Using Physics-informed Neural Networks for Inverse Problems

João M Pereira

Instititute de Matemática Pura e Aplicada

13 de Maio de 2024

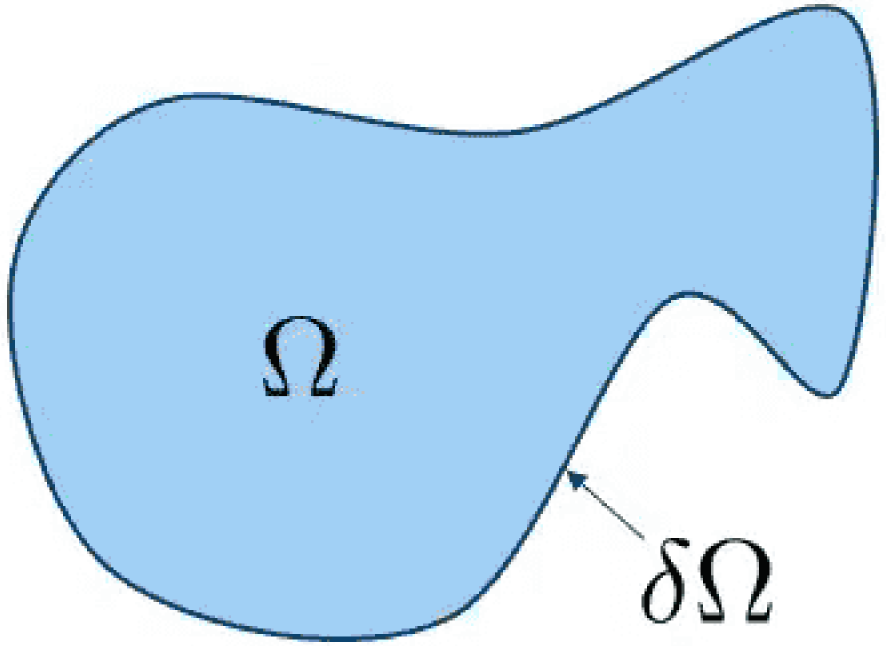
Seminário de Pós-graduação

Laboratório Nacional de Computação Científica

Physics-Informed Neural Networks (PINNs)

- Example: Solve a boundary value problem

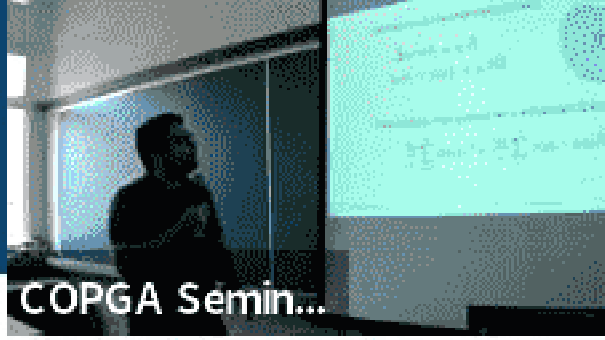
$$\Delta u = 0,\ x \in \Omega$$
$$u(x) = g(x),\ x \in \delta\Omega$$



- PINN Solution: Train a neural net $\hat{u}$ with domain Ω and loss

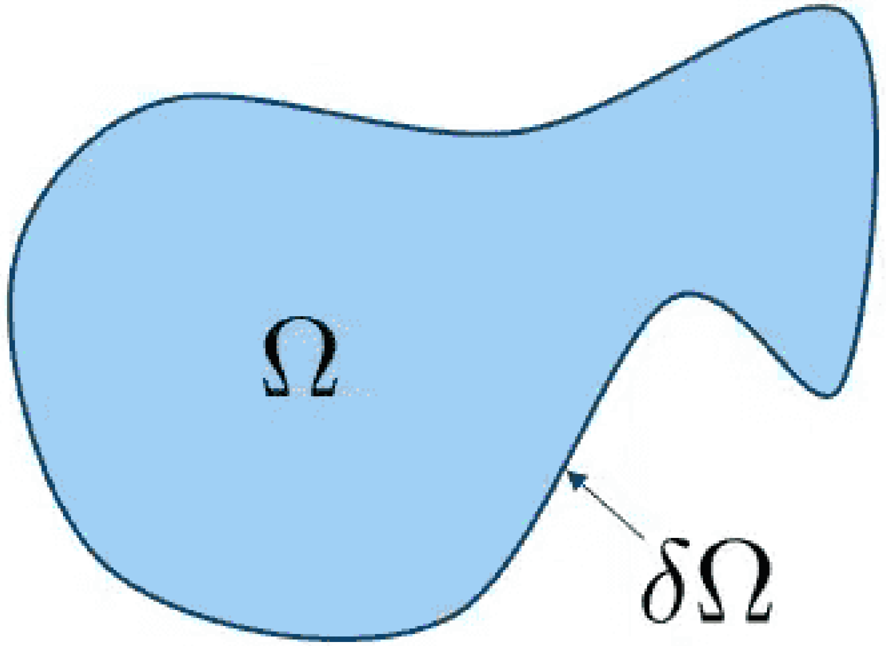
$$\int_{\Omega} \Delta\hat{u}(x)^2\, dx + \int_{\delta\Omega} (\hat{u}(x) - g(x))^2\, dx$$

Physics-Informed Neural Networks (PINNs)



- Example: Solve a boundary value problem

$$\Delta u = 0,\ x \in \Omega$$
$$u(x) = g(x),\ x \in \delta\Omega$$



- PINN Solution: Train a neural net $\hat{u}$ with domain Ω and loss

$$\frac{\lambda_\Omega}{p}\sum_{i=1}^{p}\Delta\hat{u}(x_i)^2 + \frac{\lambda_{\delta\Omega}}{s}\sum_{i=1}^{s}(\hat{u}(y_i) - g(y_i))^2$$

$x_1, \ldots, x_p \in \Omega$ $\quad$ $y_1, \ldots, y_s \in \delta\Omega$

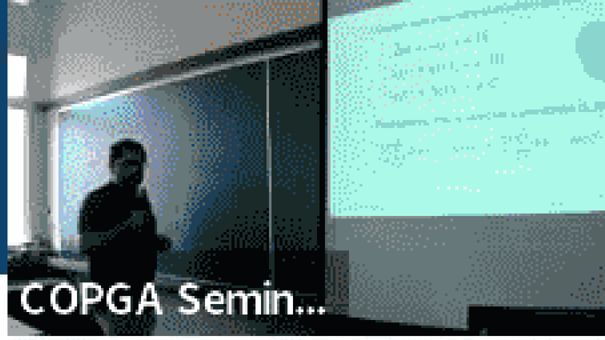
Why it works: Universal Approximation Theo

- Neural networks approximate functions (and derivatives) arbitrarily well

- These are approximately solutions to the PDE

- Evaluate derivatives efficiently with *autograd* implementation

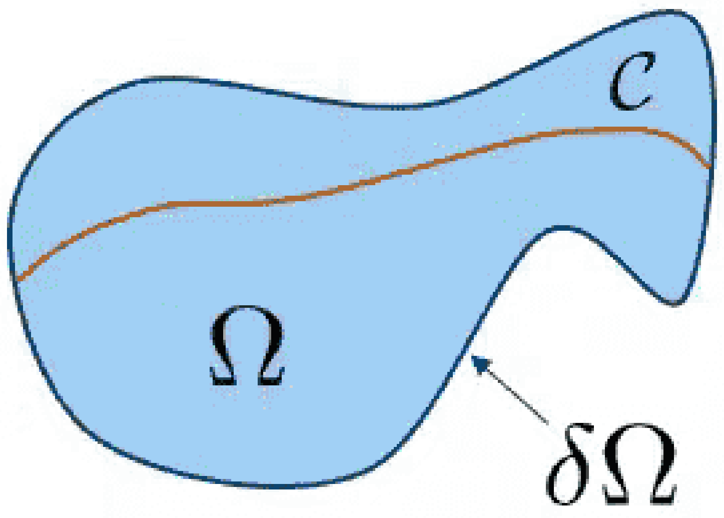
- Train the network to approximate the boundary and solve PDE



PINNs and Inverse Problems

- **Example:** Solve a boundary value problem with unknown parameter α

$$\Delta u = \alpha u,\ x \in \Omega$$
$$u(x) = g(x),\ x \in \delta\Omega$$
$$u(x) = h(x),\ x \in \mathcal{C}$$



- **PINN Solution:** Train a neural net $\hat{u}$ with domain Ω, parameter α and loss

$$\frac{\lambda_\Omega}{p}\sum_{i=1}^{p}\Delta(\hat{u}(x_i)-\alpha\hat{u}(x))^2+\frac{\lambda_{\delta\Omega}}{s}\sum_{i=1}^{s}(\hat{u}(y_i)-g(y_i))^2+\frac{\lambda_{\mathcal{C}}}{q}\sum_{i=1}^{q}(\hat{u}(z_i)-h(z_i))^2$$

$x_1,\ldots,x_p \in \Omega$ $\quad$ $y_1,\ldots,y_s \in \delta\Omega$ $\quad$ $z_1,\ldots,z_q \in \mathcal{C}$

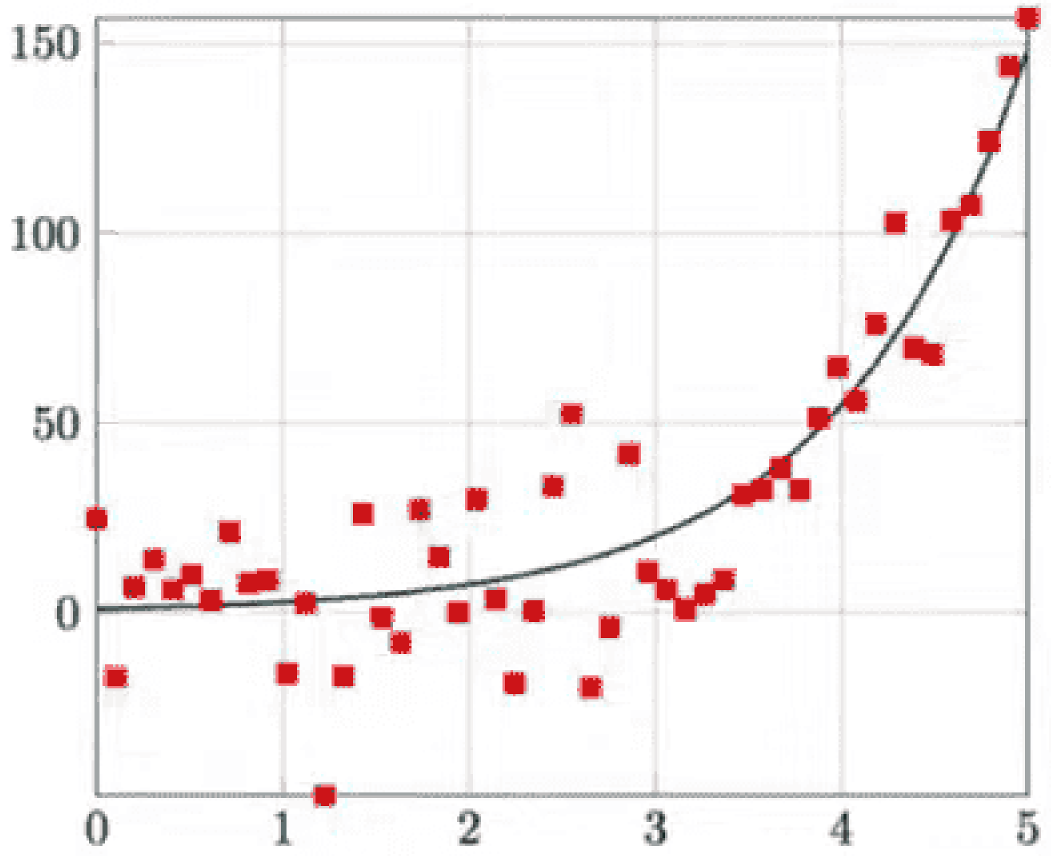
Talk Overview

- Main papers
  1. A method for learning Partial Differential Equations
     with A. Hasan, R. Ravier, S. Farsiu and V. Tarokh
  2. A method for learning latent Stochastic Differential Equations
     with A. Hasan, S. Farsiu and V. Tarokh
- Extras (work in progress, time permitting)
  - **Neural Conjugate Flows –** a causal and time-reversible architecture for Ordinary Differential Equations
    with A. Bizzi, L. Nissenbaum
  - PINNs for seismic inversion (project with Petrobras)
    with several students and postdocs at IMPA

Learning PDEs

Given:

- Noisy data points of a function
- Function is the solution to a PDE or ODE



Goals:

- Obtain an approximation of the function;
- Learn the underlying PDE/ODE

$$f(t) = e^t$$

$$f'(t) = f(t)$$

How to recover the PDE?

- PDEs are usually linear combinations of simple derivative terms
- Examples

Wave equation (1D)	$u_{tt} - u_{xx} = 0$
Heat equation (1D)	$u_t - u_{xx} = 0$
Helmholtz Equation (2D)	$u_{xx} + u_{yy} + u = 0$
Inviscid Burgers equation	$u_t + uu_x = 0$
Korteweg-de Vries equation	$u_t - 6uu_x + u_{xxx} = 0$

How to recover the PDE: A dictionary of deri

- The user defines a dictionary of possible derivative terms
- Assume the PDE is a linear combination of these terms

User defined dictionary

$$a_1 u + a_2 u_{xx} + a_3 u u_x + a_4 u_{xxx} + a_5 u_t = 0$$

Learn linear coefficients

- Example: Heat Equation

$$1 u_{xx} \qquad + (-1) u_t = 0$$

How to recover the PDE: A dictionary of deri

- The user defines a dictionary of possible derivative terms
- Assume the PDE is a linear combination of these terms

User defined dictionary

$$a_1 u + a_2 u_{xx} + a_3 u u_x + a_4 u_{xxx} + a_5 u_t = 0$$

Learn linear coefficients

- Example: Korteweg-de Vries equation

$$-6 u u_x + 1 u_{xxx} + (-1) u_t = 0$$

How to recover the PDE from a dictionary of deriva

- Sample random points in domain $p_1, \ldots, p_K$

- If u is a solution of the PDE

$$a_1 u + \quad a_2 u_{xx} + \quad a_3 u u_x + \quad a_4 u_{xxx} + \quad a_5 u_t \quad = 0$$

- For all $p_1, \ldots, p_K$

$$a_1 u(p_k) + a_2 u_{xx}(p_k) + a_3 u(p_k) u_x(p_k) + a_4 u_{xxx}(p_k) + a_5 u_t(p_k) = 0$$

How to recover the PDE from a dictionary of deriva

- In matrix form:

$$\underbrace{\begin{bmatrix} u(p_1) & u_{xx}(p_1) & u(p_1)u_x(p_1) & u_{xxx}(p_1) & u_t(p_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u(p_K) & u_{xx}(p_K) & u(p_K)u_x(p_K) & u_{xxx}(p_K) & u_t(p_K) \end{bmatrix}}_{\mathcal{M}_u(\mathbf{p})} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = 0$$

- The vector $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5)$ is in the null space of $\mathcal{M}_u(\mathbf{p})$

How to recover the PDE from a dictionary of deriva

- In matrix form: $\mathcal{M}_u(\mathbf{p})\,\mathbf{a} = 0$

- Null space vector is singular vector with singular value 0

- Obtain null space by finding singular vector with smallest singular value

- Calculate smallest singular value using min-max principle

$$\begin{aligned} \min_{\mathbf{a}} \quad & \|\mathcal{M}_u(\mathbf{p})\,\mathbf{a}\|_2^2 \\ \text{subject to} \quad & \|\mathbf{a}\|_2 = 1 \end{aligned}$$

Bringing together the losses

- Fitting the neural network $\hat{u}(\cdot;\theta)$ to the data

$$\mathcal{L}_{\text{fit}}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\tilde{u}_i - \hat{u}(\tilde{p}_i;\theta))^2$$

Fit the function at sample points $(\tilde{u}_i, \tilde{p}_i)$

- Learning the PDE

$$\mathcal{L}_{\text{PDE}}(\theta, \mathbf{a}) = \|\mathcal{M}_{\hat{u}(\cdot;\theta)}(\mathbf{p})\,\mathbf{a}\|_2^2$$

1. Sample random points
2. Evaluate dictionary terms to build this matrix
3. Calculate derivatives with auto-differentiation

- Encourage law sparsity

$$\mathcal{L}_{\ell_1}(\mathbf{a}) = \|\mathbf{a}\|_1$$

Bringing together the losses

- Training

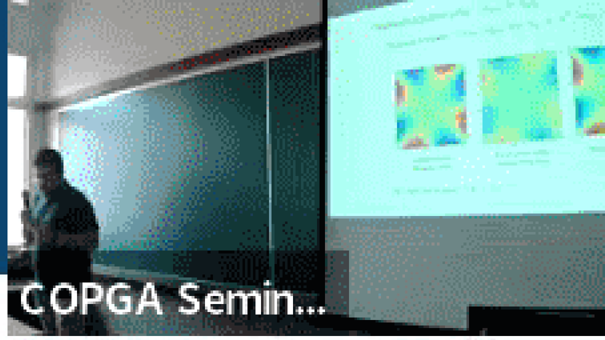
$$\min_{\{\theta,\mathbf{a}\}} \quad \lambda_{\text{fit}}\mathcal{L}_{\text{fit}}(\theta)(1+\lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}}(\theta,\mathbf{a})+\lambda_{\text{sp}}\mathcal{L}_{\text{sp}}(\mathbf{a}))$$

$$\text{subject to} \quad \|\mathbf{a}\|=1$$

Enforced by
1. projecting gradient after back-propagation
2. rescaling after optimization step

- Additional feature:
  - Minimizing $\mathcal{L}_{\text{PDE}}(\theta,\mathbf{a})$ in terms of θ enforces the neural network to be a solution to learnt PDE
  - Learnt function is smoother

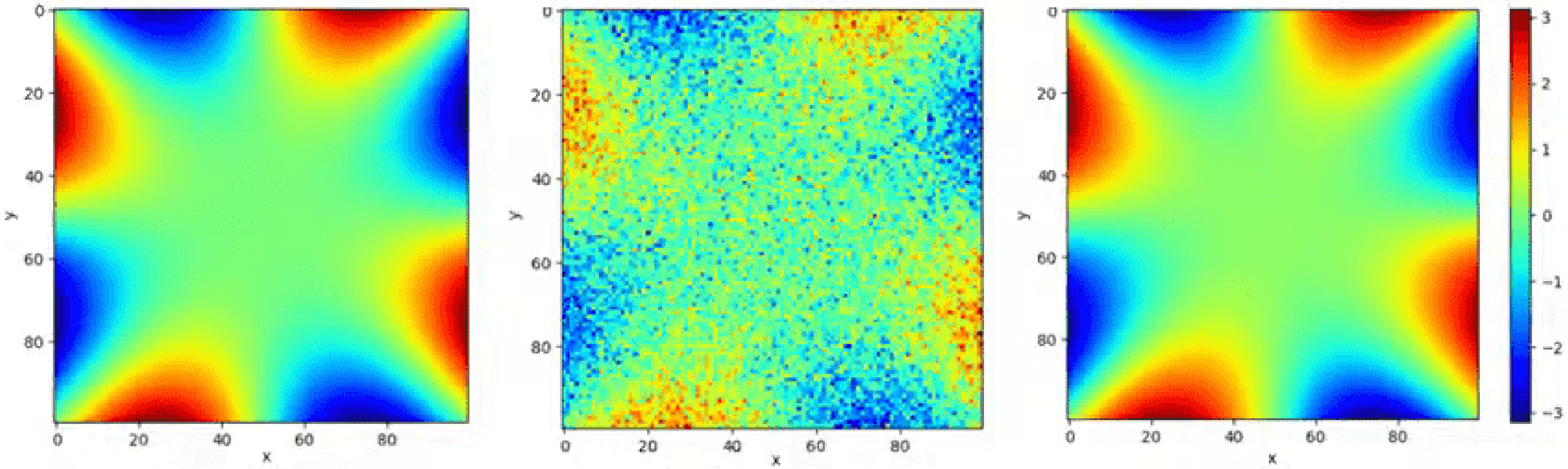
Results



Helmholtz Equation (2D): $u_{xx} + u_{yy} + u = 0$

Derivative dictionary: $(u_{xx},\ u_{yy},\ u_x,\ u_y,\ u,\ u^2,\ uu_x,\ uu_y)$

$(1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0)$

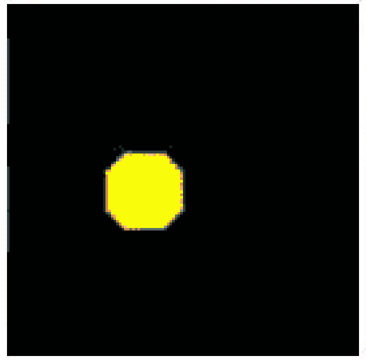


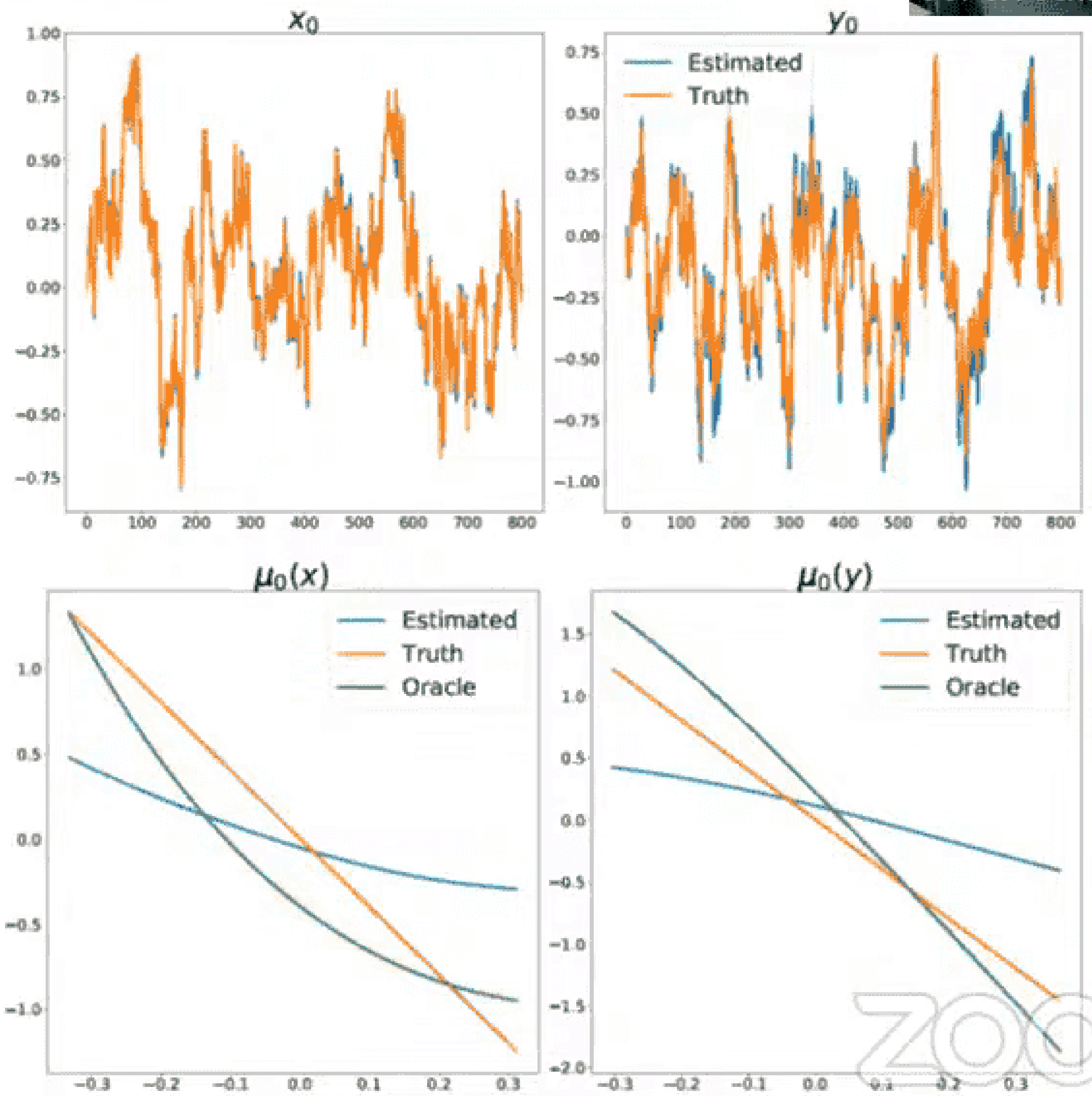
Original function (Solution to PDE) | Noisy function values (Input/Training Data) | Output of the neural network

PDE coefficient error: 3.6×10^{-2}

Second Method: Latent SDEs

Input





Crash course on SDEs

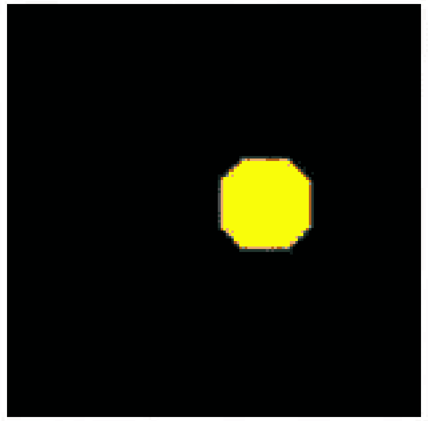
- Stochastic differential equation

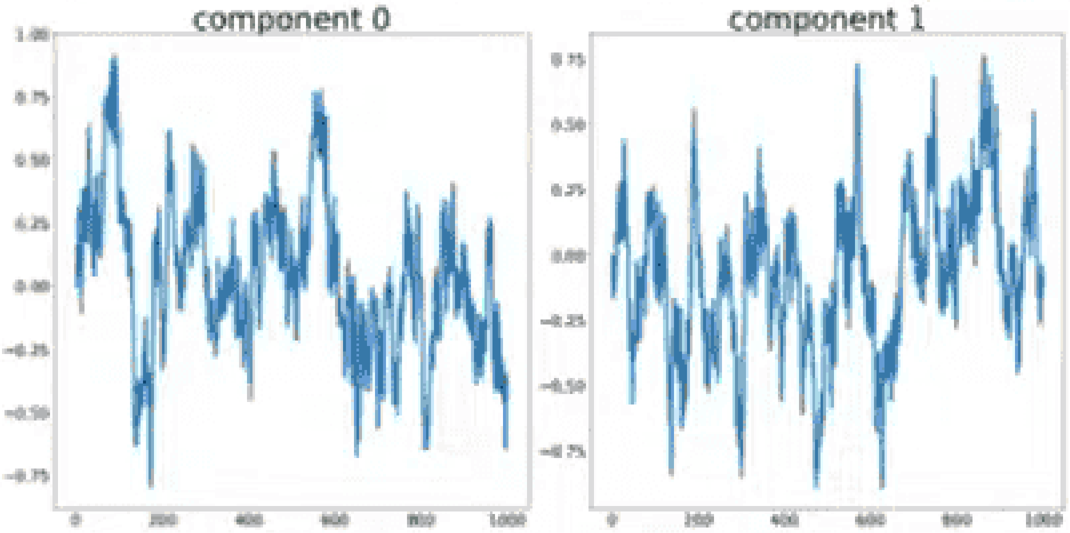
$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

Drift coefficient
(Deterministic)

Diffusion coefficient
(Stochastic)

Model





Input: X_t

Latent SDE: Z_t

$$X_t = f(Z_t) + \epsilon_t$$

$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

Model parameters: $f,\ \mu,\ \sigma$

Itô's lemma

- Suppose Z_t is a solution of the SDE

$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

- Then $Y_t = g(Z_t)$ is a solution of other SDE

$$dY_t = \tilde{\mu}(Y_t)dt + \tilde{\sigma}(Y_t)dW_t$$

- The formula for $\tilde{\mu}, \tilde{\sigma}$ in terms of μ, σ, g is given by Itô's lemma

Which model is the true one? Both can be!

$$X_t = f(Z_t) + \epsilon_t$$
$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

$Y_t = g(Z_t)$ ⇕ $f = \tilde{f} \circ g$

$$X_t = \tilde{f}(Y_t) + \epsilon_t$$
$$dY_t = \tilde{\mu}(Y_t)dt + \tilde{\sigma}(Y_t)dW_t$$

Which model is the true one? Both can be!

$$X_t = f(Z_t) + \epsilon_t$$

$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

Can only learn f, μ, σ up to a one-to-one transformation in latent space (g)

$$X_t = \tilde{f}(Y_t) + \epsilon_t$$

$$dY_t = \tilde{\mu}(Y_t)dt + \tilde{\sigma}(Y_t)dW_t$$

No need to learn diffusion coefficient

Theorem (Informal)

Suppose that $(f,\, \mu,\, \sigma)$ are the true underlying model parameters of

$$X_t = f(Z_t) + \epsilon_t$$
$$dZ_t = \mu(Z_t)dt + \sigma(Z_t)dW_t$$

Then under some technical conditions of μ and σ, there exists $(\tilde{f},\, \tilde{\mu},\, \tilde{\sigma})$ such that

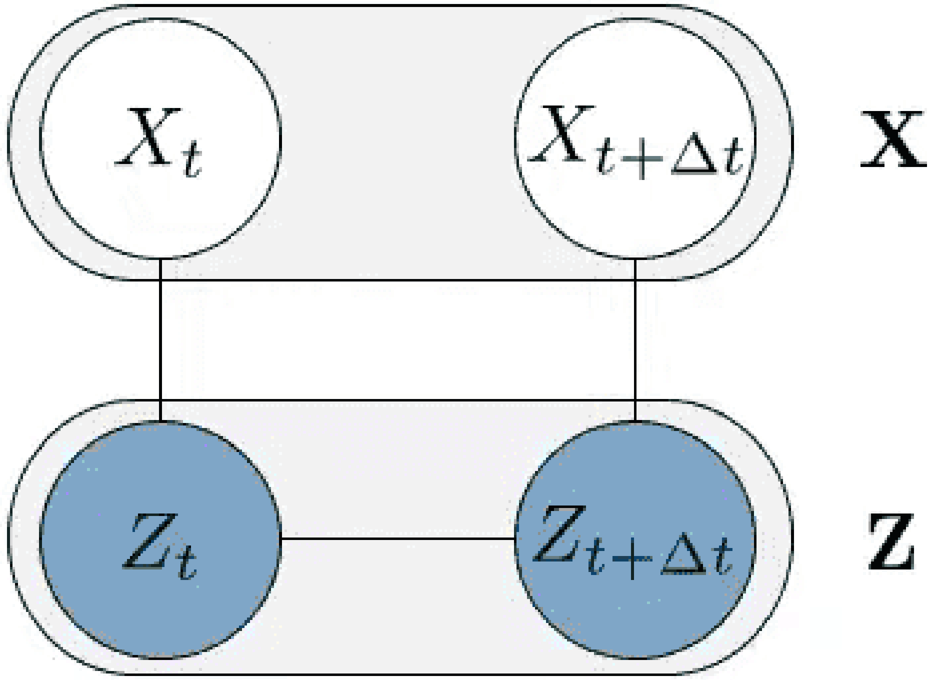
$$X_t = \tilde{f}(\tilde{Z}_t) + \epsilon_t$$
$$d\tilde{Z}_t = \tilde{\mu}(\tilde{Z}_t)dt + \tilde{\sigma}(\tilde{Z}_t)dW_t$$

and $\tilde{\sigma}$ is isotropic, that is, $\tilde{\sigma}(z) = I_n$ for all $z \in \mathbb{R}^n$

- Can focus on learning SDE with isotropic diffusion coefficient

Variational Auto-Encoder: Encoder

$$p_\phi(\mathbf{X}, \mathbf{Z}) = p_f(X_{t+\Delta t}|Z_{t+\Delta t})p_\mu(Z_{t+\Delta t}|Z_t)p_f(X_t|Z_t)p_\gamma(Z_t).$$



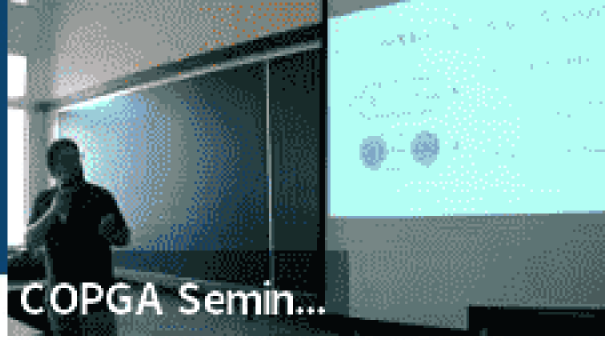
$$X_t = f(Z_t) + \epsilon_t$$

$$Z_{t+\Delta t} - Z_t \approx \mathcal{N}(\mu(z_t)\Delta t, \Delta t I_n)$$

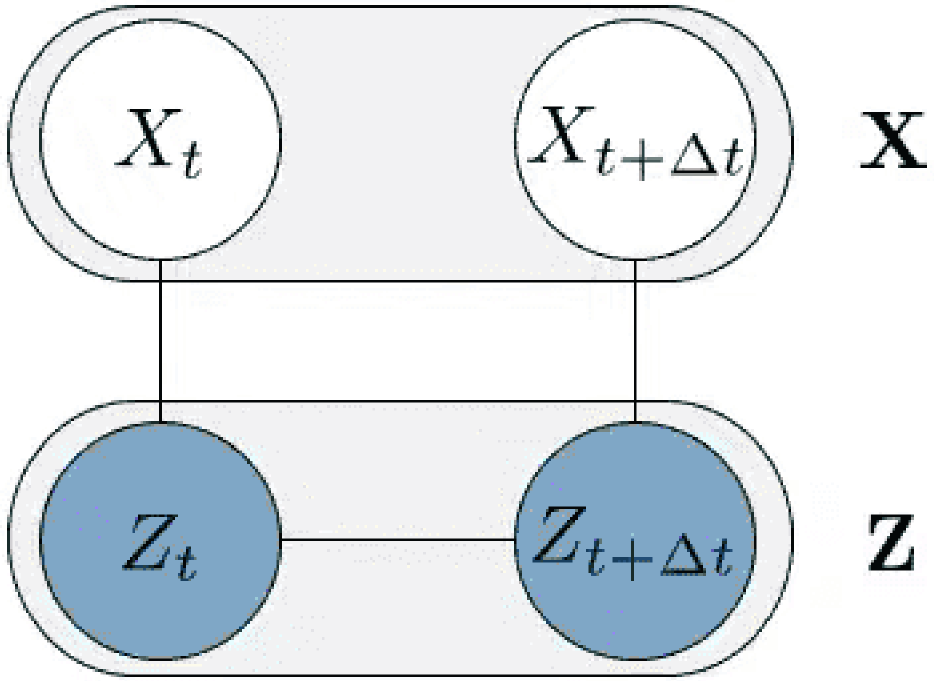
Euler-Maruyama Approximation

$$X_{t+\Delta t} = f(Z_{t+\Delta t}) + \epsilon_{t+\Delta t}$$

Variational Auto-Encoder: Encoder



$$p_\phi(\mathbf{X}, \mathbf{Z}) = p_f(X_{t+\Delta t}|Z_{t+\Delta t})p_\mu(Z_{t+\Delta t}|Z_t)p_f(X_t|Z_t)p_\gamma(Z_t).$$



$$p_f(X_t|Z_t) = p_\epsilon(X_t - f(Z_t))$$

$$p_\mu(Z_{t+\Delta t}|Z_t) = \frac{1}{(2\pi\Delta t)^{\frac{d}{2}}}\exp\left(-\frac{\|Z_{t+\Delta_t} - Z_t - \mu(Z_t)\Delta t\|^2}{2\Delta t}\right)$$

$$p_f(X_{t+\Delta t}|Z_{t+\Delta t}) = p_\epsilon(X_{t+\Delta t} - f(Z_{t+\Delta t}))$$

Variational Auto-Encoder: Decoder and Los

- Decoder

$$q_\psi(\mathbf{Z}|\mathbf{X}) = q_{\psi_1}(Z_{t+\Delta t}|X_{t+\Delta t}, Z_t)q_{\psi_2}(Z_t|X_t)$$

Ensures $q_\psi(\mathbf{Z}|\mathbf{X})$ approximates $p_\phi(\mathbf{Z}|\mathbf{X})$

Maximizes the likelihood of $p_\phi(\mathbf{X})$

- Loss

$$\begin{aligned}\mathcal{L}(\phi,\psi) &= D_{KL}\left(q_\psi(\mathbf{Z}|\mathbf{X})q_\mathcal{D}(\mathbf{X}) \,\|\, p_\phi(\mathbf{Z}|\mathbf{X})q_\mathcal{D}(\mathbf{X})\right) - \mathbb{E}_{q_\mathcal{D}(\mathbf{X})}\left[p_\phi(\mathbf{X})\right], \\ &= \mathbb{E}_{q_\mathcal{D}(\mathbf{X})}\left[\mathbb{E}_{q_\psi(\mathbf{Z}|\mathbf{X})}\left[\log q_\psi(\mathbf{Z}|\mathbf{X}) - \log p_\phi(\mathbf{X},\mathbf{Z})\right]\right].\end{aligned}$$

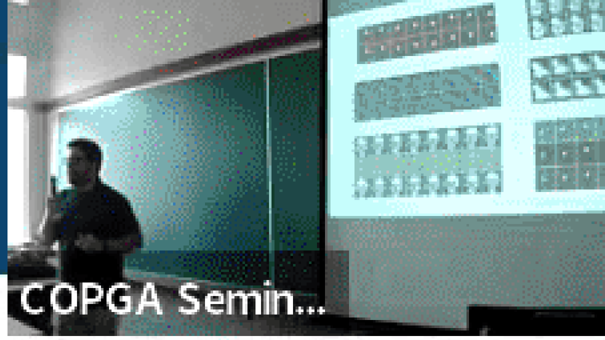
Identifiability

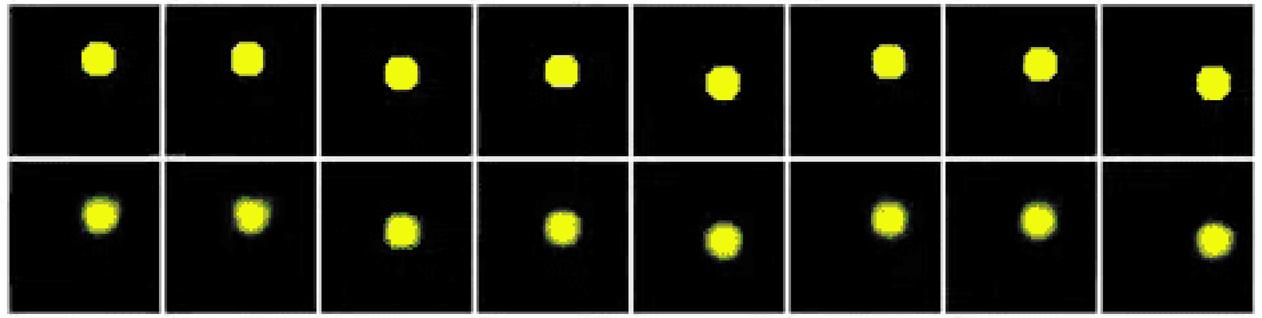
Theorem (Informal)

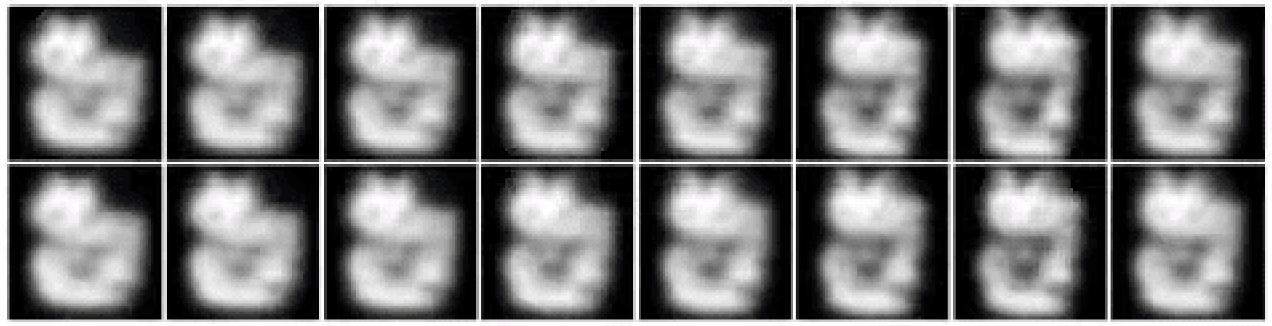
Suppose that the true generative model of $\mathbf{X}$ has true parameters (f^*, μ^*, γ^*). Then, under several technical conditions, and in the limit of infinite data, the proposed variational auto-encoder we obtain the true model up to an isometry. That is, there exist a matrix $Q \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$ such that the learnt parameters (f, μ, γ) and the true parameters (f^*, μ^*, γ^*) are related through:

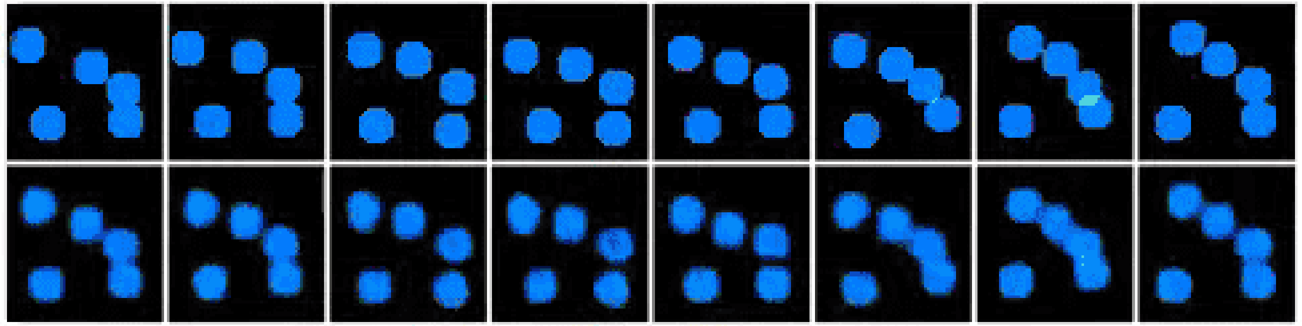
$$f(z) = f^*(Qz + b), \quad \forall z \in \mathbb{R}^n$$
$$\mu(z) = Q^T \mu^*(Qz + b), \quad \forall z \in \mathbb{R}^n$$
$$p_\gamma(z) = p_{\gamma^*}(Qz + b), \quad \forall z \in \mathbb{R}^n$$

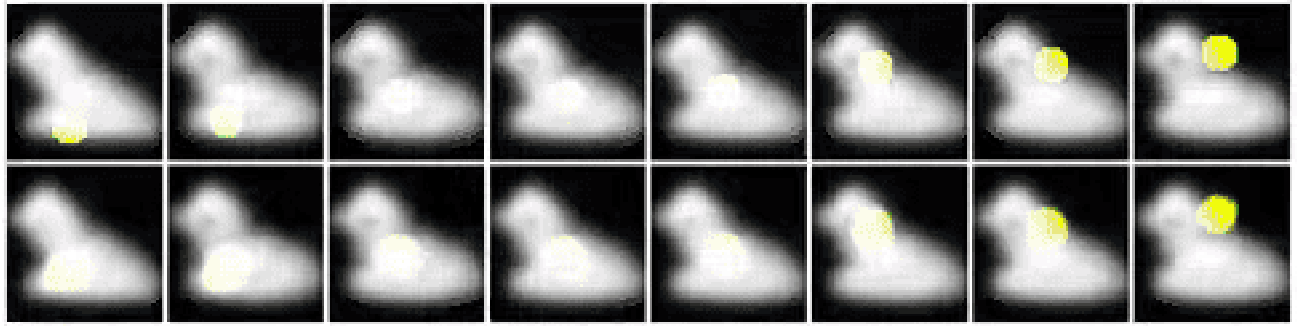
Results

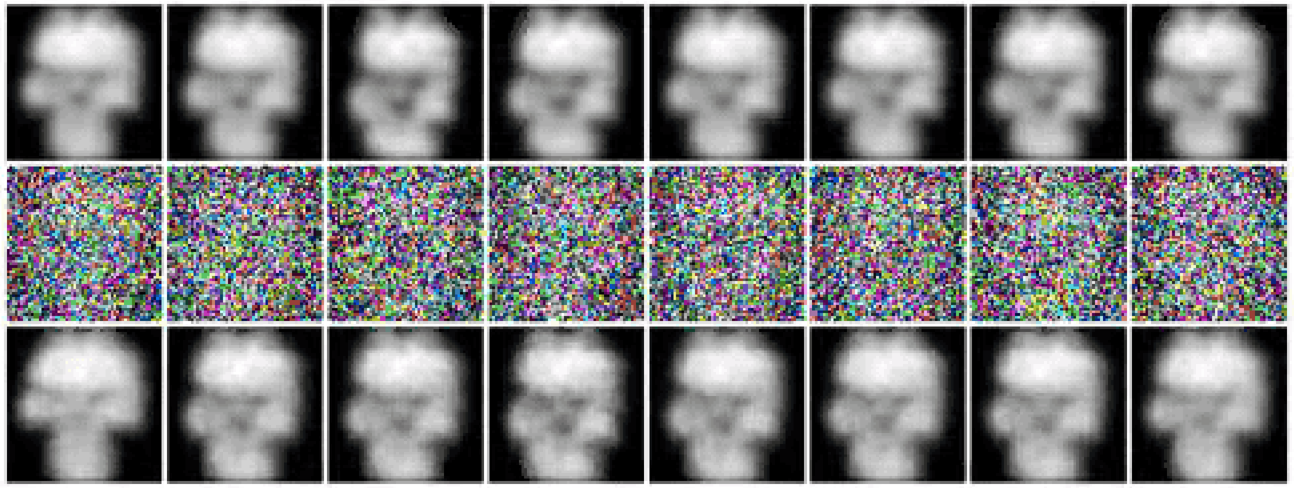


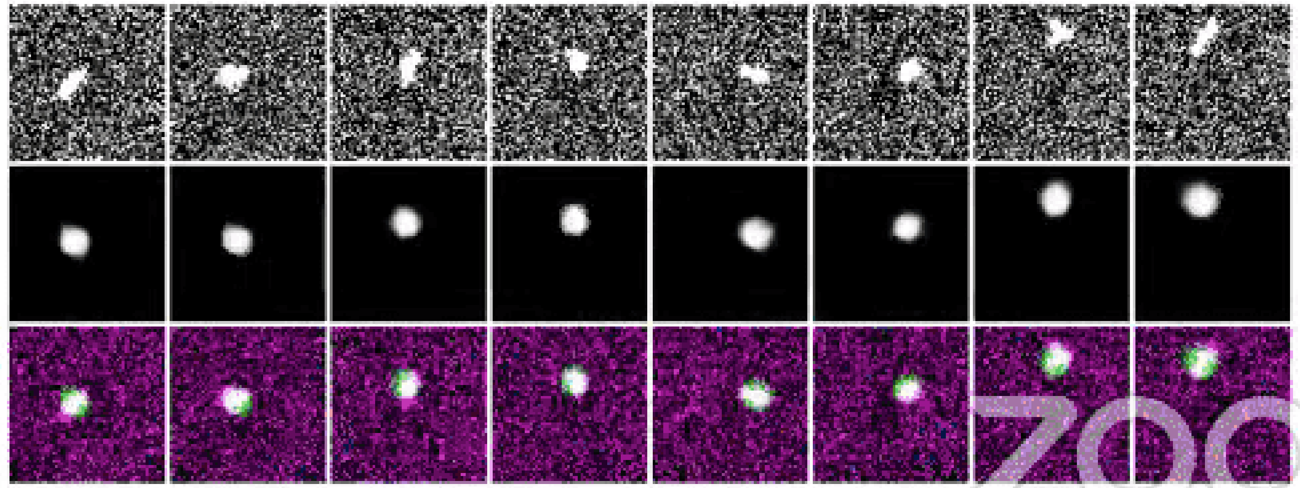












Additional results with the paper

- Variable time sampling frequency
- SDEs with time dependency
- Determining the latent dimension
- Cramér-Rao lower bounds for estimation error

Extra: Neural Conjugate Flows

- Consider an Ordinary Differential Equation

$$u_t = F(u), \quad u(0) = u_0 \in \mathbb{R}^n$$

- The flow operator has a semi-group structure:

$$\Psi_t u_0 := u(t)$$

$$\Psi_0 u_0 = u_0$$

$$\Psi_t \Psi_s = \Psi_{t+s}, \quad \forall t, s > 0$$

- Some ODEs are also reversible
  - That happens when the flow operator has a group structure $\forall t, s \in \mathbb{R}$

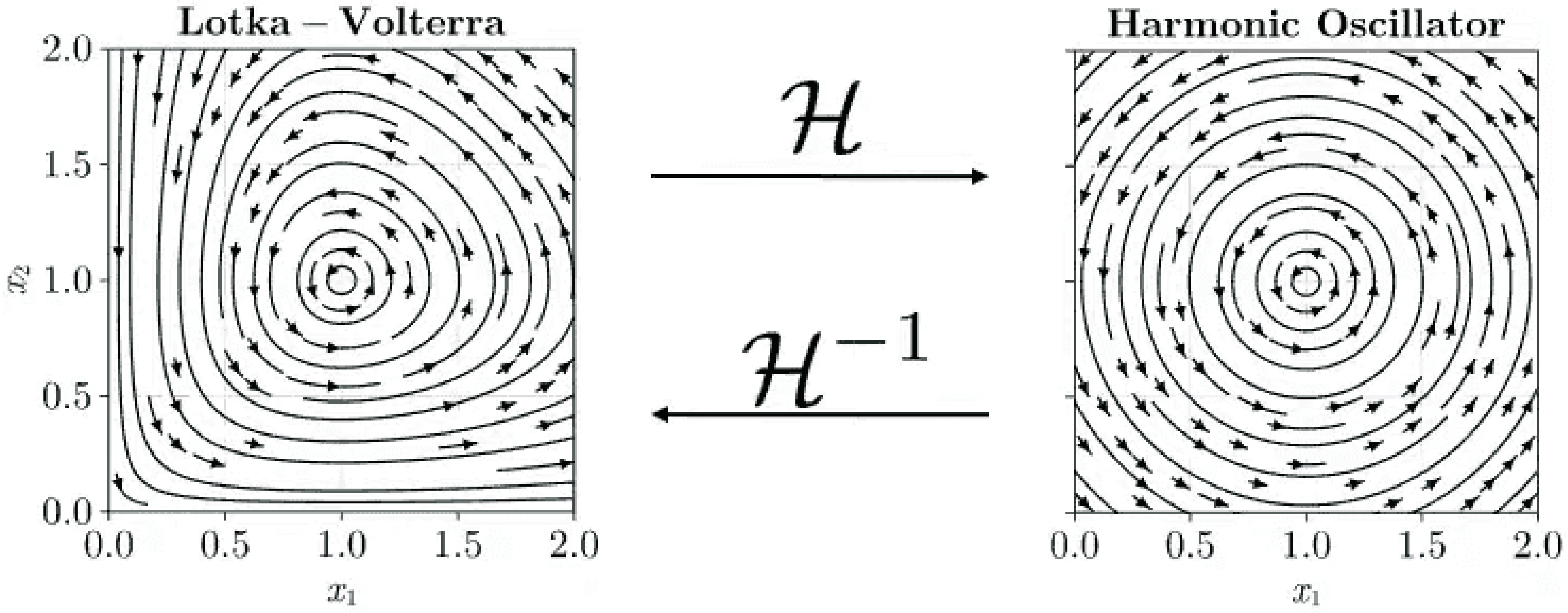
Extra: Neural Conjugate Flows

- Our architecture includes this group structure by design!

$$\Phi^t = \mathcal{H}^{-1} \circ \Psi^t \circ \mathcal{H}$$

Bijective Invertible function learnt by a neural network

Flow operator with analytic solution



Extra: Neural Conjugate Flows

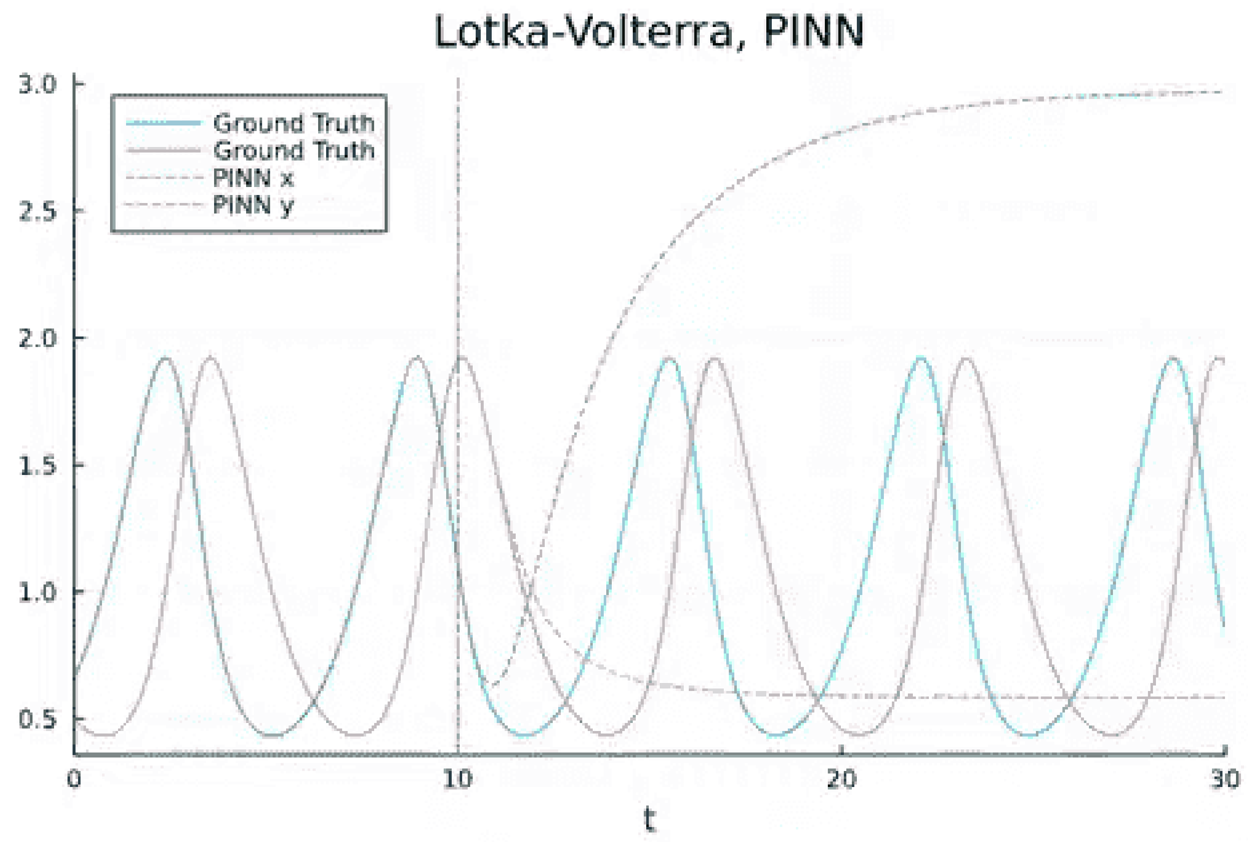
- If we know the topology of the equation, we can incorporate that knowledge directly to the architecture

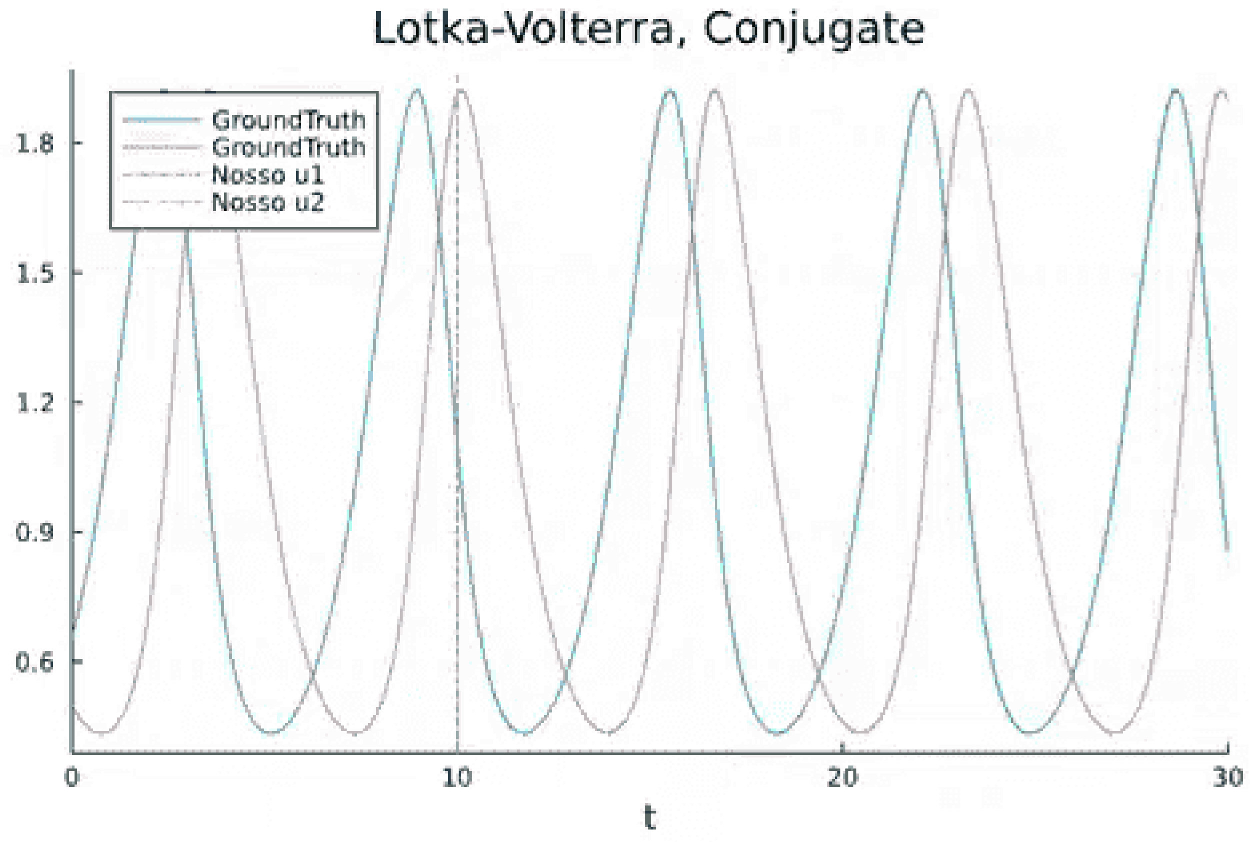
- If we do not know the topology of the equation, we can always "destroy the topology": allows us to solve any ODE problem

- We show that our neural network is an Universal Approximator for any solution of an ODE.

Extra: Neural Conjugate Flows

- Extrapolation Power





Extra: Seismic inversion with PINNs

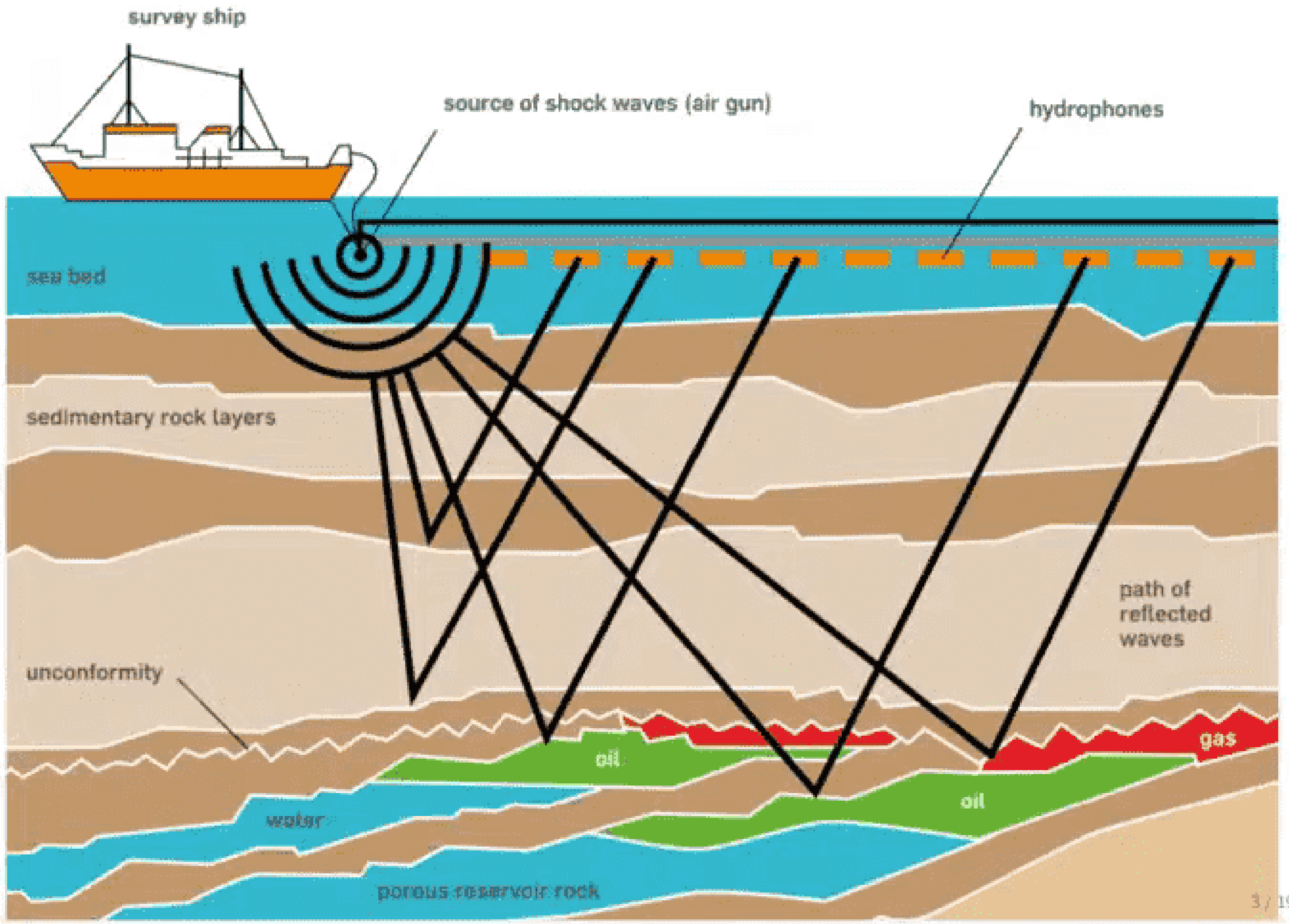
Centro Pi
Centro de Projetos
e Inovação IMPA

+

BR PETROBRAS

Team:
J. M. P + L. Nissenbaum
1 Master student
5 Ph.D students
2 Postdocs

Extra: Seismic inversion with PINNs



Extra: Seismic inversion with PINNs

- Equations

$$u_{tt}(\mathbf{x},t) - \alpha(\mathbf{x})\Delta_{\mathbf{x}}u(\mathbf{x},t) = f(\mathbf{x},t),\ \mathbf{x} \in \Omega, t \in [0,T]$$ ← Wave Equation

$$u(\mathbf{x},0) = u_0(\mathbf{x}),\ \mathbf{x} \in \Omega$$

$$u_t(\mathbf{x},0) = u_1(\mathbf{x}),\ \mathbf{x} \in \Omega$$

← Initial conditions

$$u(\mathbf{z},t) = u_{\mathcal{S}}(\mathbf{z},t),\ \mathbf{z} \in \mathcal{S}, t \in [0,T]$$ ← Seismogram Measurements

- Unknowns $u(\mathbf{x},t)$ $\alpha(\mathbf{x})$

The end! Questions?

Papers:

A. Hasan, **J. M. P**, R. Ravier, S. Farsiu and V. Tarokh
Learning partial differential equations from data using neural networks
ICASSP 2020, pp. 3962–3966, 2020.

A. Hasan, **J. M. P**, S. Farsiu and V. Tarokh
Identifying Latent Stochastic Differential Equations with Variational Auto-Encoders
IEEE Transactions of Signal Processing, 2020.

A. Bizzi, L. Nissenbaum, **J. M. P**,
Neural Conjugate Flows: a Physics-Informed Architecture with Differential Flow Structure
In Preparation

Code:

https://github.com/alluly/pde-estimation

https://github.com/alluly/ident-latent-sde