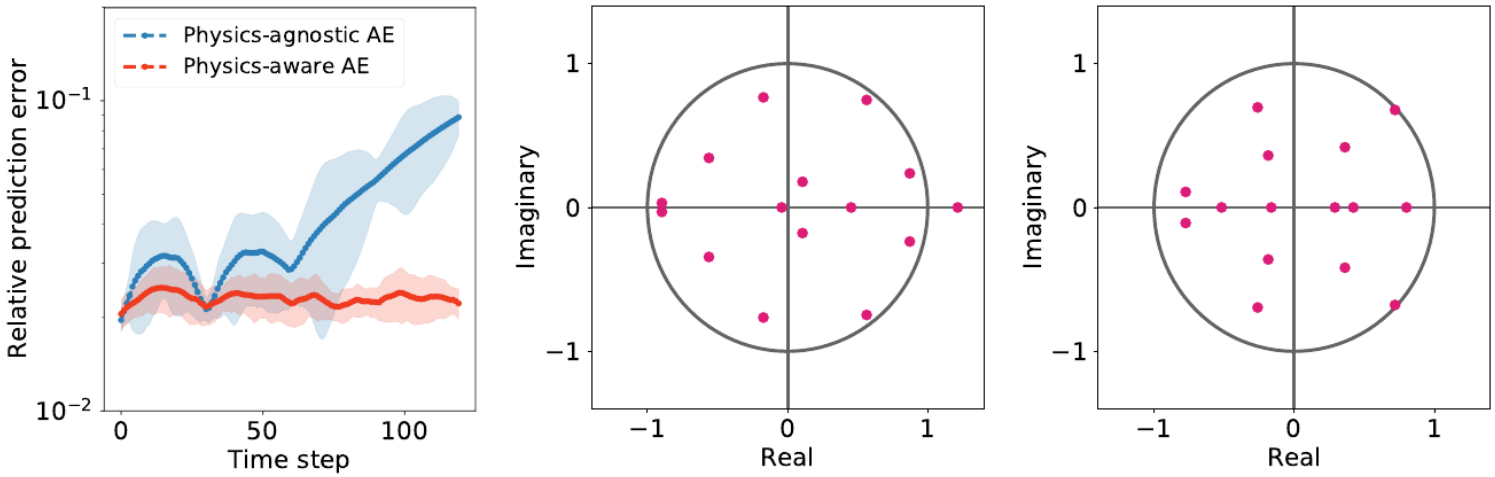
One can interpret $\mathcal{V}$ as a generalized energy function as it is often related to the total energy of the system. Dissipative system: $\mathcal{V}$ will decrease along trajectories.

Physics-motivated losses — Stability

If governing equations are linear, $\boldsymbol{f}(\boldsymbol{u})$ reduces to a matrix $\Rightarrow$ study of the eigenvalues. Stability if modulus always ≤ 1. Penalization of overflows in the loss.

One wants to impose these rules to the learned model: form of regularization.

Example with learning the (stable) flow around a circular cylinder via an auto-encoder, Erichson *et al.* (2019).



(A) With LR 1e−2 and WD 1e−6. (B) Physics-agnostic model in (a). (C) Physics-aware model in (a).

Figure: Performance and stability properties of Physics-agnostic and Physics-aware learning.

Illustration with conservation laws

Expertise on Physics allows us to know there exist many conservation laws.

Allows to further reduce the size of the search space. These conservation laws are counterparts of invariances with respect to some symmetry groups.

Illustration with conservation laws

Expertise on Physics allows us to know there exist many conservation laws.

Allows to further reduce the size of the search space. These conservation laws are counterparts of invariances with respect to some symmetry groups.

Noether theorem
To every differentiable symmetry generated by local actions there corresponds a conserved current.

Each "invariance" illustrates the fact that physical laws do not change when an experiment undergoes the associated transformation, hence there is no absolute reference to carry it out.

Example: invariance wrt spatial translation $\longrightarrow$ conservation of momentum $\boldsymbol{p} = m\,\boldsymbol{v}$.
Invariance in time $\longrightarrow$ conservation of total energy.

Imposing constraints in the form of Partial Differential Equations

One knows that the underlying physical system typically obeys certain equations, e.g. Navier-Stokes. Introduction in the loss definition.
Example with Burgers, Raissi *et al.* (2017):

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} &= \alpha\, \boldsymbol{u} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} + \beta\, \Delta \boldsymbol{u}, \\ \mathcal{J} &= \left\| \boldsymbol{u} - \widehat{\boldsymbol{u}} \right\|_2^2 + \lambda\, \left\| \boldsymbol{r} \right\|_2^2, \\ \boldsymbol{r} &= \frac{\partial \boldsymbol{u}}{\partial t} - \alpha\, \boldsymbol{u} \cdot \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} - \beta\, \Delta \boldsymbol{u}. \end{aligned}$$

Spatial gradient term is easy to get from NN.

This loss mimics $\left\| \boldsymbol{u} - \widehat{\boldsymbol{u}} \right\|_2^2 + \lambda\, \left\| \dot{\boldsymbol{u}} - \widehat{\dot{\boldsymbol{u}}} \right\|_2^2$.

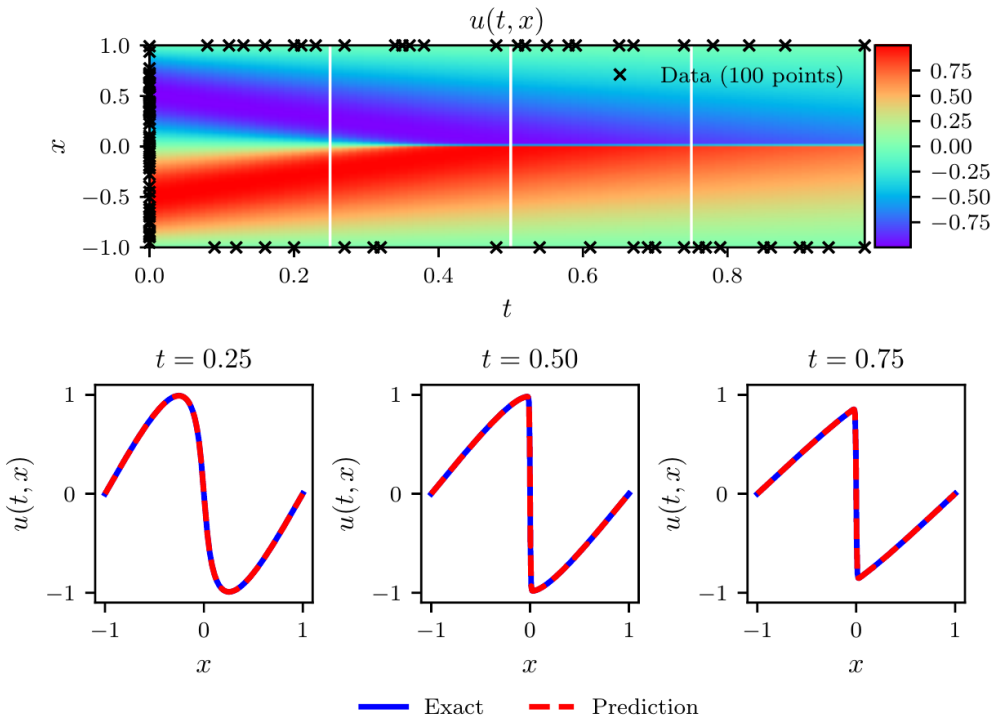


Figure: Reconstructed solution.

Deep Galerkin Method, Sirignano & Spiliopoulos (2018)

$$\begin{aligned} \frac{\partial \boldsymbol{u}(t,\boldsymbol{x})}{\partial t} + N(\boldsymbol{u}(t,\boldsymbol{x})) &= 0, && (t,\boldsymbol{x}) \in [0,T] \times \Omega, \\ \boldsymbol{u}(t=0,\boldsymbol{x}) &= \boldsymbol{u}_0(\boldsymbol{x}), \\ \boldsymbol{u}(t,\boldsymbol{x}) &= \boldsymbol{g}(t,\boldsymbol{x}), && \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

Deep neural network approximates $\boldsymbol{u}(t,\boldsymbol{x})$ from just $(t,\boldsymbol{x})$. No need to know the true $\boldsymbol{u}$.

$$\mathcal{J} = \left\| \frac{\partial \boldsymbol{u}(t,\boldsymbol{x})}{\partial t} + N(\boldsymbol{u}(t,\boldsymbol{x})) \right\|^2_{[0,T]\times\Omega} + \|\boldsymbol{u}(0,\boldsymbol{x}) - \boldsymbol{u}_0(\boldsymbol{x})\|^2_{\Omega} + \|\boldsymbol{u}(t,\boldsymbol{x}) - \boldsymbol{g}(t,\boldsymbol{x})\|^2_{[0,T]\times\partial\Omega}$$

No need for a mesh. Training on randomly sampled points in space and time.

Operators such as $\frac{\partial}{\partial t}$ or N are easy to implement on top of the network providing $\boldsymbol{u}$.
Norms are approximated with random points $\longrightarrow$ Stochastic gradient descent.

Predicting with the right variable, Ayed *et al.* (2019).

Aim: predict I_{k+1} from previous snapshots $\{I_k\}_k$. One knows that I obeys

$$\frac{\partial I(t,\boldsymbol{x})}{\partial t} + \boldsymbol{w}(t,\boldsymbol{x}) \cdot \boldsymbol{\nabla} I(t,\boldsymbol{x}) = D\,\Delta I.$$

Initial conditions not known, nor the boundary conditions, nor the diffusion parameter D. Instead, the displacement field $\boldsymbol{w}$ is learned from previous snapshots with the loss:

$$\mathcal{J} = \sum_{\boldsymbol{x}\in\Omega} \left| I_{t+1}(\boldsymbol{x}) - \widehat{I}_{t+1}(\boldsymbol{x}) \right| + \lambda_1 \left(\boldsymbol{\nabla}\cdot\widehat{\boldsymbol{w}}_t(\boldsymbol{x})\right)^2 + \lambda_2 \left\|\widehat{\boldsymbol{w}}_t(\boldsymbol{x})\right\|_2^2 + \lambda_3 \left\|\boldsymbol{\nabla}\widehat{\boldsymbol{w}}_t(\boldsymbol{x})\right\|_2^2.$$

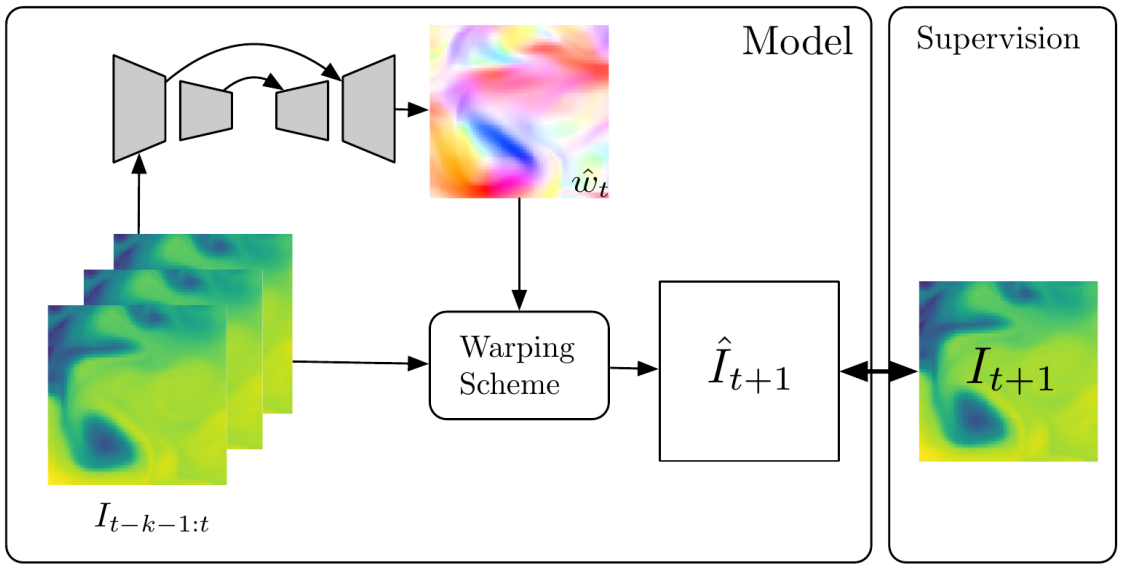


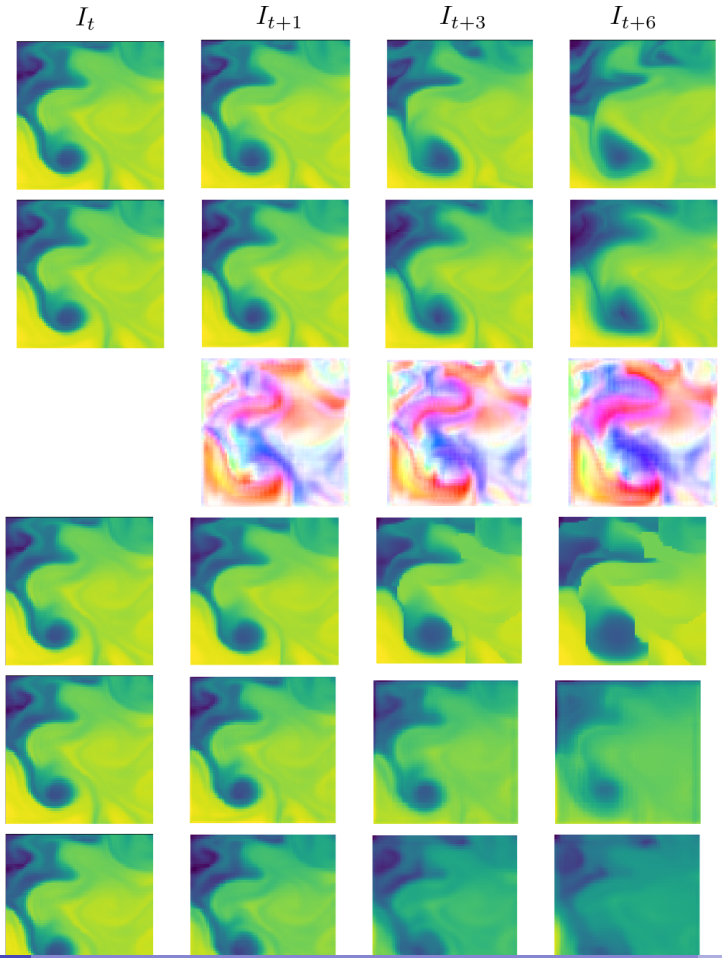
Figure: Sketch of the strategy to predict the next image $\hat{I}_{t+1}$.

Predicting with the right variable, Ayed *et al.* (2019).

Warping scheme:

$$\widehat{I}_{t+1}(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\Omega} k\left(\boldsymbol{x} - \widehat{\boldsymbol{w}}_t, \boldsymbol{y}\right)\ I_t\left(\boldsymbol{y}\right),$$

$$k\left(\boldsymbol{x} - \widehat{\boldsymbol{w}}_t, \boldsymbol{y}\right) = \frac{1}{4\,\pi\,\widehat{D}\,\Delta t}\ \exp\left(-\frac{1}{4\,\widehat{D}\,\Delta t}\ \left\|\boldsymbol{x} - \widehat{\boldsymbol{w}} - \boldsymbol{y}\right\|^2\right).$$



How to learn?

Structure / functional form to use

Determinism of the reconstruction dynamics

Partial Observability

The way a system is observed is often critical and conditions the architecture and the resulting performance.

Determining d: Trajectories in the phase space cannot cross. [Sketch]

Reconstruct a flow in dimension $d_r = 2$ then $d_r = 3$, etc., until the trajectory does not cross itself anymore.
The smallest dimension this happens is the embedding dimension d_p.

Embeddings for dynamical systems

We seek a model for a dynamical system $\Rightarrow$ requires two ingredients:

- a coordinate system,
- a representation for the system evolution.

The first ingredient is obtained by an **embedding**: a (diffeomorphic) function g that maps the observed data y_t to a set of coordinates $\widetilde{x}$ according to $g(\{y_t\}_t) = (\widetilde{x}_0, \widetilde{x}_1, \ldots)$. It preserves the topology and the invariants (Lyapunov exponents, ...).

We assume the system lies on a low dimensional attractor $x(t) \in \mathbb{A} \subset \mathbb{R}^d$, a smooth submanifold of $\mathbb{R}^d$ and has a dimension $\dim(\mathbb{A}) < d$.

Then, **Whitney embedding theorem** says that if the number of coordinates $d_c > 2\dim(\mathbb{A})$ then the dynamics of the state can be entirely captured by the new coordinate system.

Made practical by the **Takens embedding theorem** by asserting that one can take this number of delays.

Embeddings for dynamical systems

Three common mappings:

1. **Delays**: $\widetilde{x}_i = y(t - i\Delta t)$ for $i = 0, 1, \ldots$ and $\Delta t \in \mathbb{R}_+$. $\widetilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{y}(t_i) \\ \boldsymbol{y}(t_{i-1}) \\ \cdots \end{bmatrix}$.
2. **Derivatives**: $\widetilde{x}_i = y^{(i)}$ for $i = 0, 1, \ldots$ $\widetilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{y} \\ \dot{\boldsymbol{y}} \\ \ddot{\boldsymbol{y}} \\ \cdots \end{bmatrix}$.
3. **Principal components**: an SVD is used to first transform the space followed by a use of delays or derivatives. $H = \begin{bmatrix} \boldsymbol{y}(t_i) \\ \boldsymbol{y}(t_{i-1}) \\ \cdots \end{bmatrix} = U \Sigma V^*$, $\widetilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \cdots \end{bmatrix}$

Embeddings for dynamical systems

In practice, how to find the embedding dimension ? False neighbor algorithm to check if topology is preserved.

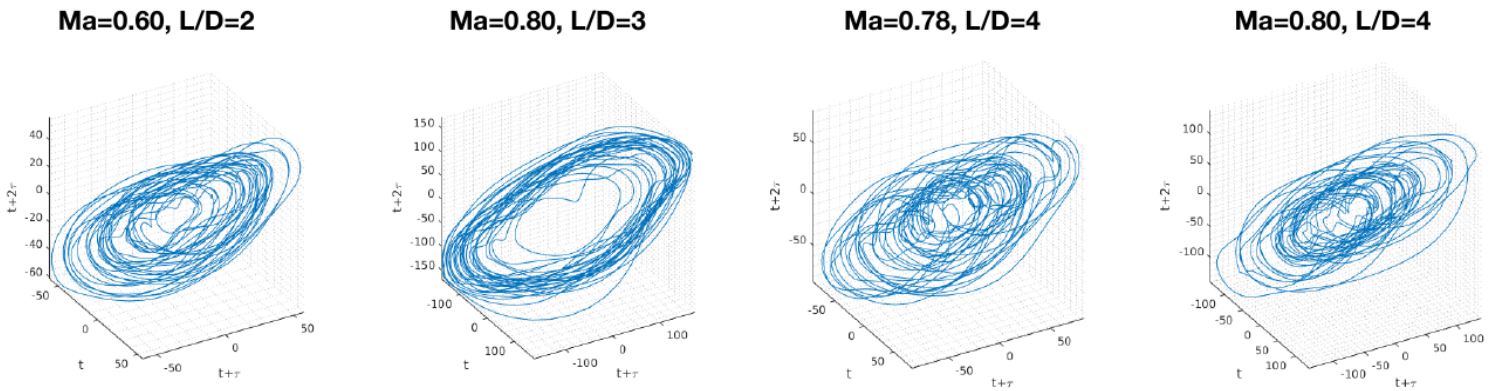


Figure: Different reconstructed phase spaces. They are topologically equivalent to the Lorenz canonical representation.

Observability

Observability

Observing the system through the measurement operator h:

- how much information can be extracted from it?
- what can be inferred on the underlying model?

For instance, how much of the state x and of its dynamics can be inferred from measurements $y(t)$?
$\Rightarrow$ observability issue (was extensively studied, in particular by the control community).

When both the model f and the observation operator h are **linear**, full observability of $x \in \mathbb{X} \subset \mathbb{R}^d$ from $y(t) \in \mathbb{R}^{n_y}$ is guaranteed whenever the Kalman rank condition is satisfied, i.e., the observability matrix $\mathcal{O}_h$ is full rank $\forall\, x \in \mathbb{X}$:

$$\mathcal{O}_h(x) := \begin{bmatrix} h \\ hf \\ \vdots \\ hf^{d-1} \end{bmatrix}.$$

Nonlinear observability

Extends to the **nonlinear framework**, for instance considering Lie derivatives of h along the vector field f:

$$\mathcal{L}_f\, h(x) := \frac{\partial h}{\partial x}\, f(x) \qquad \text{and} \qquad \mathcal{L}_f^n\, h(x) := \mathcal{L}_f \left[\mathcal{L}_f^{n-1}\, h(x)\right], \quad \forall\, n \in \mathbb{N}, \qquad \mathcal{L}_f^0\, h(x) := h$$

The system governed by f is fully observable via h iff $\mathcal{O}_h$ is full rank $\forall\, x \in \mathbb{X}$, with $\mathcal{O}_h$ writing as:

$$y(t) = h(x(t)), \qquad \dot{y}(t) = \frac{\partial h}{\partial x}\, \dot{x} \equiv \mathcal{L}_f\, h(x).$$

then

$$\mathcal{O}_h(x) = \begin{bmatrix} \frac{\partial \mathcal{L}_f^0\, h(x)}{\partial x} \\ \frac{\partial \mathcal{L}_f^1\, h(x)}{\partial x} \\ \vdots \\ \frac{\partial \mathcal{L}_f^{d-1}\, h(x)}{\partial x} \end{bmatrix} = \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \vdots \\ y^{(d-1)}(t) \end{bmatrix}.$$

Degree of observability can be quantified based on the spectrum of the observability Gramian $\mathcal{O}_h^T \mathcal{O}_h$ via the observability index $\delta_{\mathcal{O}_h}$:

$$\delta_{\mathcal{O}_h}(x) := \frac{\lambda_{\min}(\mathcal{O}_h^T \mathcal{O}_h)}{\lambda_{\max}(\mathcal{O}_h^T \mathcal{O}_h)} \quad \in [0, 1].$$

Choice of the observables

The above criteria are useful to quantify the relevance of the observables $y(t)$ to inform upon the state $\boldsymbol{x}$.

It can also serve to improve observability by modifying the observables so that $\delta_{\mathcal{O}_h}$ increases. $\longrightarrow$ pivotal for sensor placement.

While similarity transformations of coordinates do not affect the rank of the observability matrix, they can improve the observability index and provide a more robust information on the system.

Is memory necessary ?
State vector (LSTM useless) vs Partial observation (memory-capable network, LSTM, reservoir).
Alternative: embedding (delayed, differential, ODE) $\longrightarrow$ augmented input state.

A continuous viewpoint: Neural ODE

Neural ODE (ODENet), Chen *et al.* (2018)

One can interpret Resnets like Ordinary Differential Equations discretized with an Euler scheme:

$$\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \boldsymbol{f}(\boldsymbol{u}_t, t; \boldsymbol{\theta}).$$

Neural ODE (ODENet), Chen *et al.* (2018)

One can interpret Resnets like Ordinary Differential Equations discretized with an Euler scheme:

$$\boldsymbol{u}_{t+\Delta t} = \boldsymbol{u}_t + \boldsymbol{f}(\boldsymbol{u}_t, t; \boldsymbol{\theta}).$$

In the limit of more layers and smaller Δt:

$$\frac{d\,\boldsymbol{u}(t)}{d\,t} = \boldsymbol{f}(\boldsymbol{u}_t, t; \boldsymbol{\theta}).$$

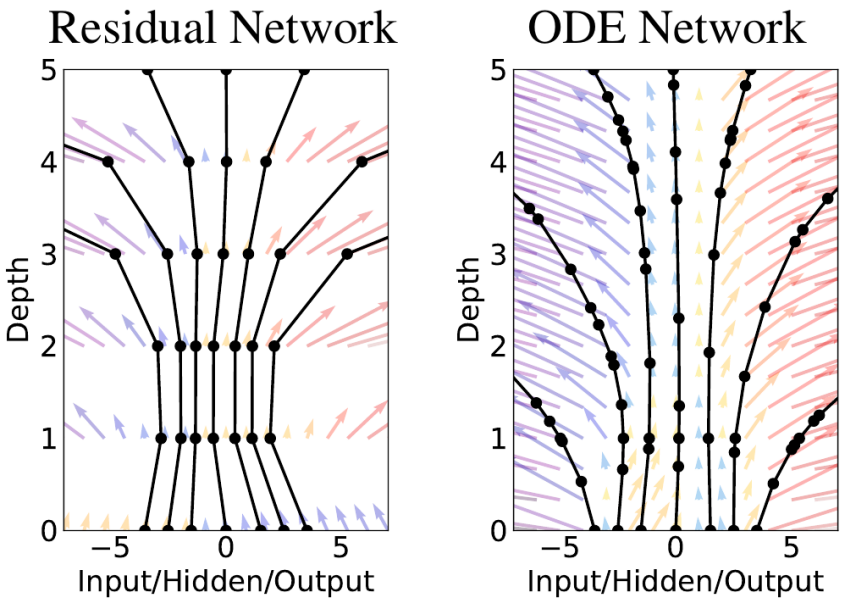


Figure: In the limit of an infinite number of layers, a ResNet (left) can be interpreted as an ODE problem (right).

One can use an ODE solver to compute $\boldsymbol{u}(t)$ at any real continuous time $t > 0$.

More flexible since the dynamics might also depend on t, for instance using a neural network with two inputs: t and $\boldsymbol{u}(t)$.

Neural ODE (ODENet), Chen *et al.* (2018)

Training via the adjoint Loss is defined at a final time $t = T$: $\mathcal{J}(\boldsymbol{u}(T))$
How to train / compute $\nabla_{\boldsymbol{\theta}} \mathcal{J}$?

$$
\begin{aligned}
\mathscr{L} &:= J(\boldsymbol{u}_T) + \ll \lambda, \dot{\boldsymbol{u}}(t) + \boldsymbol{f}(\boldsymbol{u};\boldsymbol{\theta}) \gg, \\
&= J(\boldsymbol{u}_T) + \int_t \boldsymbol{\lambda}(t)\ (\dot{\boldsymbol{u}}(t) + \boldsymbol{f}(\boldsymbol{u};\boldsymbol{\theta}))\, dt, \\
\left\langle \frac{\partial \mathscr{L}}{\partial \boldsymbol{u}}\, \delta\boldsymbol{u} \right\rangle = 0 \,\forall\, \delta\boldsymbol{u} &= \nabla_{\boldsymbol{u}} J\, \delta\boldsymbol{u} + \int_t \delta\boldsymbol{u} \left(-\dot{\boldsymbol{\lambda}} + \boldsymbol{\lambda} \nabla_{\boldsymbol{u}} \boldsymbol{f} \right) dt + [\boldsymbol{u}\,\boldsymbol{\lambda}]_0^T\,, \\
&= -\dot{\boldsymbol{\lambda}} + \boldsymbol{\lambda} \nabla_{\boldsymbol{u}} \boldsymbol{f}, \qquad \text{with } \boldsymbol{\lambda}_T = -\left. \nabla_{\boldsymbol{u}} J \right|_T\,, \\
\left\langle \frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}}\, \delta\boldsymbol{\lambda} \right\rangle = 0 \,\forall\, \delta\boldsymbol{\lambda} &= \dot{\boldsymbol{u}}(t) + \boldsymbol{f}(\boldsymbol{u};\boldsymbol{\theta}), \\
\left\langle \frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}}\, \delta\boldsymbol{\theta} \right\rangle = 0 \,\forall\, \delta\boldsymbol{\theta} &= \frac{\partial J}{\partial \boldsymbol{\theta}}\, \delta\boldsymbol{\theta} + \int_t \boldsymbol{\lambda}\, \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\, \delta\boldsymbol{\theta}\, dt, \qquad \Rightarrow \frac{\partial J}{\partial \boldsymbol{\theta}} = -\int_t \boldsymbol{\lambda}\, \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\, dt.
\end{aligned}
$$

Both $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ are obtained via an ODE solver from 0 to T and from T to 0 respectively.

All this can be solved with a single call to an ODE solver considering the augmented state: $\boldsymbol{z} := \left[\boldsymbol{u}, \boldsymbol{\lambda}, \frac{\partial J}{\partial \boldsymbol{\theta}} \right]$.

Neural ODE (ODENet) — Example

Model and complement an irregularly sampled time series $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$

- Difficult to do with recurrent networks.
- The dynamics learned / to be learned do not act directly on $\boldsymbol{x}_t$ (observations) but on the (latent) internal state $\boldsymbol{z}_t$ (state vector) [in order to be able to deal with partial or noisy information]

Neural ODE (ODENet) — Example

Model and complement an irregularly sampled time series $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$

- Difficult to do with recurrent networks.
- The dynamics learned / to be learned do not act directly on $\boldsymbol{x}_t$ (observations) but on the (latent) internal state $\boldsymbol{z}_t$ (state vector) [in order to be able to deal with partial or noisy information]

1. Transform the $\boldsymbol{x}_t$ sequence into a guessed initial (probabilistic) internal state $q(\boldsymbol{z}_0 \mid \boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ using an RNN (going reverse in time, from $\boldsymbol{x}_N$ to $\boldsymbol{x}_0$; the RNN has to be learned as well)
2. Apply the dynamics $\boldsymbol{f}_\theta$ previously learned to $\boldsymbol{z}_0$ using an ODE solver and produce guessed states $\boldsymbol{z}_{t>0}$
3. From $\boldsymbol{z}_{t>N}$, deduce missing observations $\boldsymbol{x}_{t>N}$.

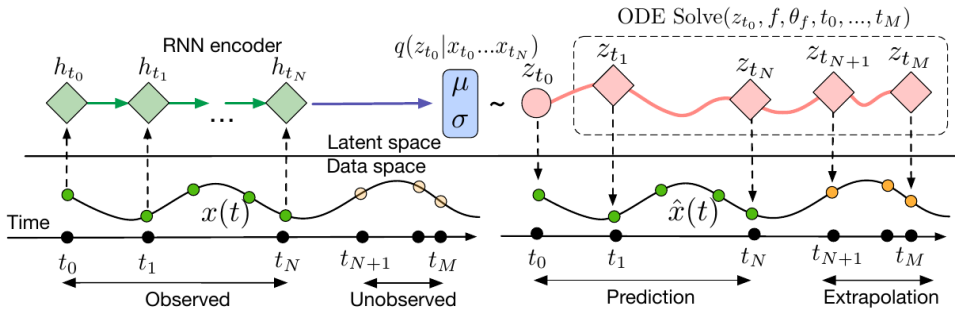
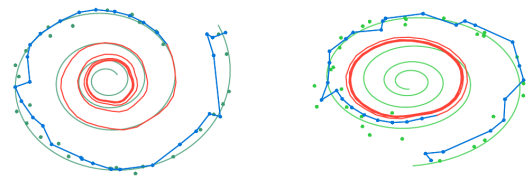
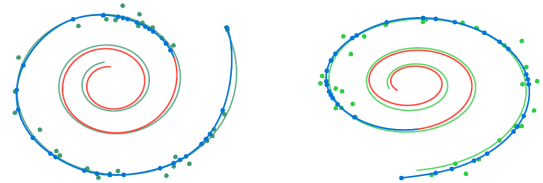


Figure: Sketch of the prediction strategy for future observations $\boldsymbol{x}_{t>N}$.



(a) Recurrent Neural Network

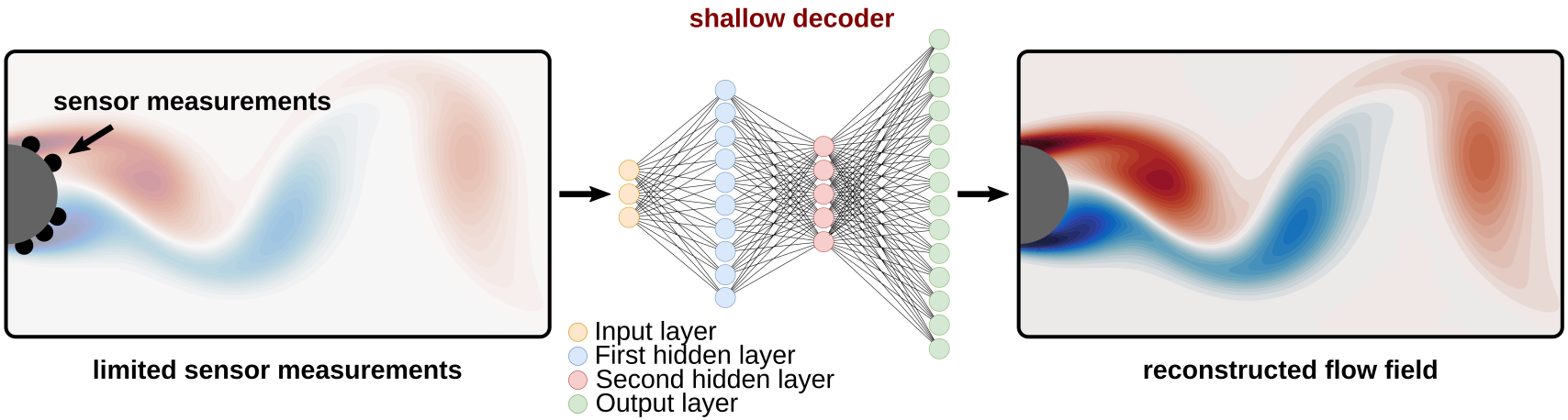


(b) Latent Neural Ordinary Differential Equation

Field reconstruction

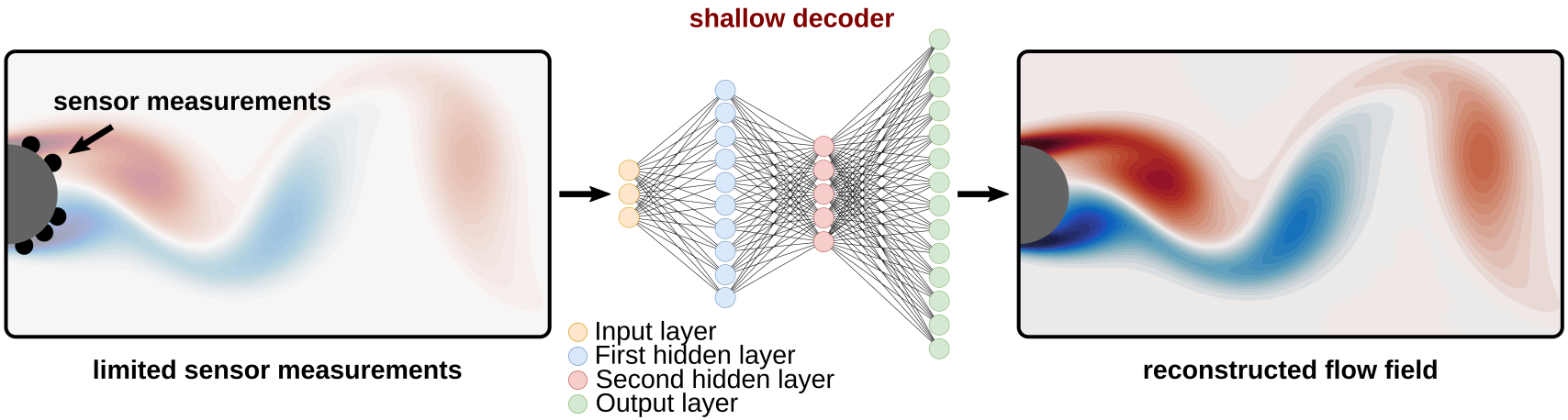
A shallow decoder for flow reconstruction, Erichson *et al.* (2020)

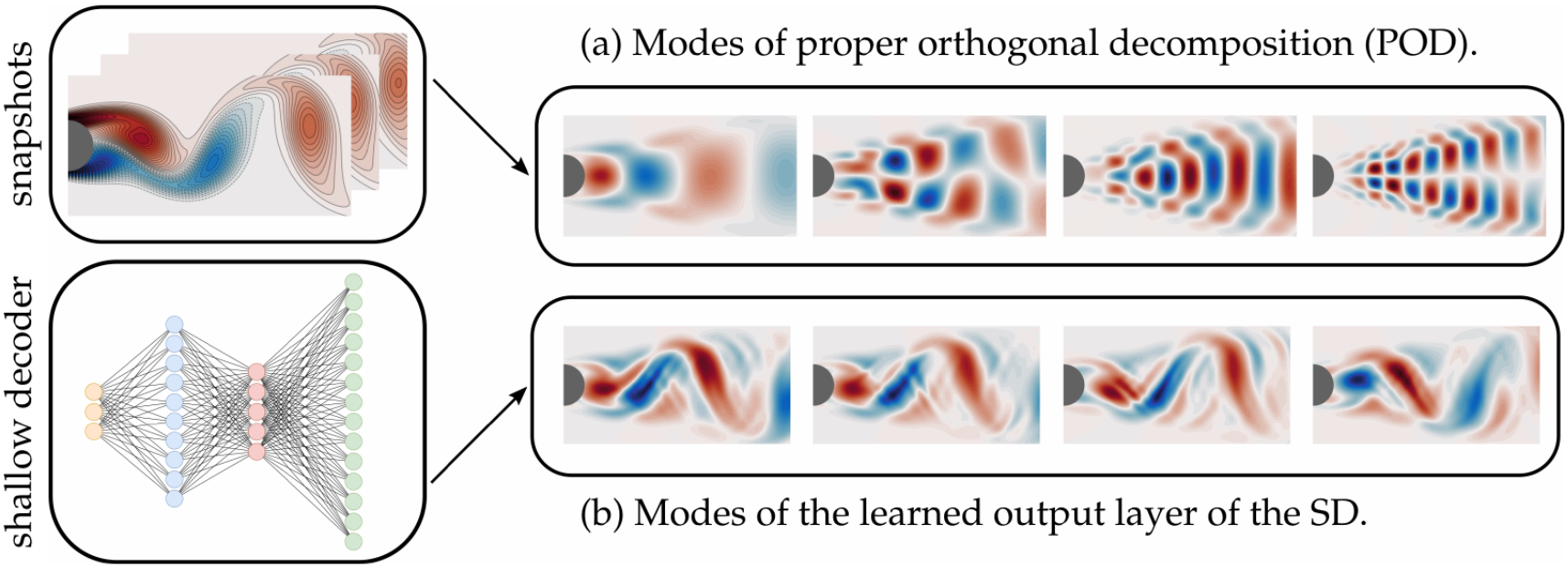
From sensors $\boldsymbol{s} \in \mathbb{R}^5 \longrightarrow$ to field estimation $\widehat{\boldsymbol{x}} \in \mathbb{R}^{78406}$



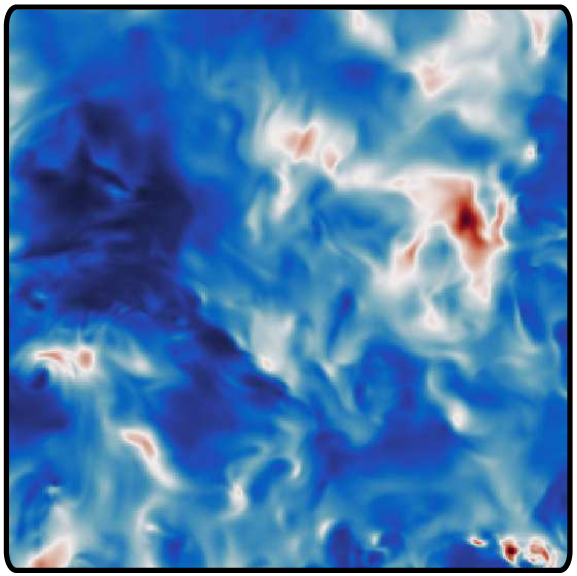
A shallow decoder for flow reconstruction, Erichson *et al.* (2020)

From sensors $\boldsymbol{s} \in \mathbb{R}^5 \longrightarrow$ to field estimation $\widehat{\boldsymbol{x}} \in \mathbb{R}^{78406}$

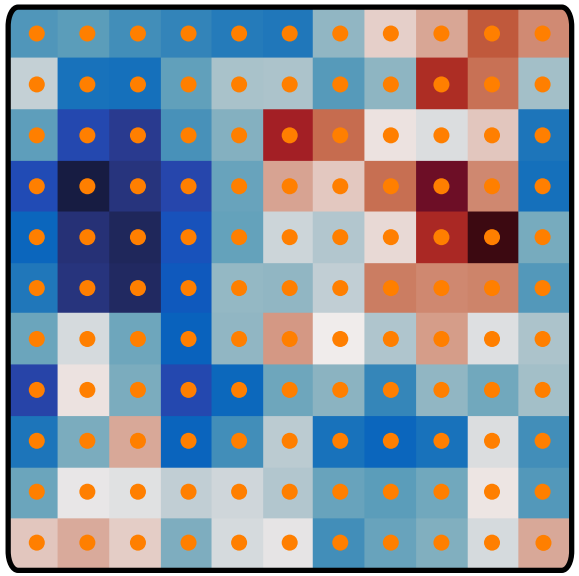




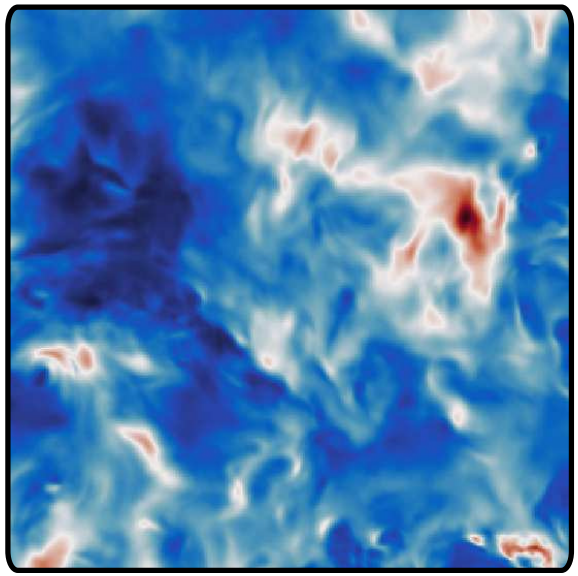
Super-resolution — Rise...



Snapshot

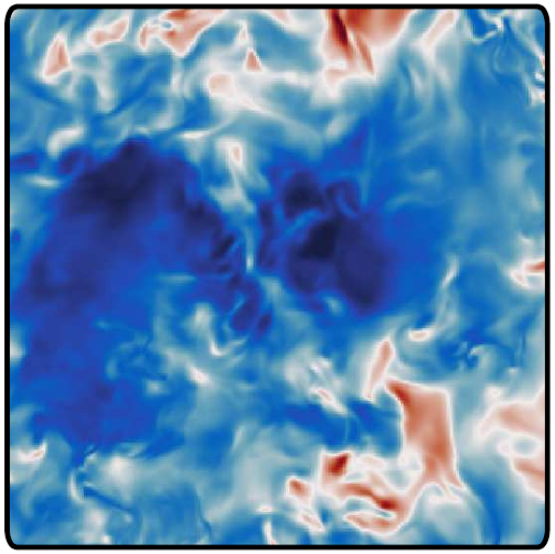


Low resolution

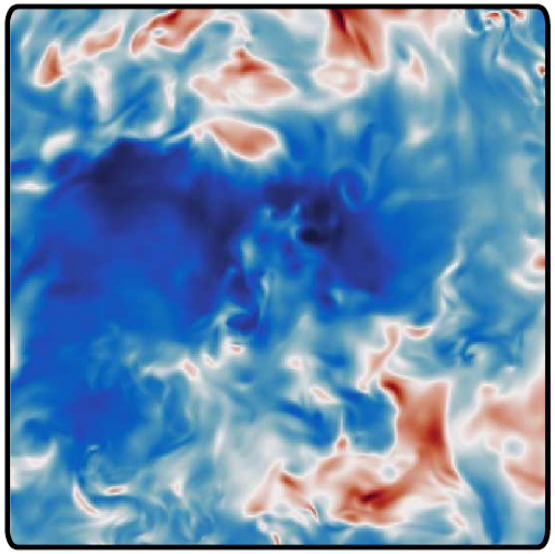


Shallow decoder

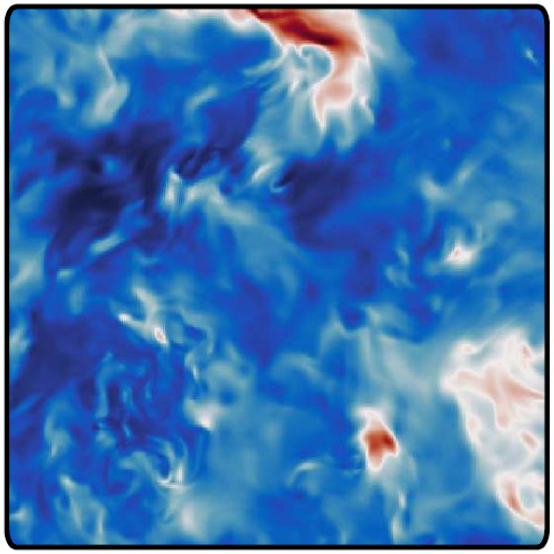
. . . and fall — On the requirement of satisfying sufficient statistics



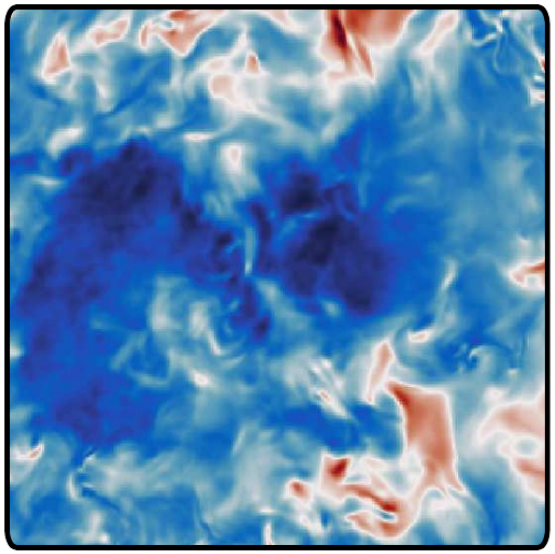
(a) Test snapshot $t = 1$



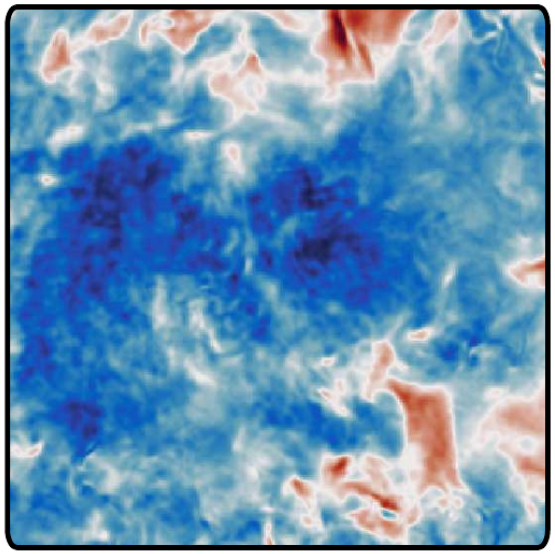
(c) Test snapshot $t = 20$



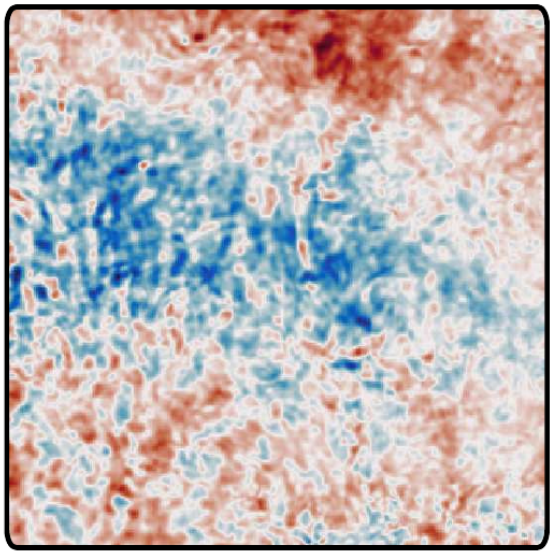
(e) Test snapshot $t = 50$



(b) Reconstruction



(d) Reconstruction



(f) Reconstruction

Weak solution of PDEs — Zang *et al.* (2020)

High-dimensional boundary value problem

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \left(a \nabla u \right) + b \nabla u + cu + d = 0, & \qquad \text{in } \Omega \times [0, T], \\ u \left(\boldsymbol{x}, t \right) = g \left(\boldsymbol{x}, t \right), & \qquad \text{on } \partial\Omega \times [0, T], \\ u \left(\boldsymbol{x}, 0 \right) = h \left(\boldsymbol{x} \right), & \qquad \text{on } \Omega. \end{aligned}$$

Weak solution of PDEs — Zang *et al.* (2020)

High-dimensional boundary value problem

$$\begin{aligned} \frac{\partial u}{\partial t} - \nabla \left(a \nabla u \right) + b \nabla u + cu + d = 0, & \qquad \text{in } \Omega \times [0, T], \\ u \left(\boldsymbol{x}, t \right) = g \left(\boldsymbol{x}, t \right), & \qquad \text{on } \partial\Omega \times [0, T], \\ u \left(\boldsymbol{x}, 0 \right) = h \left(\boldsymbol{x} \right), & \qquad \text{on } \Omega. \end{aligned}$$

Weak formulation

For time-independent problems, *for instance*

$$\left\langle A \left[u \right], \varphi \right\rangle := \int_{\Omega} a \nabla u \nabla \varphi + b \nabla u \varphi + cu\varphi + d\varphi \, \mathrm{d}\boldsymbol{x} = 0, \qquad \forall \, \varphi \in H_0^1.$$

Approximation basis for the solution is given by a neural network $\longrightarrow$
$u \left(\boldsymbol{x}, t \right) \approx u_\theta \left(\boldsymbol{x}, t \right) \in H^1 \left(\Omega; \mathbb{R} \right)$.
Test functions $\{ \varphi \} \in H_0^1 \left(\Omega; \mathbb{R} \right)$
The choice of the projection $\langle \cdot \rangle$ is *yours*.

Induced operator norm minimization

Let $A[u] : H_0^1(\Omega) \to \mathbb{R}$ such that $A[u](\varphi) := \langle A[u], \varphi \rangle$.
Its operator norm is

$$\|A[u]\|_{\text{op}} := \max \left\{ \langle A[u], \varphi \rangle / \|\varphi\|_2 \mid \varphi \in H_0^1,\ \varphi \neq 0 \right\}.$$

u is a weak solution of the PDE iff $\|A[u]\|_{\text{op}} = 0$.

Induced operator norm minimization

Let $A[u] : H_0^1(\Omega) \to \mathbb{R}$ such that $A[u](\varphi) := \langle A[u], \varphi \rangle$.
Its operator norm is

$$\|A[u]\|_{\text{op}} := \max \left\{ \langle A[u], \varphi \rangle / \|\varphi\|_2 \ \mid \varphi \in H_0^1,\ \varphi \neq 0 \right\}.$$

u is a weak solution of the PDE iff $\|A[u]\|_{\text{op}} = 0$.
Hence, a formulation for an approximation of the weak solution u_θ could be

$$\min_{u_\theta \in H^1} \|A[u_\theta]\|_{\text{op}}^2 \iff \min_{u_\theta \in H^1} \max_{\varphi \in H_0^1} |\langle A[u_\theta], \varphi \rangle|^2 / \|\varphi\|_2^2.$$

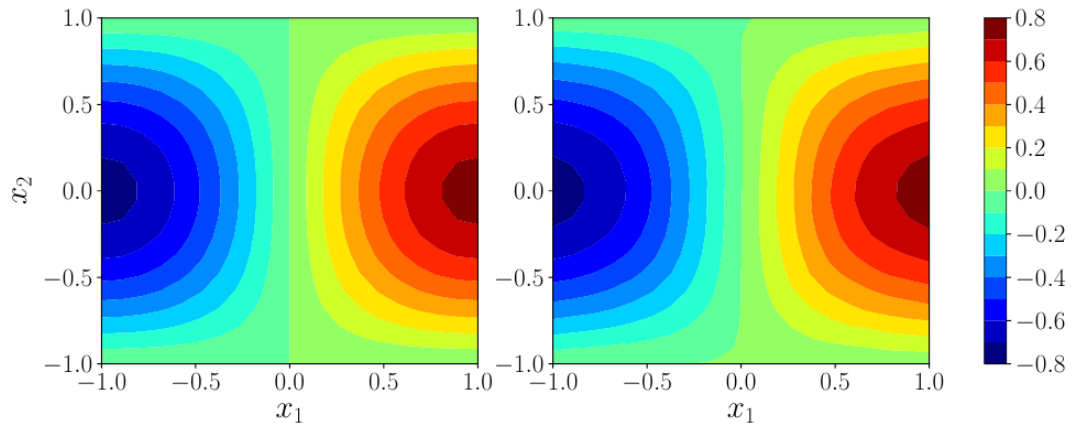
- A deep neural network parameterized by $\theta \longrightarrow u_\theta$,
- An adversarial network parameterized by η tending to maximize $\langle A[u_\theta], \varphi_\eta \rangle \longrightarrow \varphi_\eta$,

Example with a parabolic equation

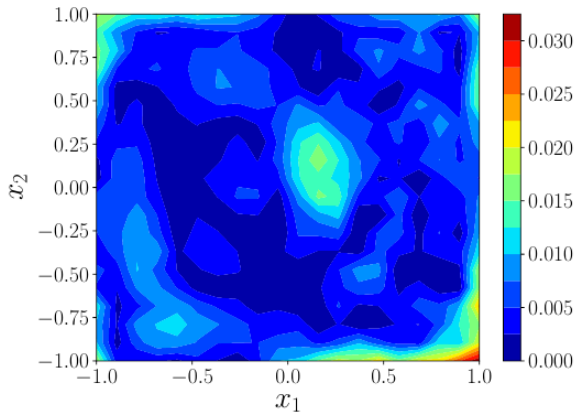
Nonlinear reaction-diffusion equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \Delta u - u^2 &= 0, && \text{in } \Omega \times [0, T], \\ u(\boldsymbol{x}, t) &= g(\boldsymbol{x}, t), && \text{on } \partial\Omega \times [0, T], \\ u(\boldsymbol{x}, 0) &= h(\boldsymbol{x}), && \text{on } \Omega. \end{aligned}$$

Crank-Nicolson time scheme. 6 hidden layers, 40 neurons per layer. Trained on collocation points in Ω.



(a) True $u^*(x, T)$ (left) vs estimated $u_\theta(x, T)$ (right)



(b) $|u_\theta(x, T) - u^*(x, T)|$

AYED IBRAHIM, DE BÉZENAC EMMANUEL, PAJOT ARTHUR, BRAJARD JULIEN & GALLINARI PATRICK, Learning Dynamical Systems from Partial Observations, *arXiv e-prints*, p. arXiv:1902.11136, 2019.

CHEN RICKY T. Q., RUBANOVA YULIA, BETTENCOURT JESSE & DUVENAUD DAVID, Neural Ordinary Differential Equations, *arXiv e-prints*, p. arXiv:1806.07366, 2018.

ERICHSON N.B., MUEHLEBACH M. & MAHONEY M.W., 2019 Physics-informed autoencoders for Lyapunov-stable fluid flow prediction. Preprint: arXiv:1905.10866.

ERICHSON N. BENJAMIN, MATHELIN LIONEL, YAO ZHEWEI, BRUNTON STEVEN, MAHONEY MICHAEL & KUTZ J. NATHAN, Shallow neural networks for fluid flow reconstruction with limited sensors, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **476** (2238), p. 20200097, 2020.

RAISSI M., PERDIKARIS P. & KARNIADAKIS G.EM, 2017 Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. Preprint: arXiv:1711.10561.

SIRIGNANO JUSTIN & SPILIOPOULOS KONSTANTINOS, DGM: A deep learning algorithm for solving partial differential equations, *Journal of Computational Physics*, **375**, p. 1339–1364, 2018.

ZANG YAOHUA, BAO GANG, YE XIAOJING & ZHOU HAOMIN, Weak adversarial networks for high-dimensional partial differential equations, *Journal of Computational Physics*, **411**, p. 109409, 2020.