SAISON 22/23

Formation

Introduction au Deep Learning

19

Des neurones pour la physique
Les Physics-informed neural networks

FIDLE

MIAI Grenoble Alpes · cnrs · UGA Université Grenoble Alpes · EFELIA - MIAI ECOLE FRANÇAISE DE L'INTELLIGENCE ARTIFICIELLE · DEVLOG · Resinfo · sari · ANITI · GRICAD · SIMaP

List and Networks

You can also subscribe to :

http://fidle.cnrs.fr/listeinfo
Fidle information list

https://listes.services.cnrs.fr/wws/info/devlog
List of ESR* « Software developers » group

https://listes.math.cnrs.fr/wws/info/calcul
List of ESR* « Calcul » group

(*) ESR is Enseignement Supérieur et Recherche, french universities and public academic research organizations

Ressources

https://fidle.cnrs.fr

Powered by CNRS CRIC, and UGA DGDSI
of Grenoble, Thanks !

- Course materials (pdf)
- Practical work environment*
- Corrected notebooks
- Videos (YouTube)

(*) Procedure via Docket or pip
Remember to get the latest version !

Program

20 Séquences
du 17 novembre
au 14 mai 2023

1. **History, Fundamental Concepts**
   - **Basic Regression** — DNN
   - **Basic Classification** — DNN
2. 3. **Hight Dimensionnal Data** — CNN
4. Demystify **mathematics** for neural networks.
5. **Training strategies** — Evaluation / **Sparse data (text)** — Embedding
6. **Sequences data** — RNN
7. PyTorch — A small detour with **PyTorch**.
8. **«Attention is All You Need»** — Transformers
9. New ! **Graph Neural Network** — GNN
10. **Autoencoder networks** — AE
11. **Variational Antoencoder** — VAE
12. ▸ **JDLS 2023** — **Project session** — «My project in 180 s»
13. **Generative Adversarial Networks** — GAN
14. New ! **Diffusion Model** — Text to image
15. **AI, Law, Society** and **Ethics**
16. New ! **Model and training optimization** — Resource efficiency
17. **Jean-Zay** — GPU acceleration
18. **Deep Reinforcement Learning** — RL
19. New ! **Physics-Informed Neural Networks** — PINNS
20. **JDLS 2023** — Deep Learning for Science !

FIDLE

SAISON 22/23

Roadmap

19 New !

Physics-Informed Neural Networks

PINNS

19.1 What are PINNs ?

➔ Introduction

➔ Proof of concept

19.2 Loss balancing

➔ Manual balancing, it’s ugly but it works

➔ Learning rate annealing

➔ Self adaptive, soft attention like

➔ Augmented Lagrangian Methods

19.3 Sampling & other tricks

➔ Non adaptive sampling

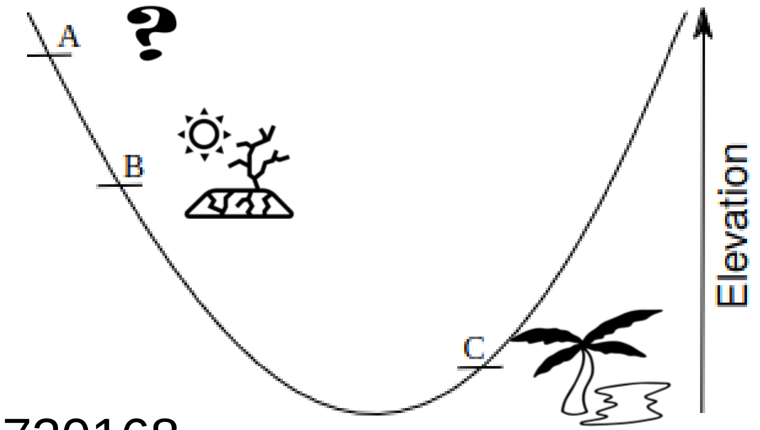
➔ Adaptive sampling

➔ Tricks you already know

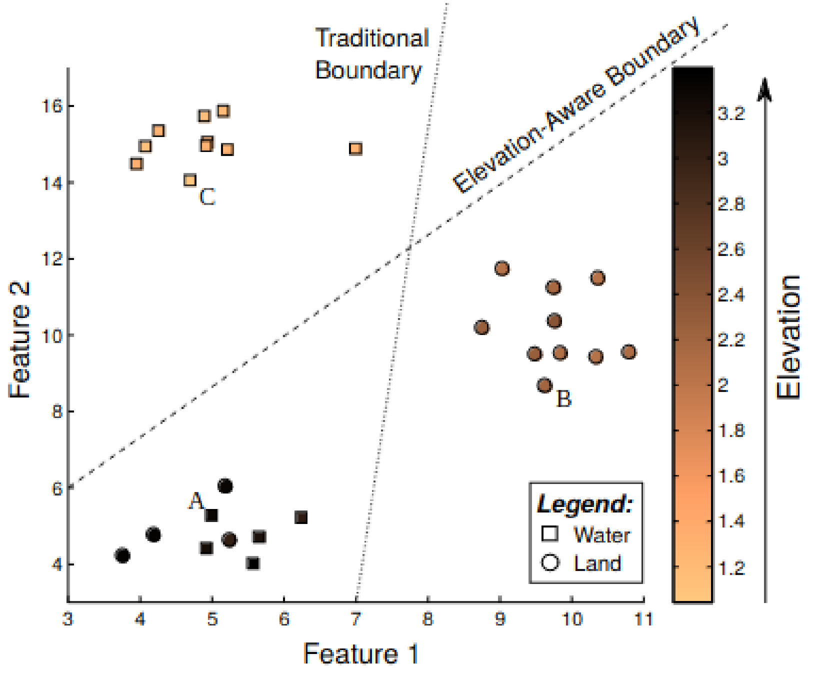
What are PINNs ?

FIDLE

Why theory guided machine learning ?



(b) Lake cross-section.



https://doi.org/10.1109/TKDE.2017.2720168

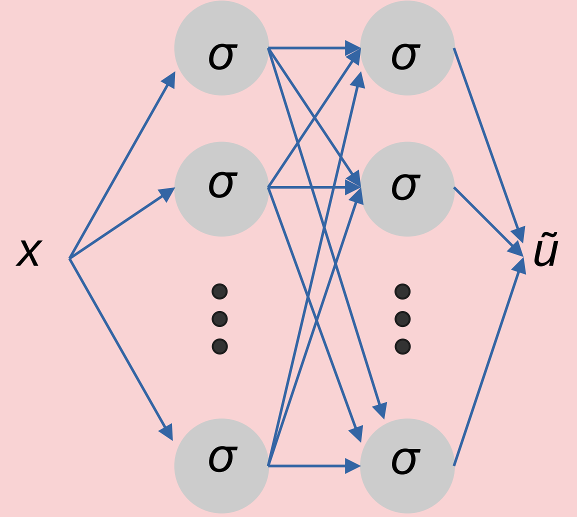
How to inform NN of relations between physical quantities ?

ODE $$\frac{dN}{dt} - kN = 0$$

PDE $$\frac{\partial \rho}{\partial t} + \vec{\nabla} \cdot (\rho \vec{V}) = 0$$

IDE $$\frac{\partial u}{\partial x} + \int_{x_1}^{x} f(t,u)dt = g(x,u)$$

- ❖ **Physics Informed NNs**
- ❖ **Deep Galerkin Method**
- ❖ **Feynman-Kac formula**
- ❖ **Deep Ritz Method**

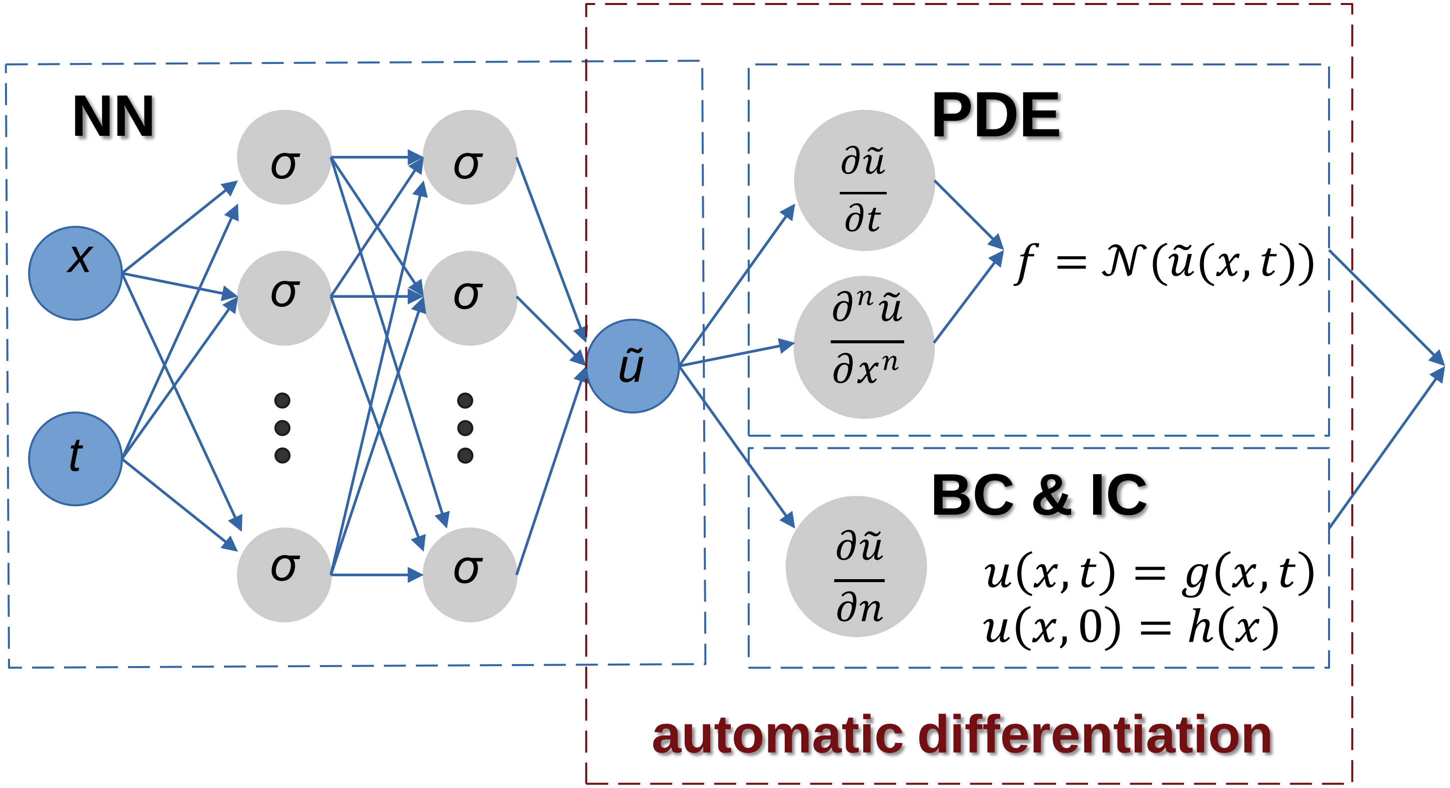


PINNs' idea

$$\mathcal{N}[u] = f, \;\; x \in \Omega, \;\; t \in [0, T]$$

$$u(x, 0) = h(x), \;\; x \in \Omega$$

$$u(x, t) = g(x, t), \;\; t \in [0, T], x \in \partial\Omega$$



$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_{BC} + \mathcal{L}_{IC}$$

Loss functions:

$$\mathcal{L}_u = \frac{1}{N_r} \sum_{i=1}^{N_r} [r(x_r^i, t_r^i)]^2$$

$$\mathcal{L}_{BC} = \frac{1}{N_b} \sum_{i=1}^{N_b} [u(x_b^i, t_b^i) - g_b^i]^2$$

$$\mathcal{L}_{IC} = \frac{1}{N_0} \sum_{i=1}^{N_0} [u(x_b^0, 0) - h_0^i]^2$$

M. Raïssi et al, https://arxiv.org/abs/1711.10561

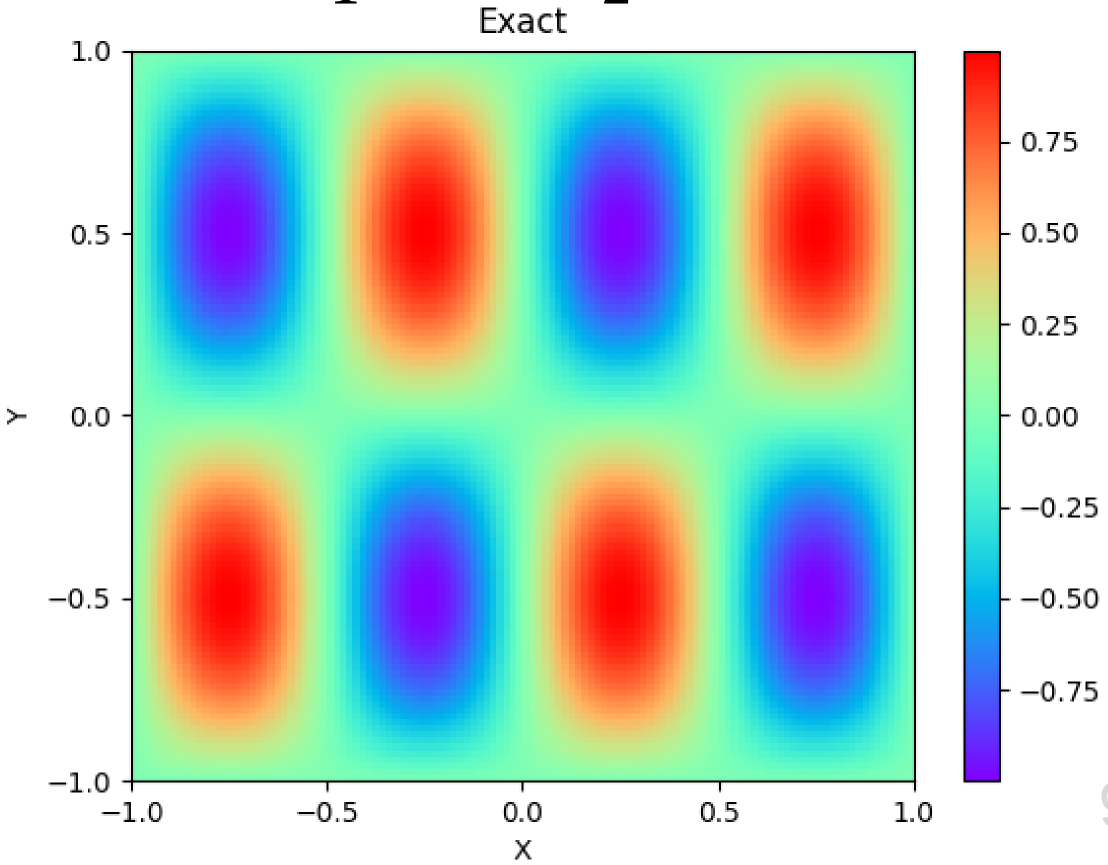
Proof of concept, an example

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u = s, \quad x \in [-1,1], \quad y \in [-1,1]$$

$$s = (1 - \pi^2({a_1}^2 + {a_2}^2)) \sin(a_1 \pi x) \sin(a_2 \pi y)$$

$$u(x,y) = 0 \ \ over \ \partial\Omega$$

$$a_1 = 2, a_2 = 1$$



$$u_{exact}(x,y) = \sin(a_1 \pi x) \sin(a_2 \pi y)$$

TP classical PINN

Notebook : 19_classical_pinns.ipynb

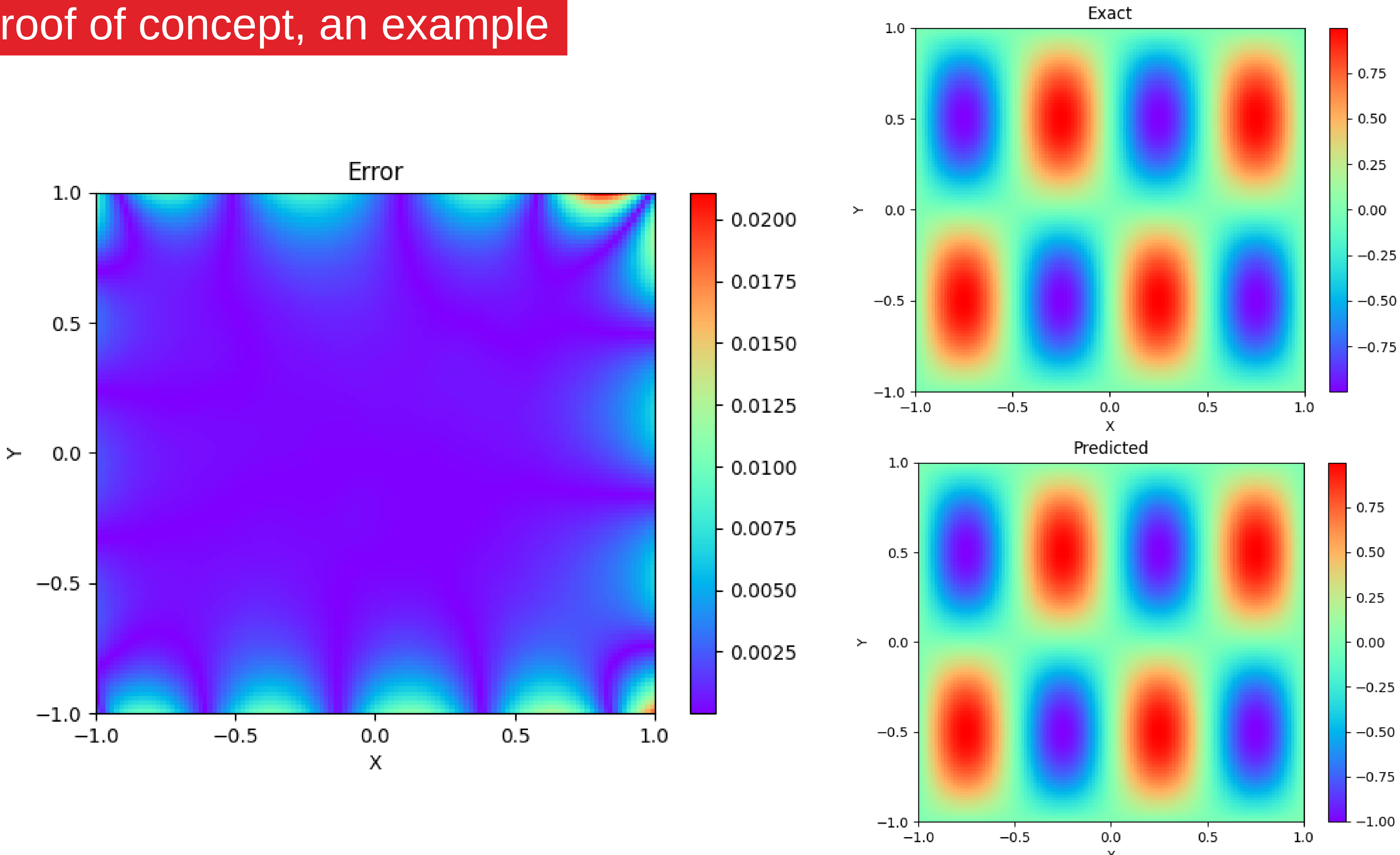
Objective :

Implement a simple PINN

Dataset :

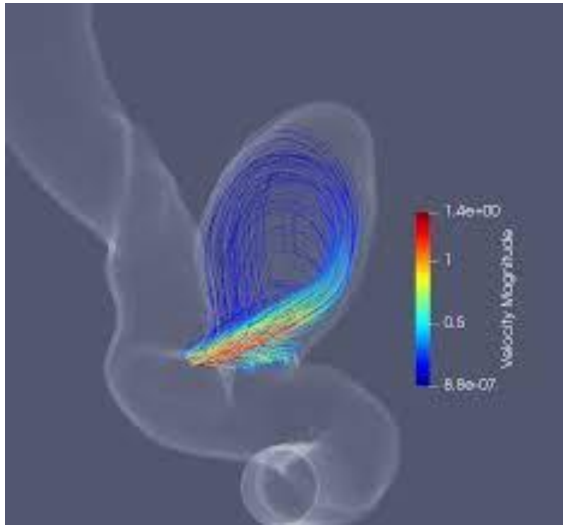
No thanks

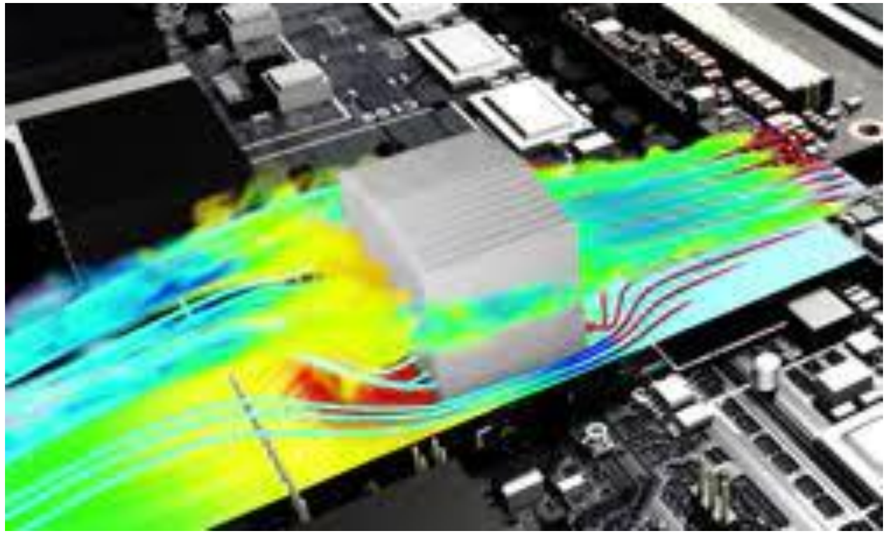
Proof of concept, an example



Summary

- **Outperformed by classical methods to solve PDE**
- **No mesh, no labeled data**
- **Ability to solve**
  - ❑ ODE, PDE, IDE
  - ❑ ill posed problems, inverse problems, parametric PDE …





Images from Nvidia Modulus

Many libraries: DeepXDE (https://doi.org/10.1137/19M1274067), Nvidia Modulus (https://docs.nvidia.com/deeplearning/modulus/index.html), NeuralPDE.jl, …

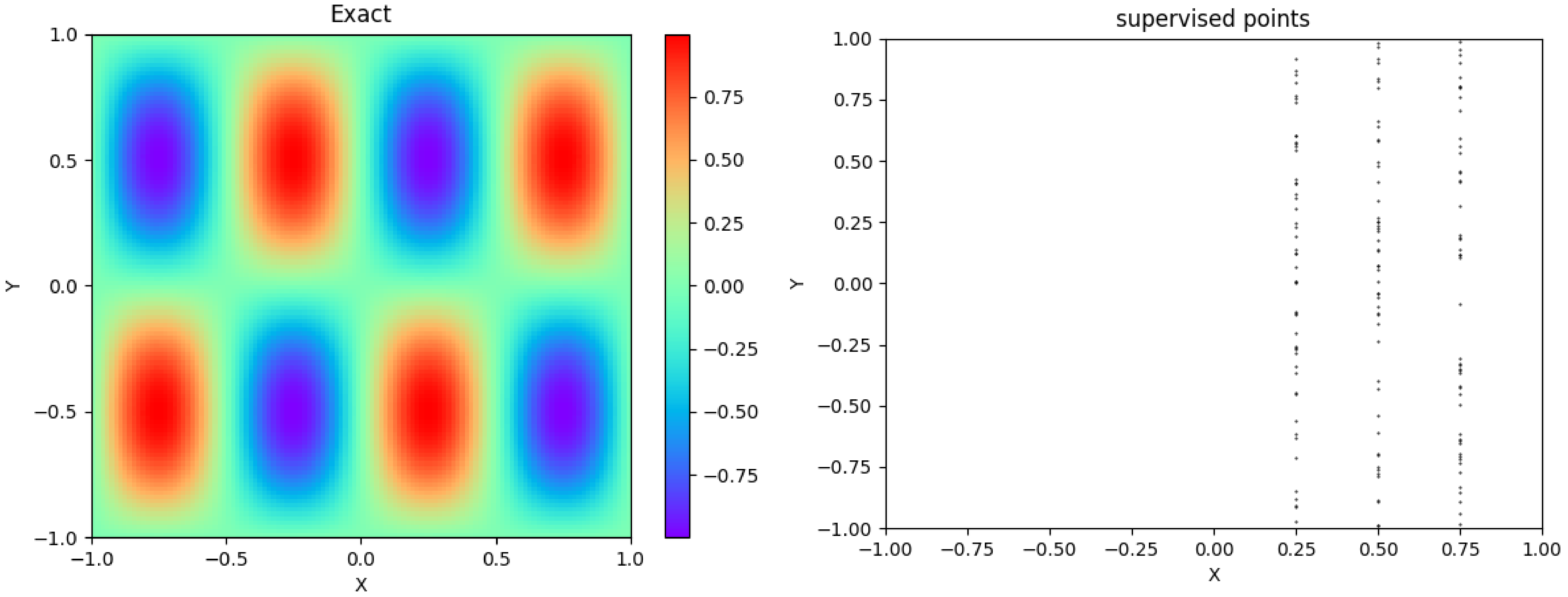
Case of study ill posed: inverse problem

Retrieve PDE from 'experimental points' (assumed PDE shape and given initial guess)

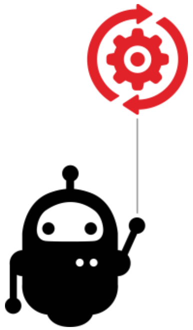
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + u = s, \quad x \in [-1,1], \quad y \in [-1,1]$$

$$s = (1 - \pi^2 ({a_1}^2 + {a_2}^2)) \sin(a_1 \pi x) \sin(a_2 \pi y), a_1 \, and \; a_2 \; unknown$$

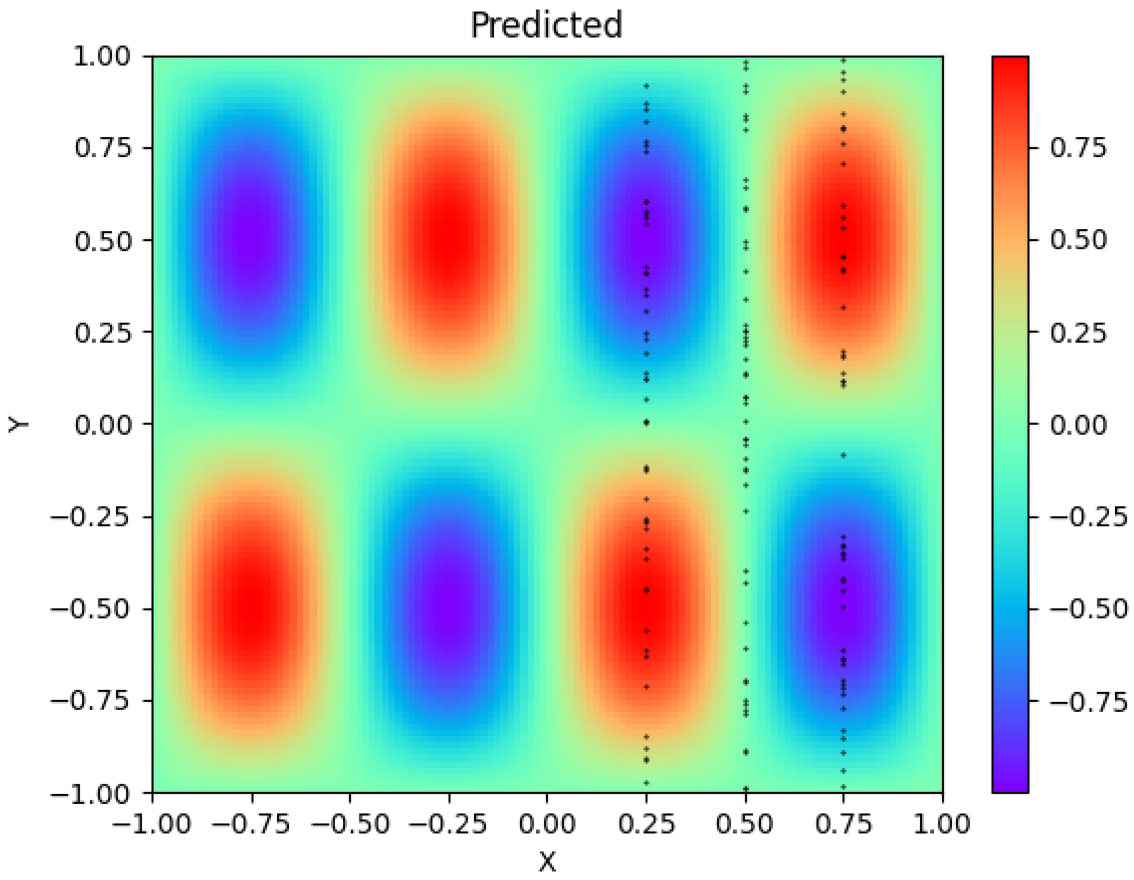
$$u(x,y) = 0 \;\; over \; \partial\Omega$$



Case of study ill posed: inverse problem



	a_1	a_2
correct	2	1
estimated	2.00027	1.00041



Try varying the initial guess for *a_1* and *a_2*, the number of points (collocation, boundaries, experimental), add noise, add terms to the PDE (and new unknown variables) …

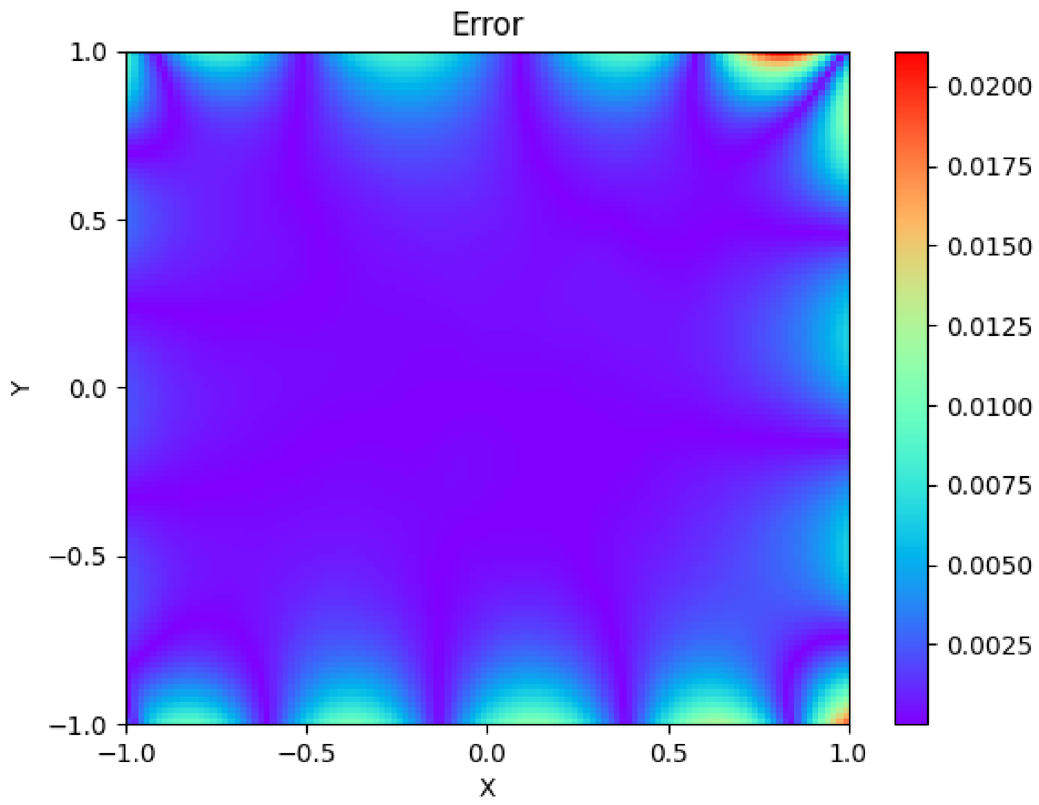
Loss regularization

FIDLE

It's naive but it works !

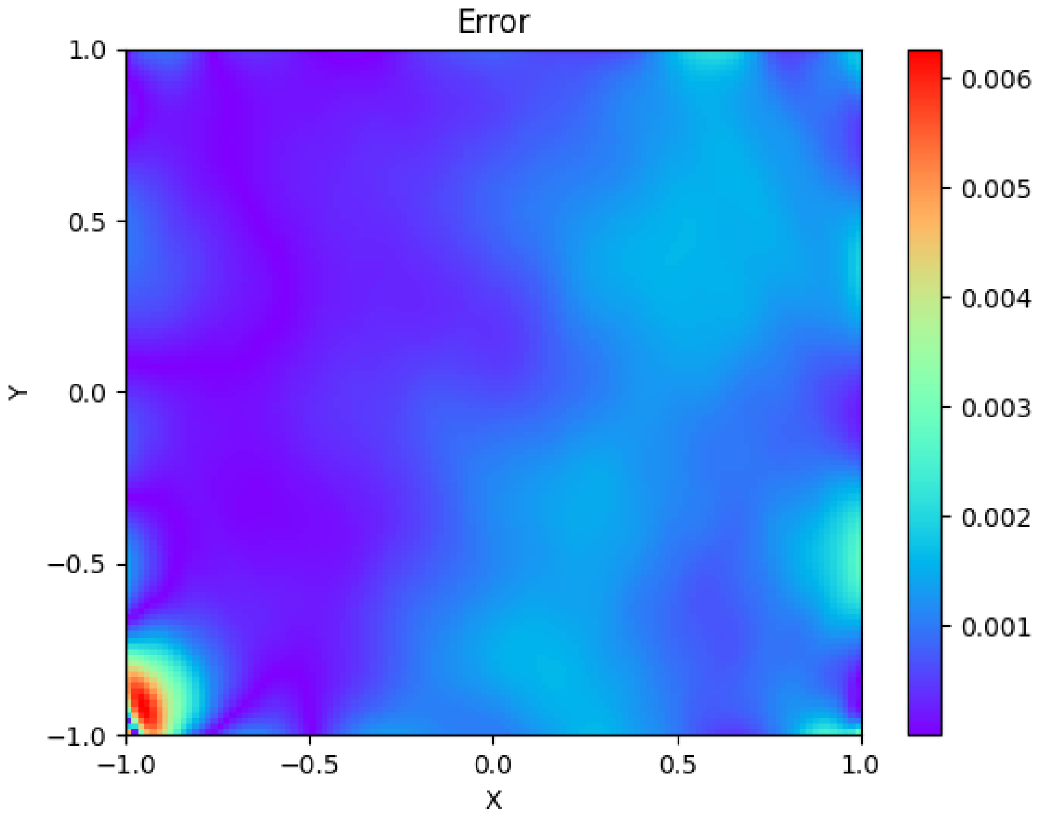
$$\mathcal{L} = \mathcal{L}_u + \omega_{BC}\mathcal{L}_{BC} + \omega_{IC}\mathcal{L}_{IC}$$

classic



$$\omega_{BC} = 1$$

naive



$$\omega_{BC} = 100$$

Zhao, C. L,https://arxiv.org/abs/2007.04542v1

Learning rate annealing

Unbalanced gradients during backpropagation are common failure for PINNs

$$\mathcal{L} = \mathcal{L}_u + \sum_i \lambda_i \mathcal{L}_i$$

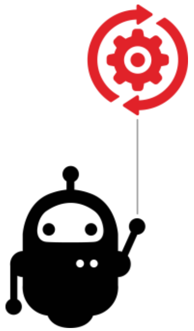
collocation **supervised (experiment, boundaries, initial ...)**

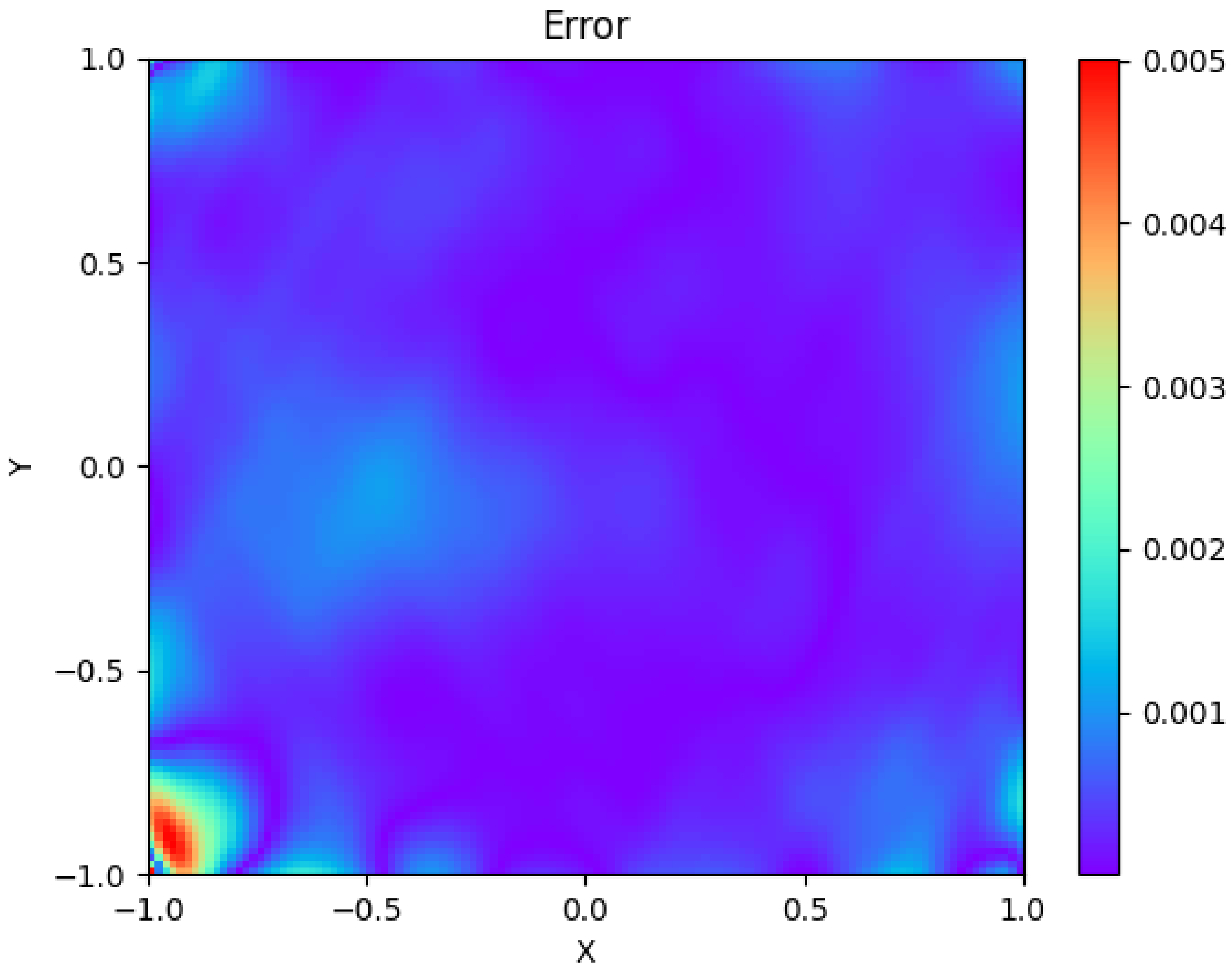
$$\lambda_i \leftarrow (1-\alpha)\lambda_i + \alpha \frac{\max_\theta(|\nabla_\theta \mathcal{L}_u|)}{\overline{|\nabla_\theta \mathcal{L}_i|}}$$

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}_u - \eta \sum_i \lambda_i \nabla_\theta \mathcal{L}_i$$

S. Wang et al, https://epubs.siam.org/doi/10.1137/20M1318043

Learning rate annealing





Self Adaptive

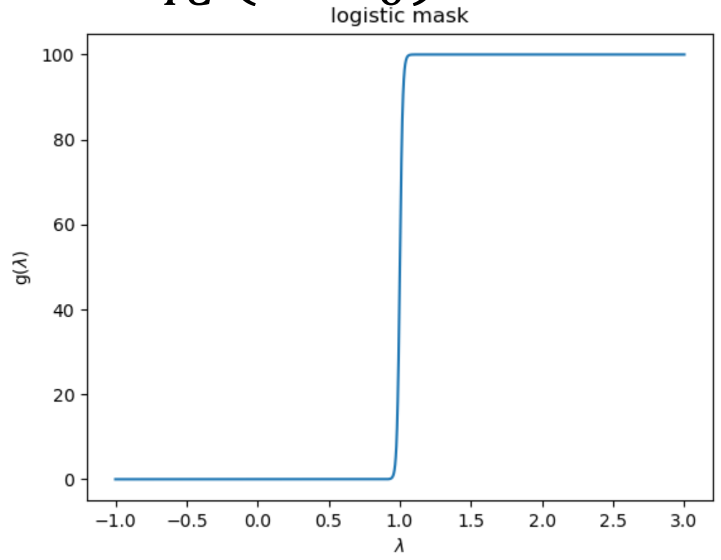
$$\mathcal{L}(\theta,\lambda_u,\lambda_b,\lambda_0)=\mathcal{L}_u(\theta,\lambda_u)+\mathcal{L}_{BC}(\theta,\lambda_b)+\mathcal{L}_{IC}(\theta,\lambda_0)$$

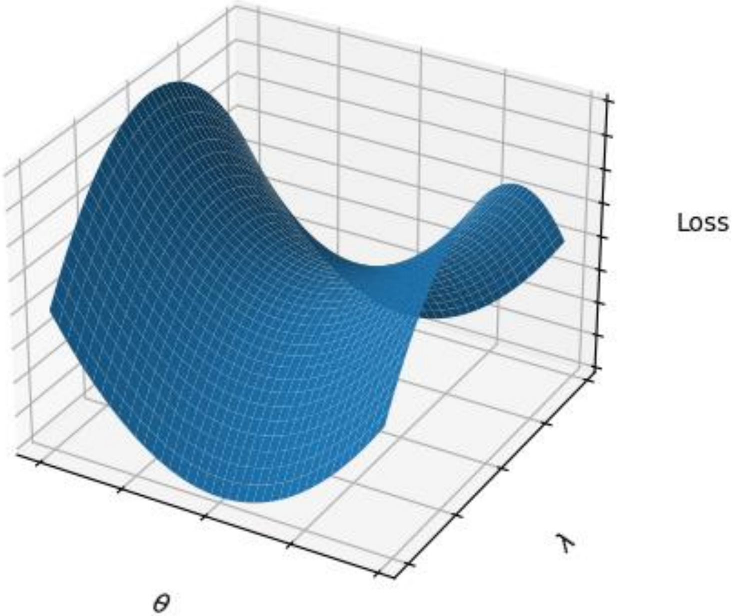
$$\mathcal{L}_u=\frac{1}{N_r}\sum_{i=1}^{N_r} g(\lambda_r^i)[r(x_r^i,t_r^i,\theta)]^2$$

$$\mathcal{L}_{BC}=\frac{1}{N_b}\sum_{i=1}^{N_b} g(\lambda_b^i)[(u(x_b^i,t_b^i,\theta)-g_b^i)]^2$$

$$\mathcal{L}_{IC}=\frac{1}{N_0}\sum_{i=1}^{N_0} g(\lambda_0^i)[(u(x_b^0,0,\theta)-h_0^i)]^2$$

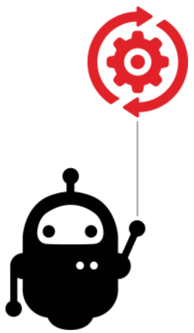
$$\min_{\theta}\max_{\lambda_u,\lambda_b,\lambda_0}\mathcal{L}(\theta,\lambda_u,\lambda_b,\lambda_0)$$

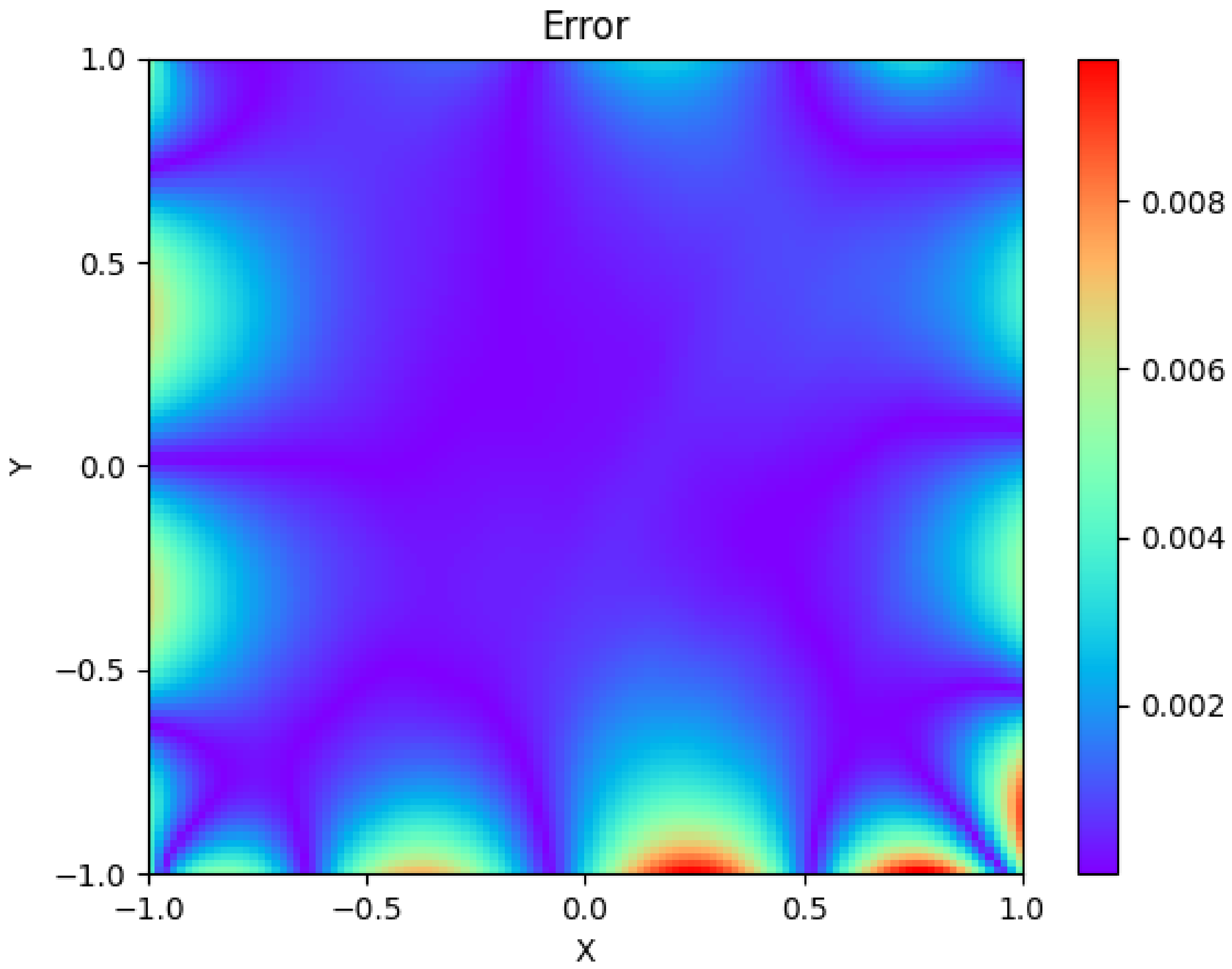




L. D. McClenny et al, https://arxiv.org/abs/2009.04544v4

Self Adaptive

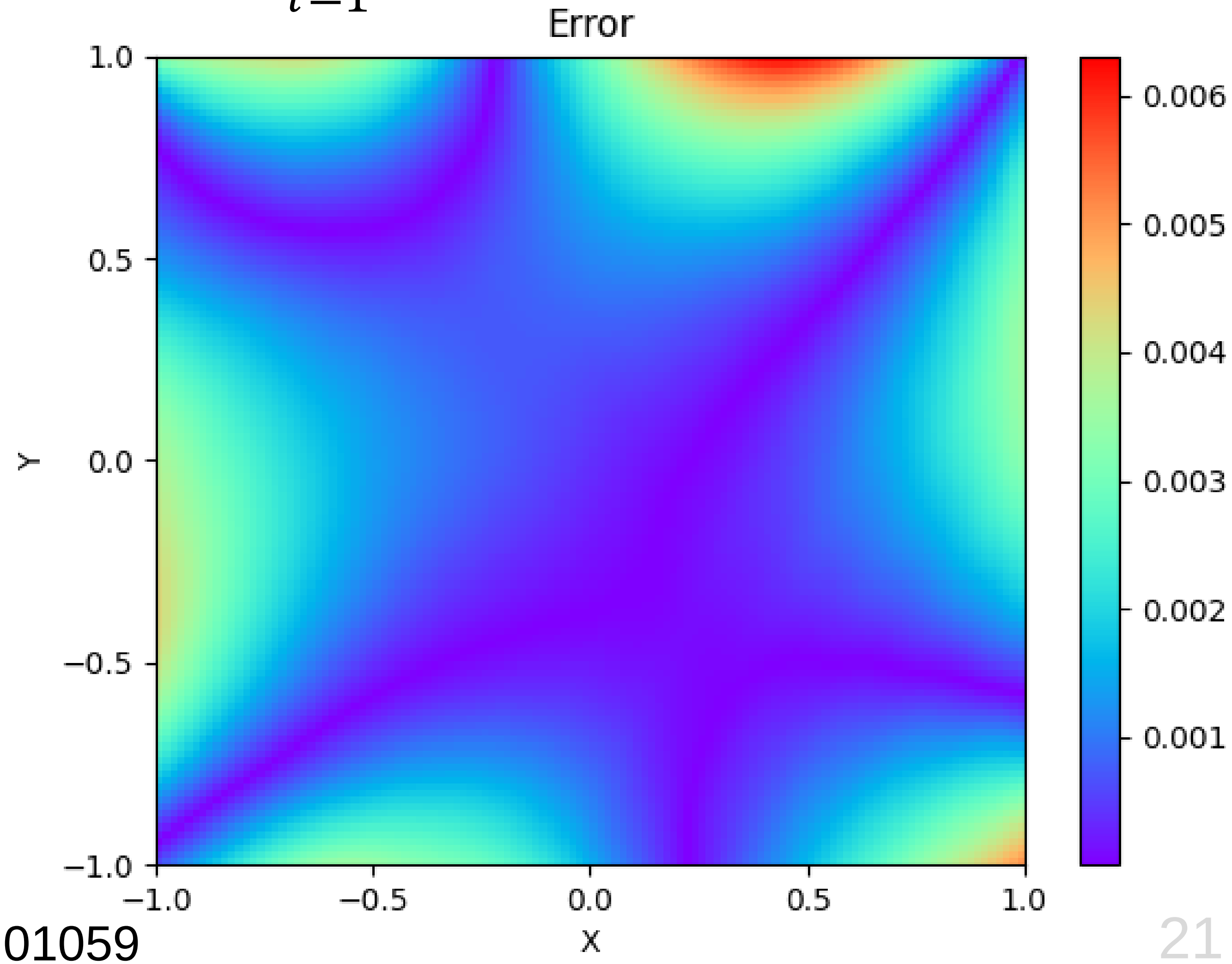




Augmented Lagrangian relaxation

$$\mathcal{L}_{BC} = \frac{\beta}{N_b}\sum_{i=1}^{N_b}[u(x_b^i, t_b^i) - g_b^i]^2 + \frac{1}{N_b}\sum_{i=1}^{N_b}\lambda_i\left(u\left(x_b^i\right) - g_b^i\right)$$

$$\min_{\theta}\max_{\lambda_i}\mathcal{L}(\theta, \lambda_i)$$



H. Son et al, https://doi.org/10.48550/arXiv.2205.01059

Other augmented Lagrangian Methods

Without modification of the descent method

LPINN: Lagrangian PINNs: A causality-conforming solution to failure modes of physics-informed neural networks, R. Mojgani et al, https://doi.org/10.48550/arXiv.2205.02902

hPINNs: Physics-Informed Neural Networks with Hard Constraints for Inverse Design. L. Lu et al, https://doi.org/10.1137/21M1397908

PECANN: Physics and Equality Constrained Artificial Neural Networks: Application to Forward and Inverse Problems with Multi-fidelity Data Fusion, Shamsulhaq Basir et al https://doi.org/10.1016/j.jcp.2022.111301

Sampling

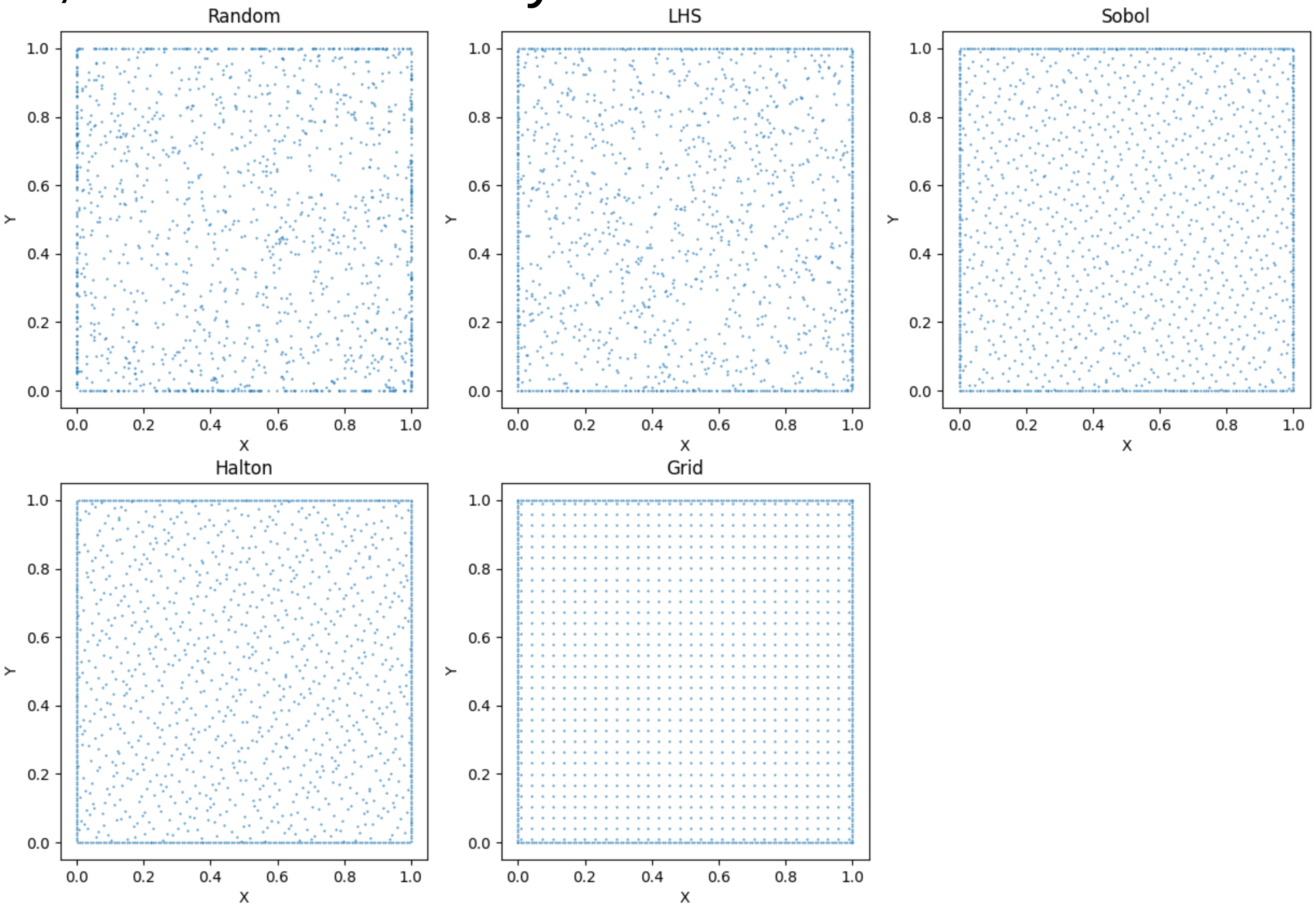
FIDLE

Basics and non adaptive sampling

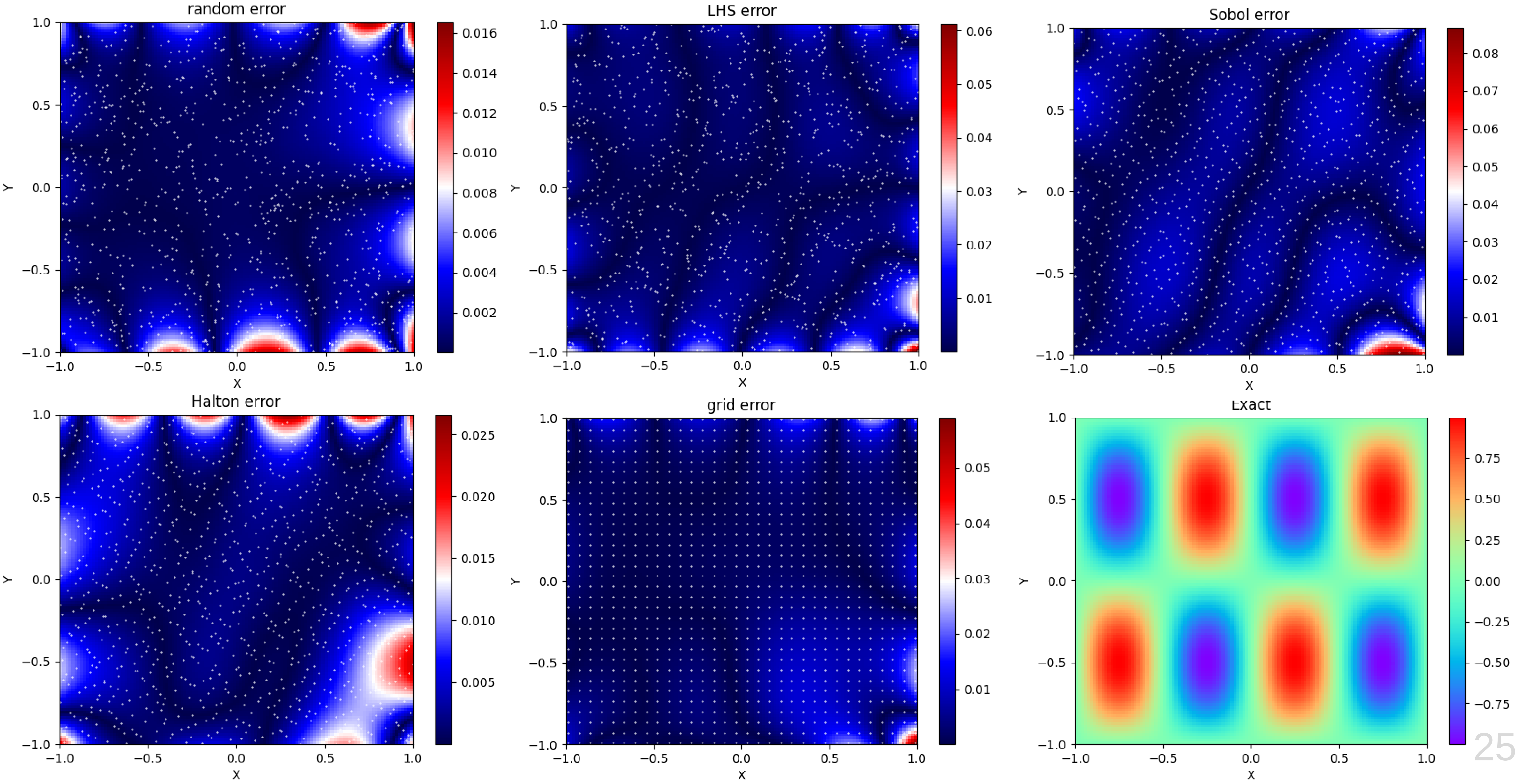
Basics

- ❑ batch [already seen, not covered here]
- ❑ number of points, how much can you offer

Sampling type



Non adaptive sampling



Residual based adaptive refinement with greed (RAR-G):

- ❑ Create sample of test points
- ❑ Determine PDE residual $\varepsilon(x) = |\mathcal{N}[\tilde{u}] - f|$
- ❑ Add points with largest residual to train sample

```
if self.iteration >= Start and (self.iteration - Start)%frequency == 0 and self.x.shape[0] < self.NCollocationMax:
    X_test, _     = self.sampler(Ntest) # Your hand crafted sampling method (grid, random uniform, ...)
    residu        = self.residual_PDE(X_test[:,0:1], X_test[:,1:2])
    _ , indices   = torch.topk(residu**2, self.GreedNum, dim=0, sorted=False)
    self.x        = torch.vstack ( (self.x, torch.take(X_test[:,0:1], indices).clone().detach().requires_grad_(True)) )
    self.y        = torch.vstack ( (self.y, torch.take(X_test[:,1:2], indices).clone().detach().requires_grad_(True)) )
    self.f_target = torch.zeros(self.x.shape).to(device)
```

Zhao, C. L. https://doi.org/10.48550/arXiv.2007.04542

Adaptive sampling

Residual based adaptive distribution (RAD):

- ❑ Create sample of test points
- ❑ Determine PDE residual $\varepsilon(x) = |\mathcal{N}[\tilde{u}] - f|$
- ❑ Determine probability distribution function according to PDE residual

$$p(x) = \frac{\varepsilon(x)}{A} \; with \; A = \int \varepsilon(x)dx$$

- ❑ Take new sample of train points randomly according to PDF

```
if self.iteration >= Start and (self.iteration - Start)%frequency == 0:
    X_test, _      = self.sampler(Ntest)  # Your hand crafted sampling method (grid, random uniform, ...)
    residu         = self.residual_PDE(X_test[:,0:1], X_test[:,1:2])
    norm           = torch.sum(torch.abs(residu))
    distribution = (torch.abs(residu) / norm ).flatten()
    indices        = torch.multinomial(distribution, self.x.shape[0])
    self.x         = torch.take(X_test[:,0:1], indices)[:,None]
    self.y         = torch.take(X_test[:,1:2], indices)[:,None]
    '# create new leaf variable to move grad_fn from UnsqueezeBackward0 to ToCopyBackward0'
    self.x         = self.x.clone().detach().requires_grad_(True)
    self.y         = self.y.clone().detach().requires_grad_(True)
```

M. Amin & al, https://doi.org/10.1111/mice.12685

Adaptive sampling

Residual based adaptive distribution with greed (RAD-G):

- ❑ Create probability density function as in RAD
- ❑ Add points taken randomly according to PDF to train sample

```
if self.iteration >= Start and (self.iteration - Start)%frequency == 0 and self.x.shape[0] < self.NCollocationMax:
    X_test, _       = self.sampler(Ntest)  # Your hand crafted sampling method (grid, random uniform, ...)
    residu          = self.residual_PDE(X_test[:,0:1], X_test[:,1:2])
    norm            = torch.sum(torch.abs(residu))
    distribution    = (torch.abs(residu) / norm ).flatten()
    indices         = torch.multinomial(distribution, self.GreedNum)
    x_new           = torch.take(X_test[:,0:1], indices)[:,None]
    y_new           = torch.take(X_test[:,1:2], indices)[:,None]
    x_new           = x_new.clone().detach().requires_grad_(True)
    y_new           = y_new.clone().detach().requires_grad_(True)
    self.x          = torch.vstack ( (self.x, x_new) )
    self.y          = torch.vstack ( (self.y, y_new) )
    self.f_target = torch.zeros(self.x.shape).to(device)
```

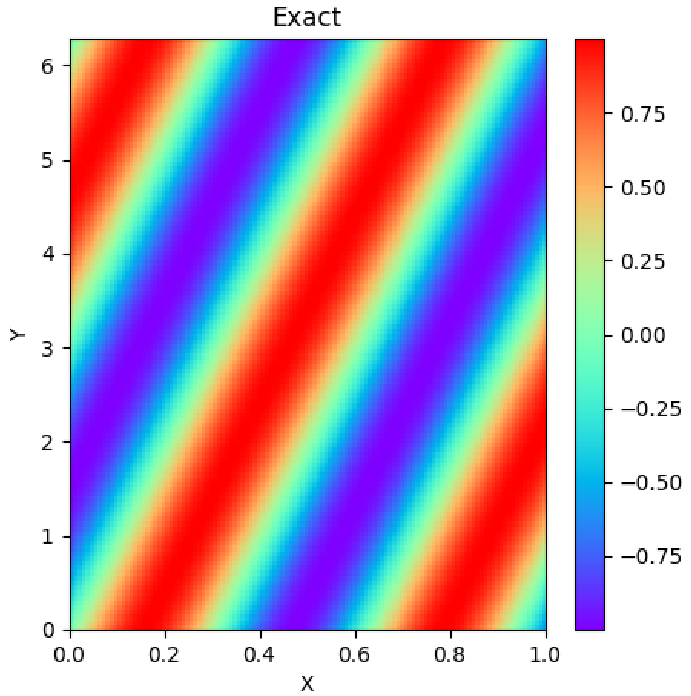
C. Wu et al, https://doi.org/10.1016/j.cma.2022.115671

Adaptive sampling

$$\frac{\partial u}{\partial t} - c\frac{\partial u}{\partial x} = 0, \quad x \in [0,2\pi], \quad t \in [0,1]$$

$$u(x,0) = \sin(x), \quad x \in [0,2\pi]$$

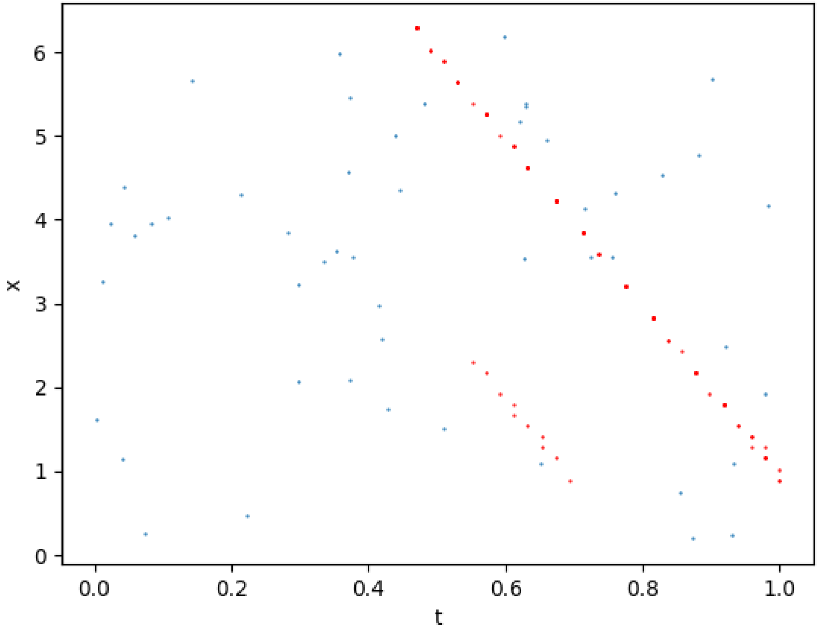
$$u(0,t) = u(2\pi,\mathrm{t}), \qquad t > 0$$

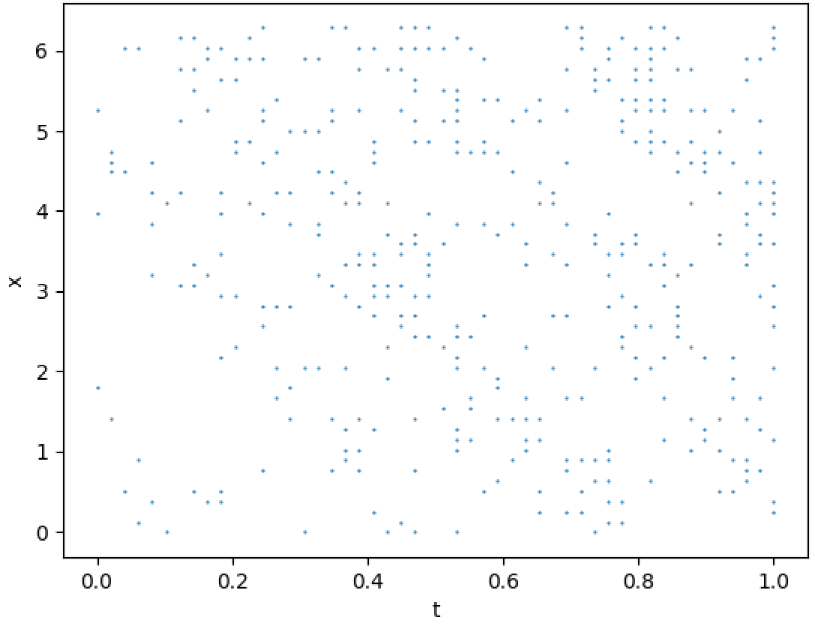


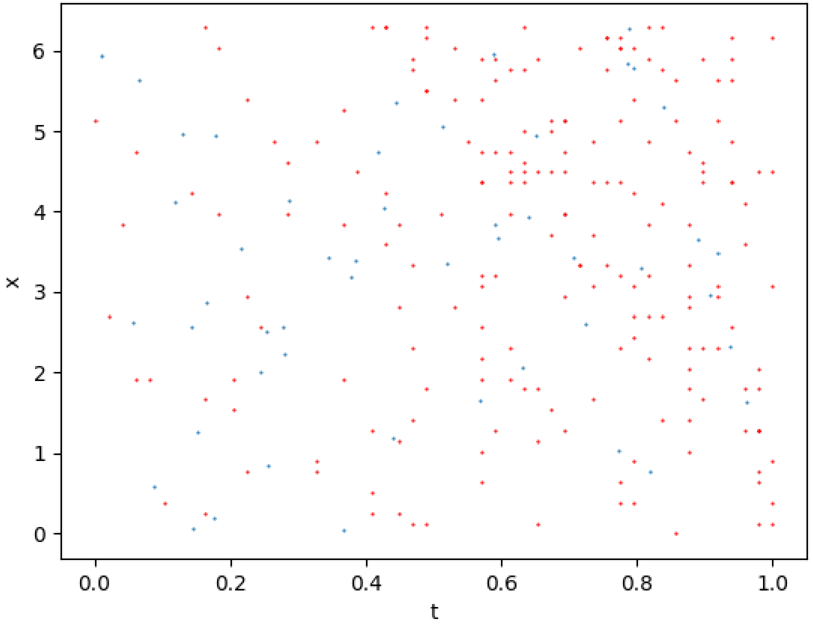
$c = 10$

RAR-G **RAD** **RAD-G**

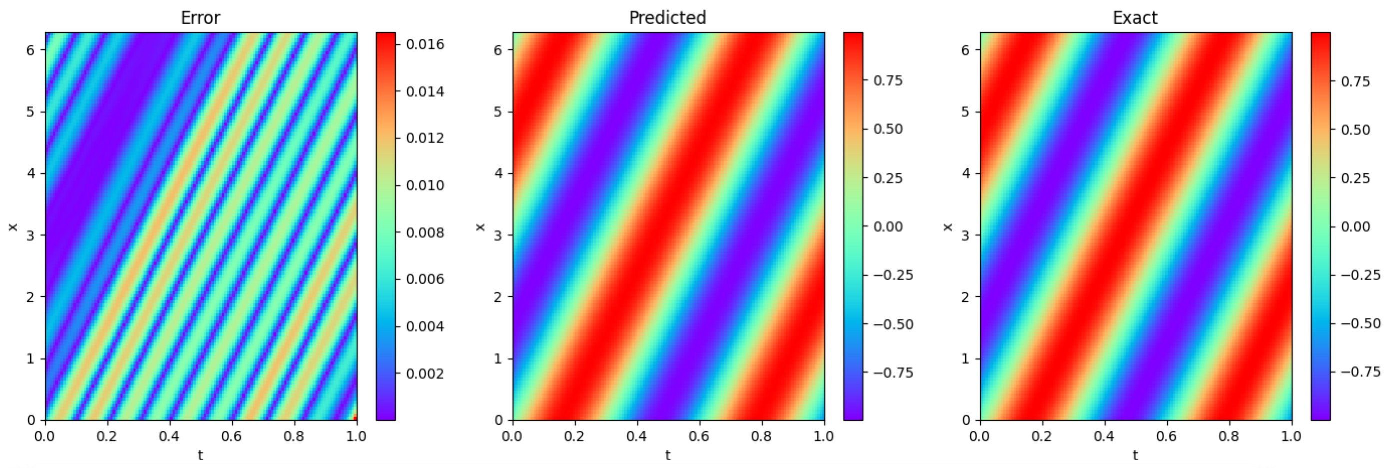
Iteration = 500



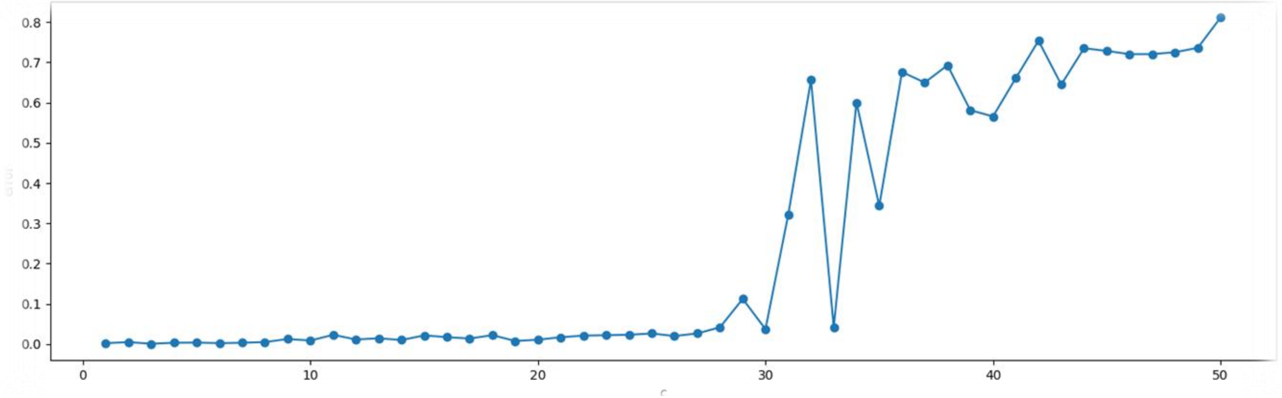


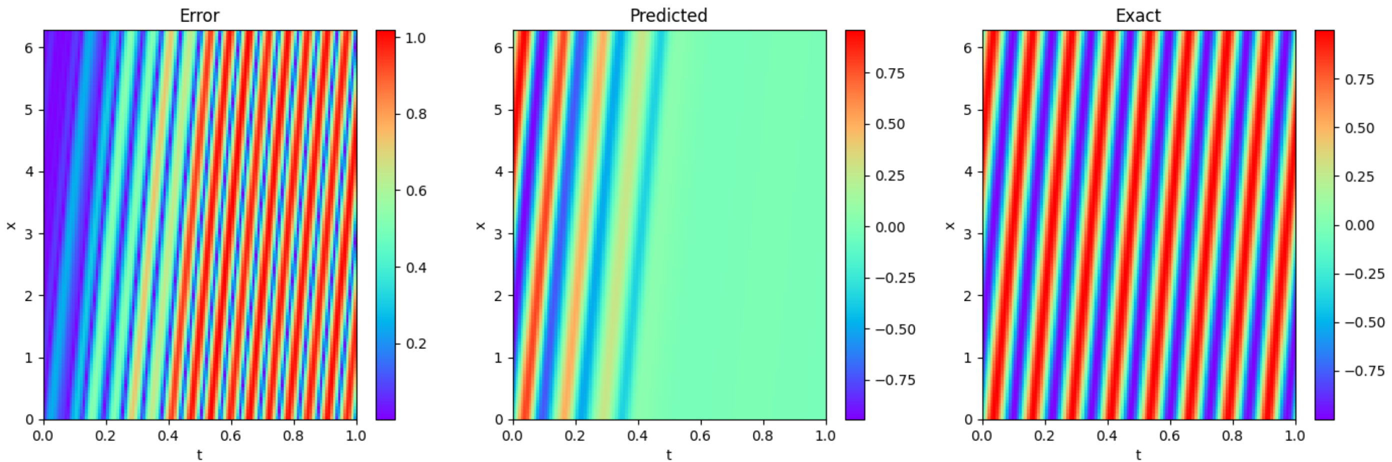


Common troubles



$c = 10$





$c = 50$

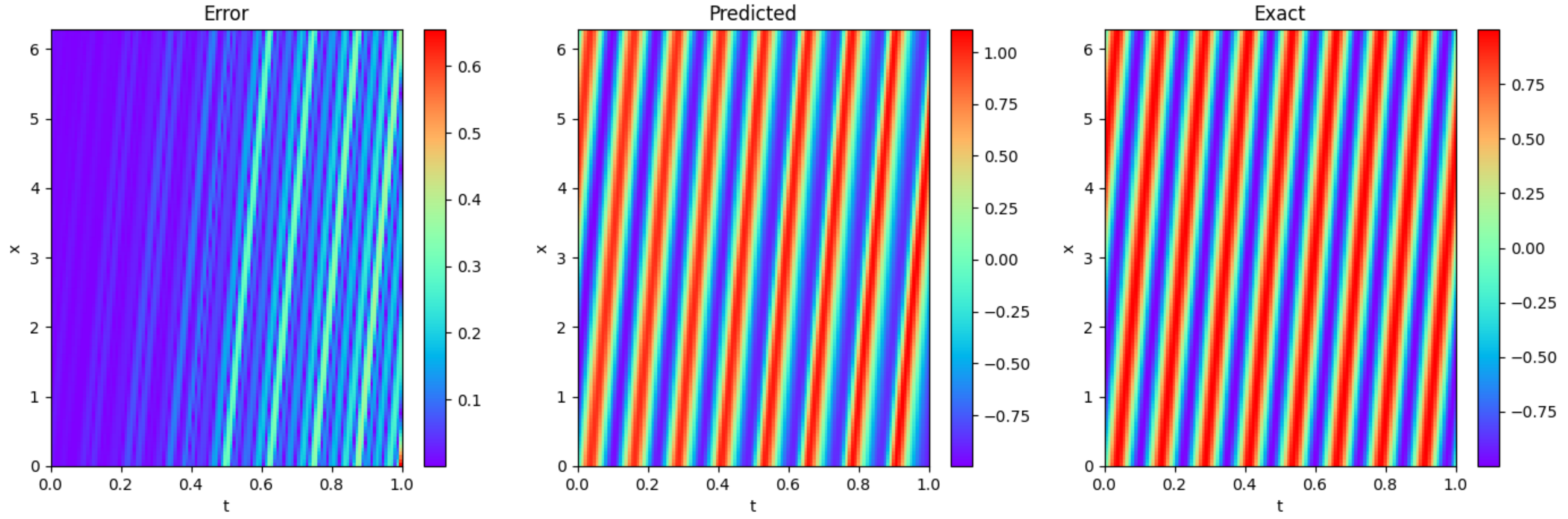
Evolutionary sampling (EVO):

- ❑ Take absolute value of residu for each collocation points
- ❑ Keep collocation points for which the residu > mean residual value
- ❑ Resample randomly the other collocation points

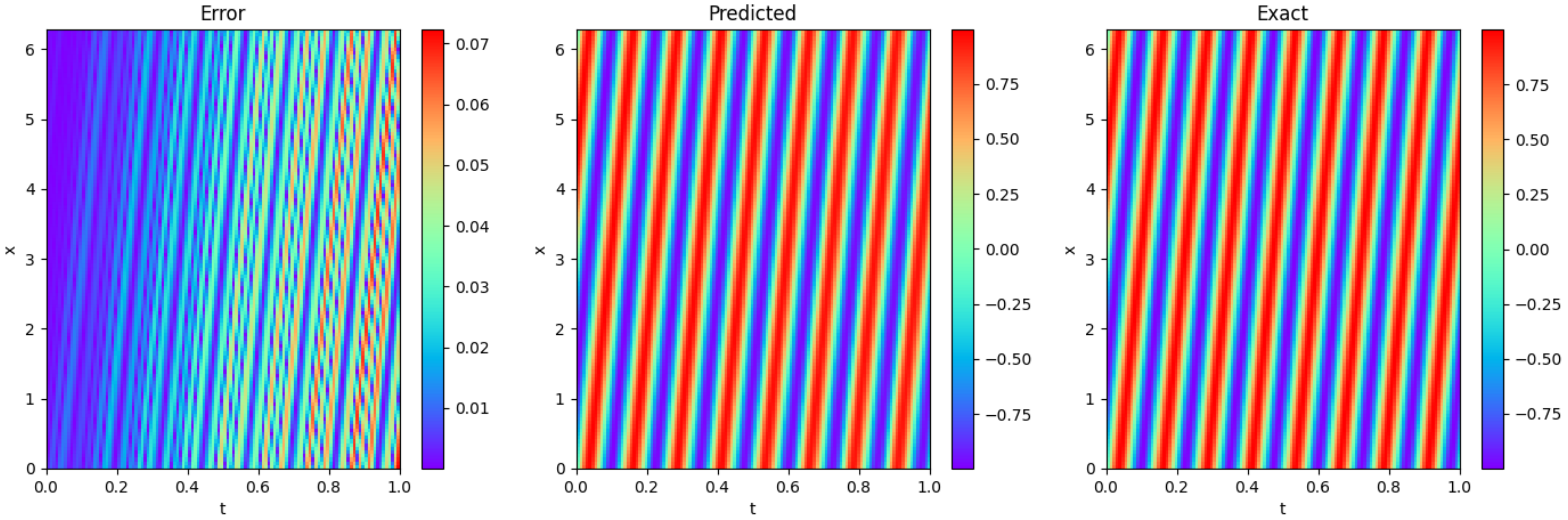
```
residu    = torch.abs(self.residual_PDE(self.x, self.t)).to(device)
threshold = torch.mean(residu)
indices   = torch.stack(list((torch.nonzero((residu >= threshold)*1, as_tuple=True)[0])), dim=0)
x_evo     = torch.take(self.x, indices)[:,None]
t_evo     = torch.take(self.x, indices)[:,None]
x_rand    = (self.lb + (self.ub - self.lb) * torch.rand(self.x.shape[0]-x_evo.shape[0]).float()[:,None]).to(device)
t_rand    = (torch.rand(self.t.shape[0] - t_evo.shape[0]).float()[:,None]).to(device)
x_evo     = torch.vstack (( x_evo, x_rand))
t_evo     = torch.vstack (( t_evo, t_rand))
self.x    = x_evo.clone().detach().requires_grad_(True)
self.t    = t_evo.clone().detach().requires_grad_(True)
```

A. Daw et al https://doi.org/10.48550/arXiv.2207.02338

classic



EVO

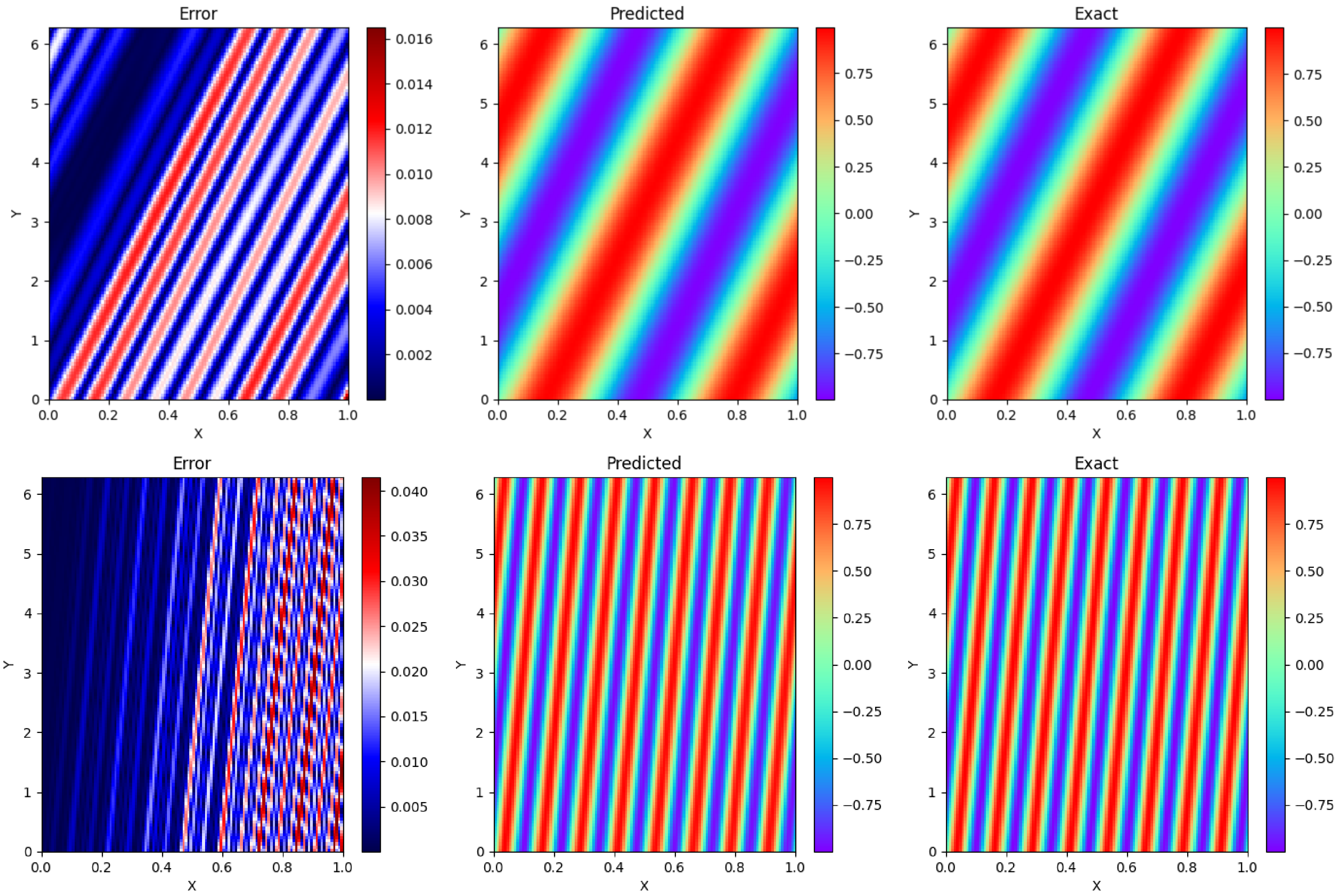


Curriculum learning

Training of NN from *c=10* to *c=50*

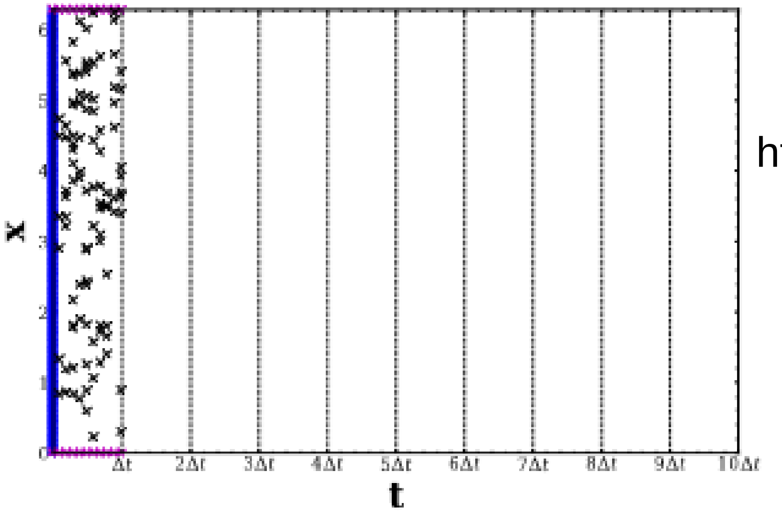
$c = 10$

$c = 50$



Many others tricks

Sequence to sequence



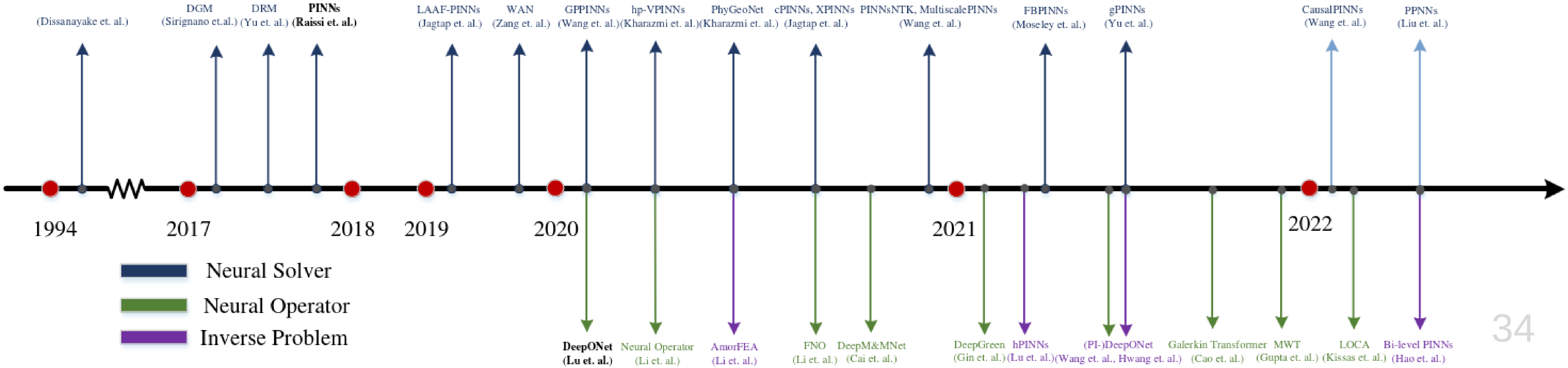
https://arxiv.org/pdf/2109.01050.pdf

Gradient enhanced

$$\mathcal{L}_{gPINN} = \mathcal{L}_{PINN} + \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\partial r}{\partial x_i}\right|^2$$

https://doi.org/10.1016/j.cma.2022.114823

And so much more… (https://doi.org/10.48550/arXiv.2211.08064)



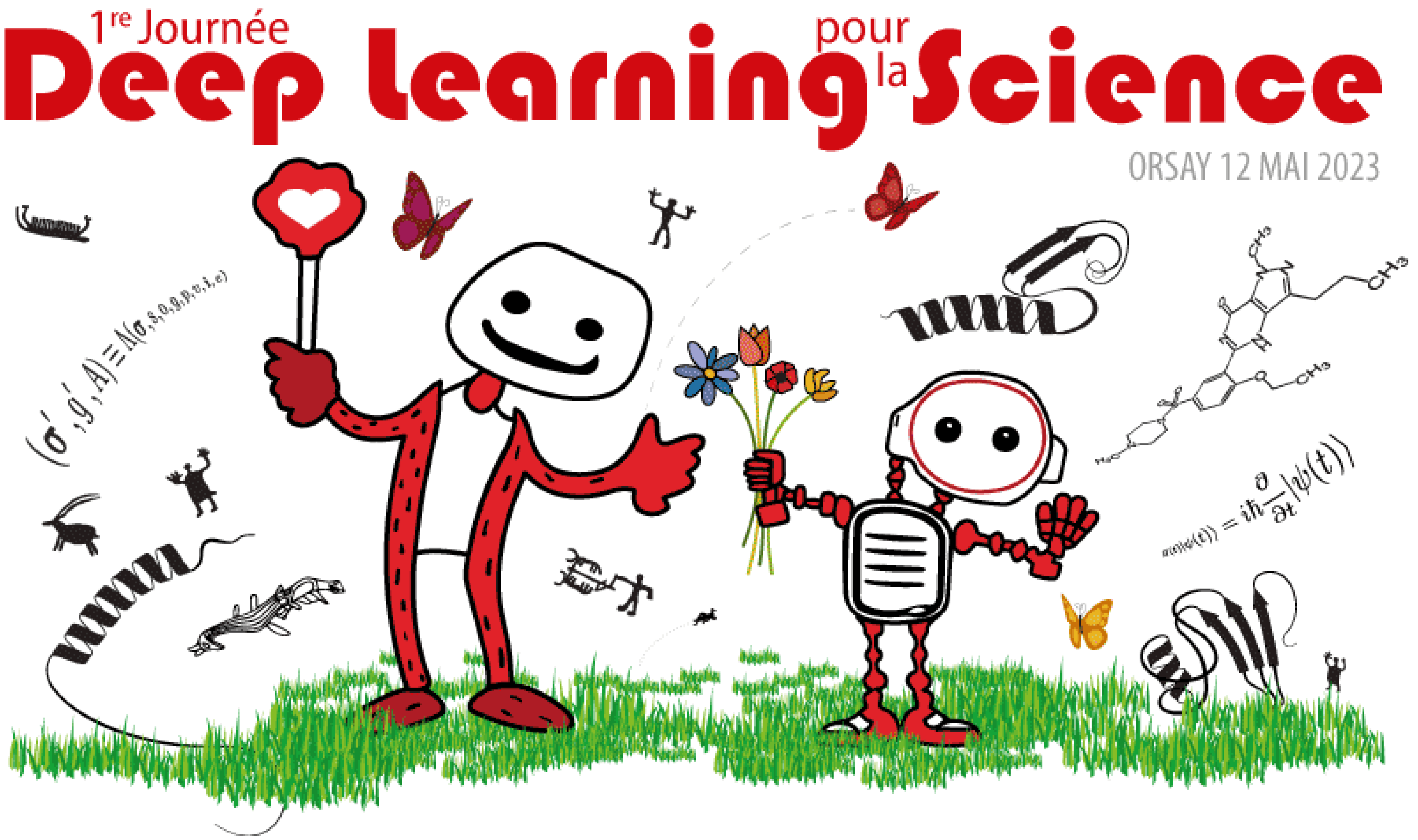
Architecture of NN
- ❑ MLP (scattered sensor measurements)
- ❑ RNN (discrete time series)
- ❑ CNN (gridded 2D domains, images)
- ❑ GNN, PointNet (Unstructured grids and irregular geometries)

Self adaptive activation functions (https://doi.org/10.1016/j.jcp.2019.109136)

Learning operators
- ❑ DeepONet (https://doi.org/10.1038/s42256-021-00302-5)
- ❑ FourierNeuralOperator (https://doi.org/10.48550/arXiv.2010.08895)

Questions Break

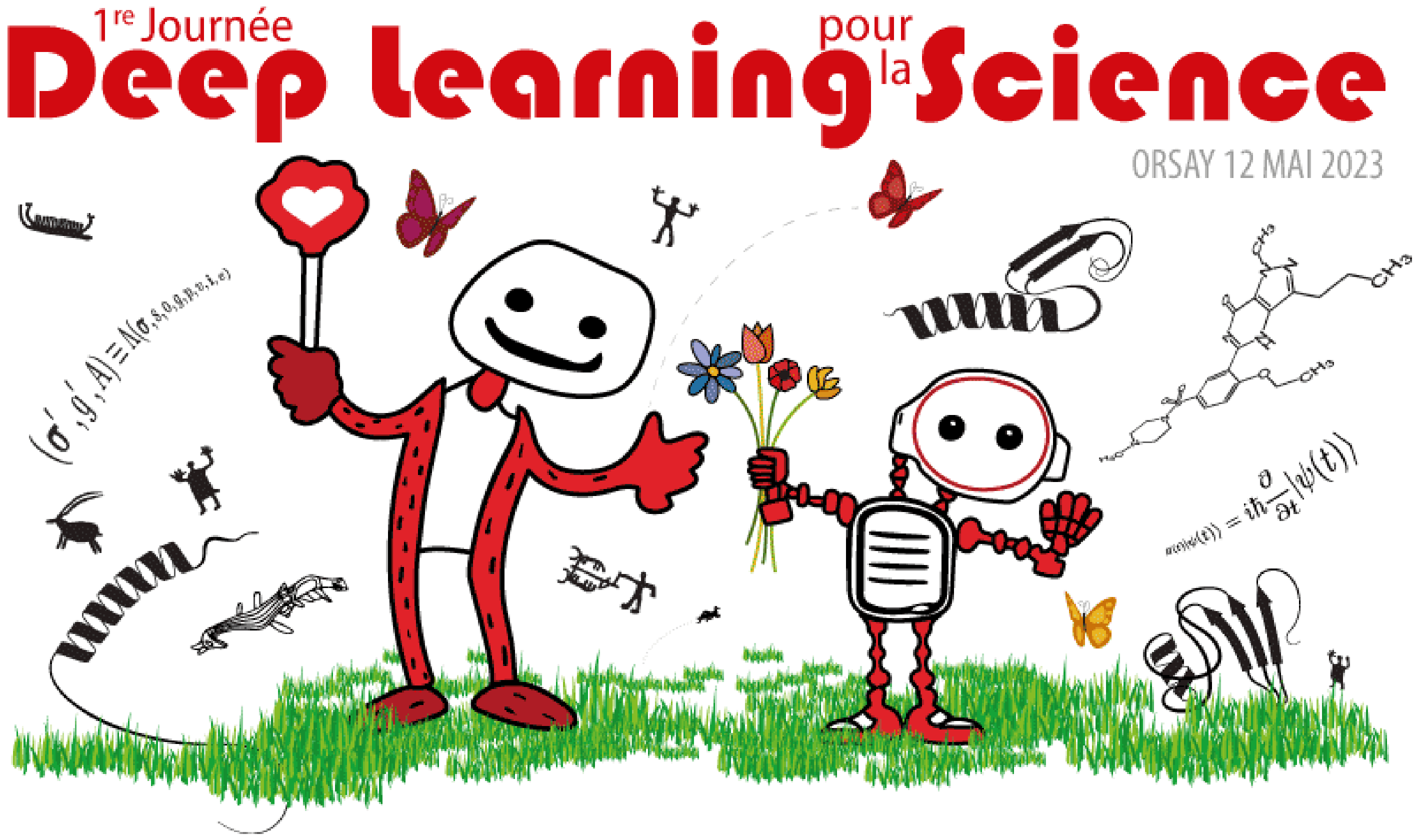
Next, on Fidle :



http://www.idris.fr/annonces/jdls2023.html

Rediffusion en live sur : https://www.youtube.com/@CNRS-FIDLE

Next, on Fidle :



http://www.idris.fr/annonces/jdls2023.html

Rediffusion en live sur : https://www.youtube.com/@CNRS-FIDLE

To be continued...

contact@fidle.cnrs.fr
FIDLE https://fidle.cnrs.fr
YouTube https://fidle.cnrs.fr/youtube

FIDLE