KU LEUVEN

LMSD Mecha(tro)nic System Dynamics

FLANDERS MAKE DRIVING INNOVATION IN MANUFACTURING

GREYDIENT

Confluences Mathématiques

21 November 2024, Paris, France

A practical introduction to Physics-Informed Neural Networks (PINNs)

Damien Bonnet-Eymard

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 955393.

RELIABLE & ROBUST DESIGN

About me

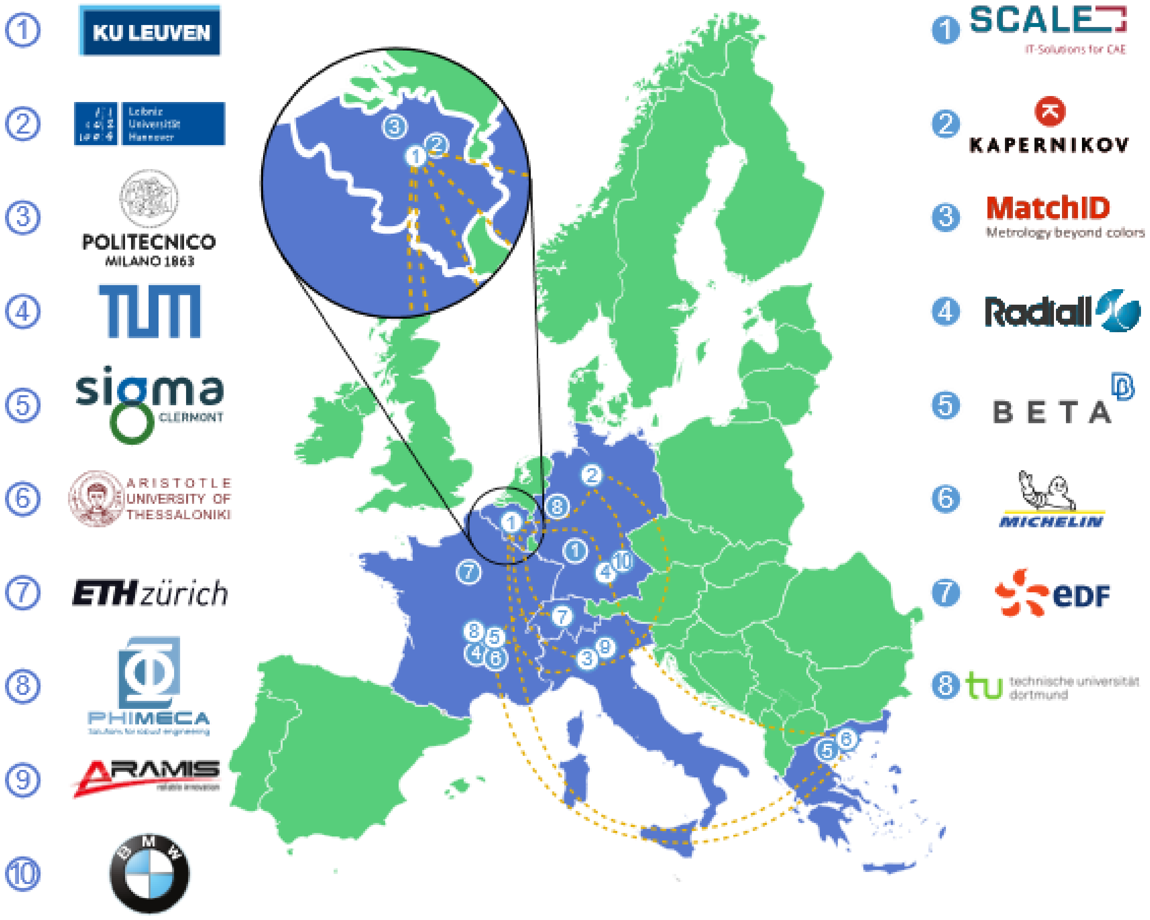
Damien Bonnet-Eymard
PhD student @ KU Leuven
ESR @Greydient

damien.bonnet-eymard@kuleuven.be

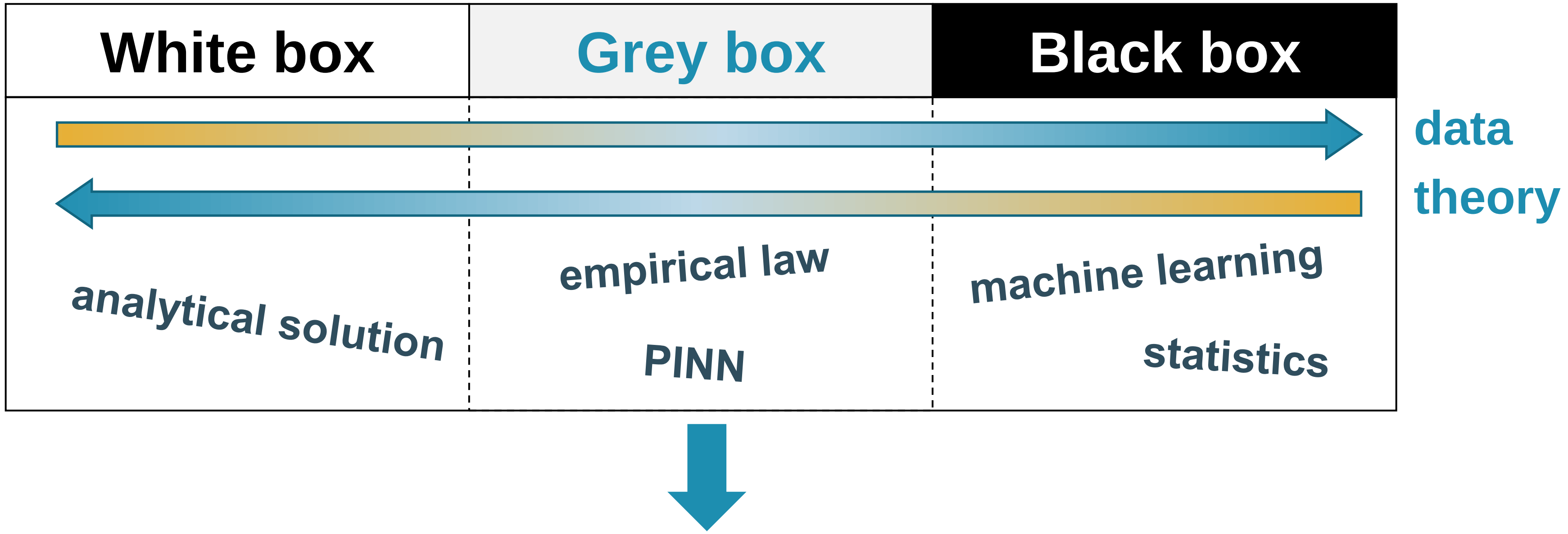
Physics-Informed neural networks for continuum mechanics applications

GREYDIENT

Marie Curie ITN project founded by the European commission



Greydient project : developing the grey-box methodology



most practical application

some physics – some data

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

Brief history of **P**hysics-**I**nformed **N**eural **N**etworks

2017

Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations
Maziar Raissi, Paris Perdikaris, George Em Karniadakis

2019

Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations
Maziar Raissi, Paris Perdikaris, George Em Karniadakis

11621+ citations

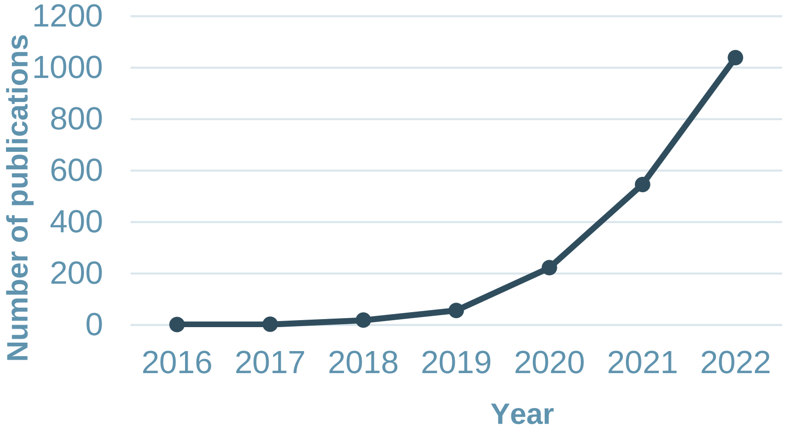


Fig.1 : Publication with title/abstract containing "Physics-Informed Neural Networks" on Dimensions (www.dimensions.ai)

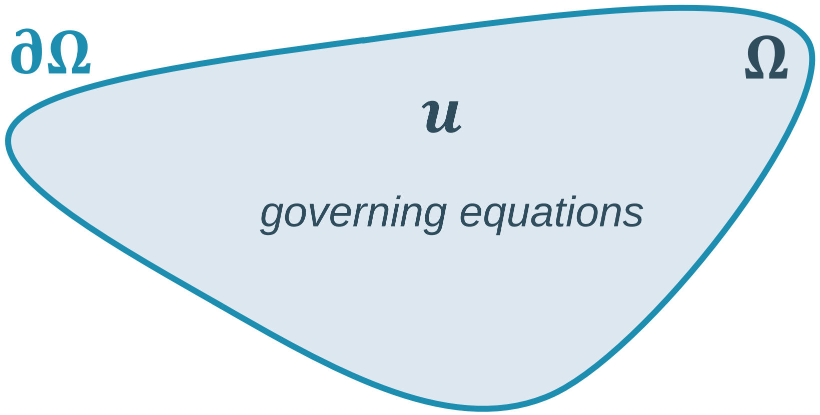
Why **PINNs** are so popular ?

- good at extrapolation/inverse problem
- benefits from late AI research
- easily applicable to any topic

⚠ *Convergence issues*

Motivation: solving a boundary value problem

Boundary **V**alue **P**roblem (**BVP**):



$$\mathbf{BC} : \partial\Omega : \ f_{BC}(u) = 0$$
$$\mathbf{IC} : u(t = 0) = u_0$$

Theory :

If well posed : *(BC well defined)*

➡ **Existence** and **uniqueness** of u[1]

The Ritz (Galerkin) method :

Discretization for numerical resolution

➡ **F**inite **E**lement **M**ethod

➡ **P**hysics-**I**nformed **N**eural **N**etworks

[1] « Picard–Lindelöf Theorem ». 2022. In *Wikipedia*.

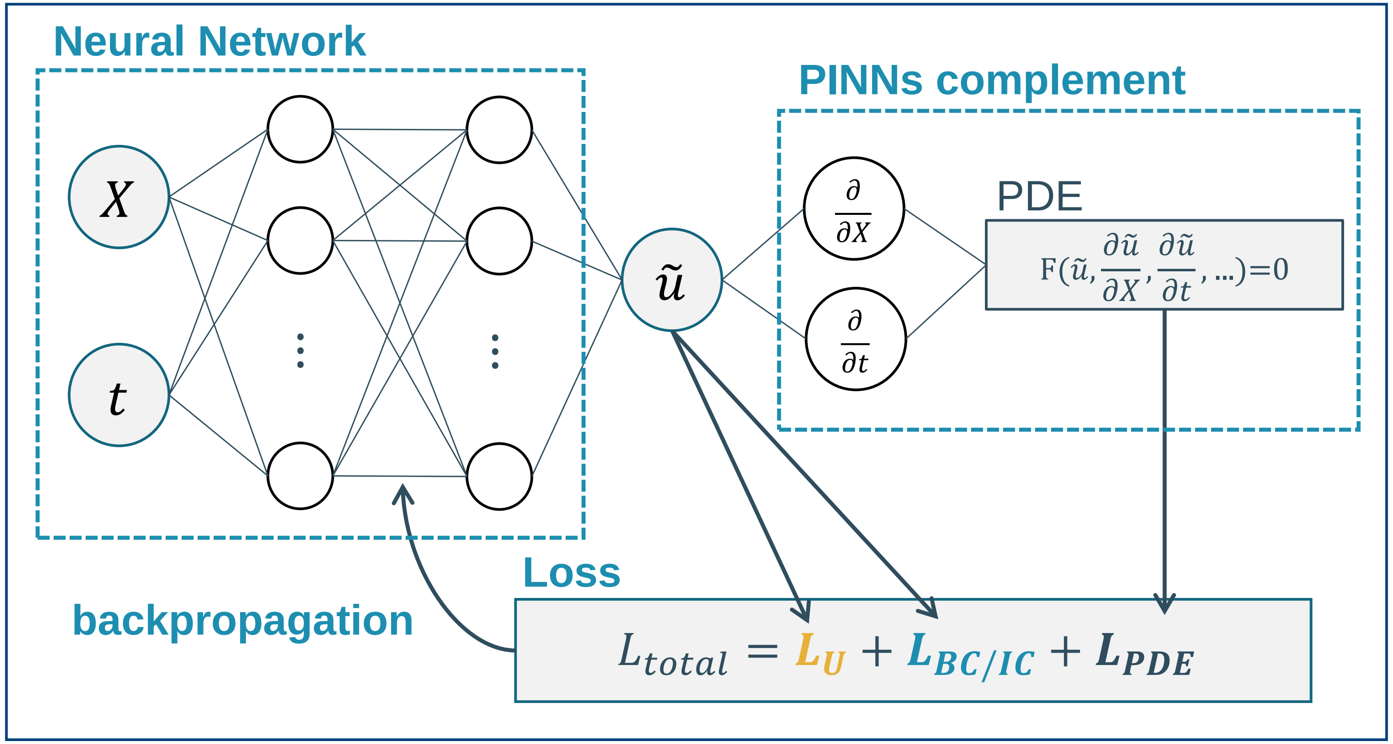
Comparing PINN and FEM : choosing the trial function space

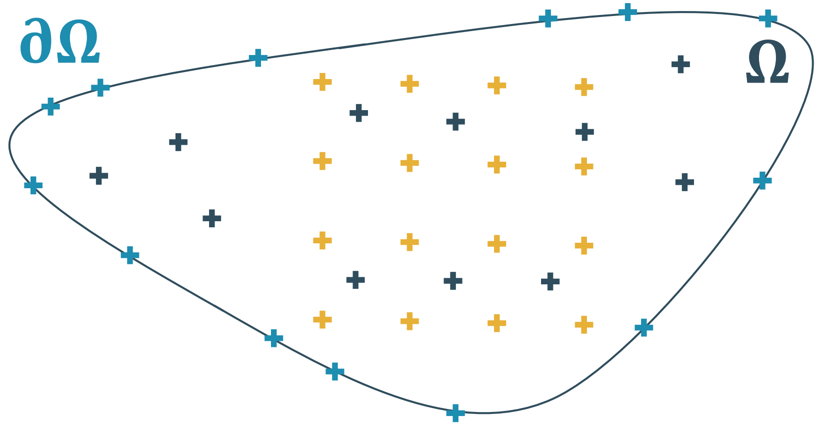
		Finite Element Method	Physics-Informed Neural Network
	Discretization	Mesh	Neural network architecture
	Trial/basis function	Piece-wise polynomials	Artificial neural network
	Parameters	Mesh nodal values	Network weight and biases
	Resolution	Matrix inversion	Stochastic optimization
	Hyper-parameter	Mesh (geometry, element)	Network, optimizer, implementation
Pros/ Cons	**Solution***	Unique	Non-unique (optimization and generalization error)
Pros/ Cons	**Boundary conditions**	All are needed (inversible matrix)	Can be missing
Pros/ Cons	**Incorporating measurement**	Can be expensive (need iterative updating)	Seamless during training (adding a residual loss term)

Hyper-parameter are crucial for the **convergence**

*of a well-posed problem for a given mesh/network

PINNs for a boundary value problem

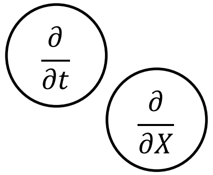




PDE evaluation : $L_{PDE} = \sum_{X \in \Omega} \left\| F(\tilde{u}, \frac{\partial \tilde{u}}{\partial X}, \frac{\partial \tilde{u}}{\partial t}, \ldots) \right\|$

BC value : $L_{BC/IC} = \sum_{X \in \partial\Omega} \| N(X) - u_{BC}(X) \|$

Ground truth data : $L_U = \sum_{X \in \Omega} \| N(X) - u \|$



performed using **A**utomatic **D**ifferentiation

Applying chain rule throw the network

PINNs for a boundary value problem

Loss

$$L_{total} = L_U + L_{BC/IC} + L_{PDE}$$

Imposing boundary condition :

SOFT

Adding a loss term :

$$L_{BC/IC} = \sum_{X \in \partial\Omega} \|N(X) - u_{BC}(X)\|$$

penalize non-respect of boundary conditions

HARD

Applying a mask function on the output :

$$u = F_{mask}[N(X)]$$

directly enforce boundary conditions

PINNs for a boundary value problem

Loss

$$L_{total} = L_U + \cancel{L_{BC/IC}} + L_{PDE}$$

Imposing boundary condition :

SOFT

Adding a loss term :

$$L_{BC/IC} = \sum_{X \in \partial\Omega} \|N(X) - u_{BC}(X)\|$$

penalize non-respect of boundary conditions

HARD

Applying a mask function on the output :

$$u = F_{mask}[N(X)]$$

directly enforce boundary conditions

Hard BC example :

$u_{x=0} = u_{x=1} = 0$

$u_{y=0} = u_{y=1} = \cos(2\pi x)$

NN output

Mask function

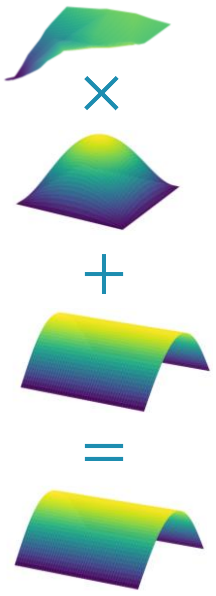
$= 0 \; on \; \partial\Omega_{BC}$

$\neq 0 \; elswhere$

BC function

$= U_{BC} \; on \; \partial\Omega_{BC}$

BC compliant output



PINNs for a boundary value problem

Loss

$$L_{total} = L_U + L_{BC/IC} + L_{PDE}$$

Imposing boundary condition :

SOFT

Adding a loss term :

$$L_{BC/IC} = \sum_{X \in \partial\Omega} \| N(X) - u_{BC}(X) \|$$

penalize non-respect of boundary conditions

HARD

Applying a mask function on the output :

$$u = F_{mask}[N(X)]$$

directly enforce boundary conditions

Relaxed constraint	Exact imposition
General and seamless to implement	Specific to every problem (generalization possible)
Multi-term optimization (make convergence harder)	Better convergence

Hard BC example :

$u_{x=0} = u_{x=1} = 0$

$u_{y=0} = u_{y=1} = \cos(2\pi x)$

NN output

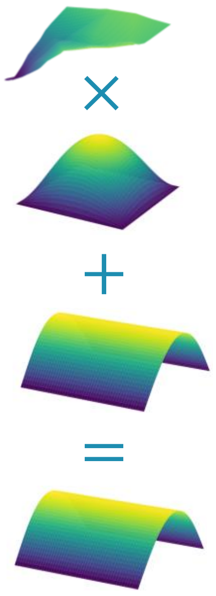
Mask function

$= 0 \; on \; \partial\Omega_{BC}$
$\neq 0 \; elswhere$

BC function

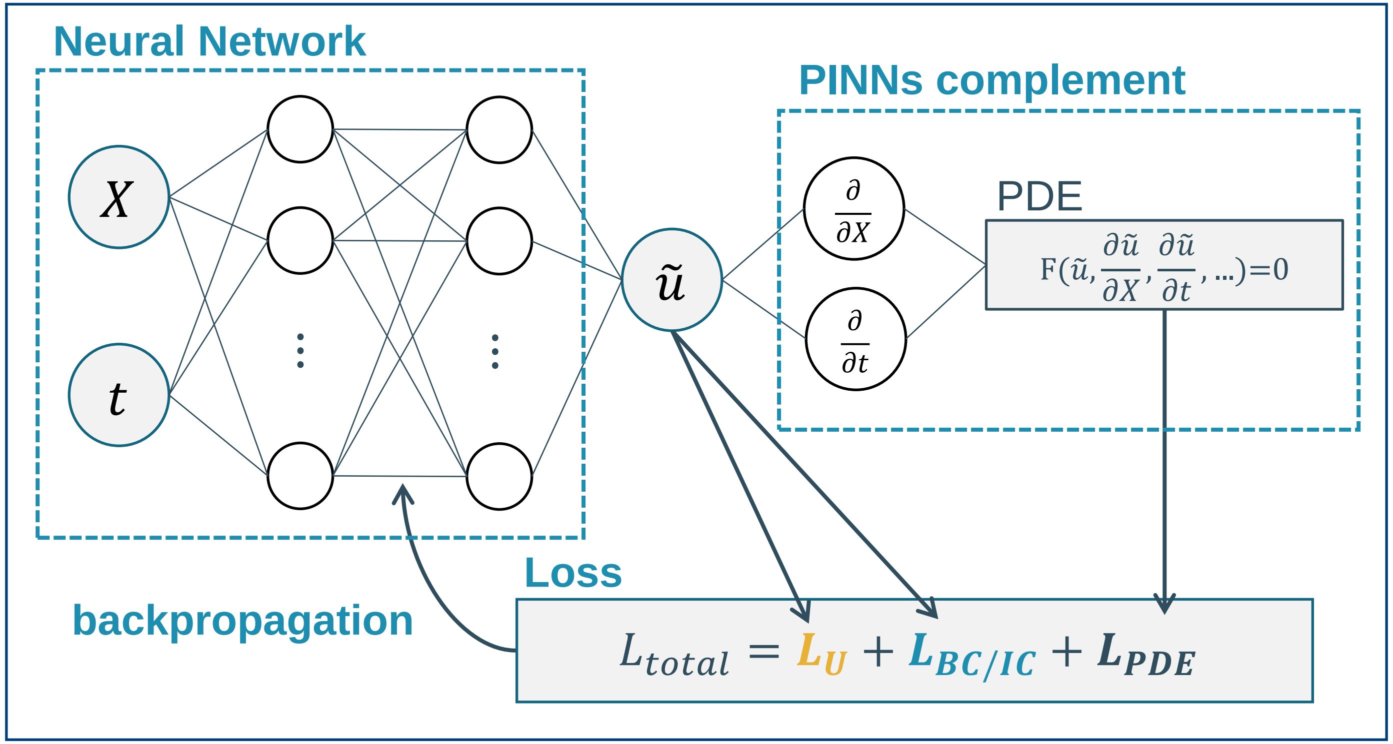
$= U_{BC} \; on \; \partial\Omega_{BC}$

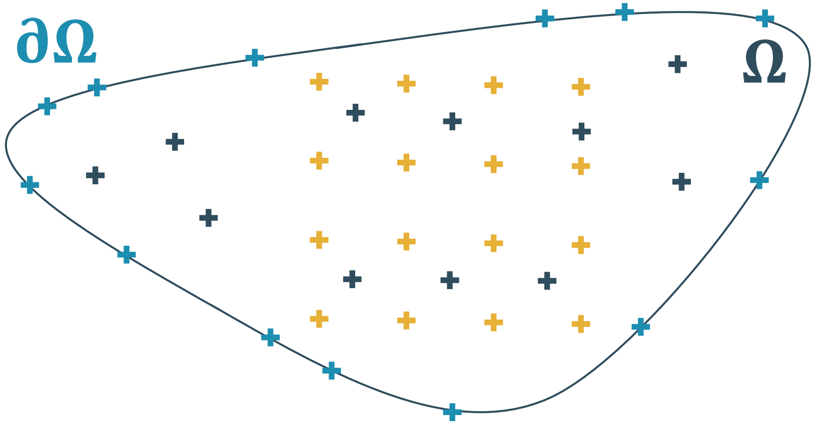
BC compliant output



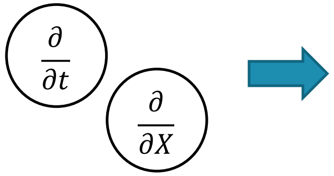
KU LEUVEN

PINNs for a boundary value problem





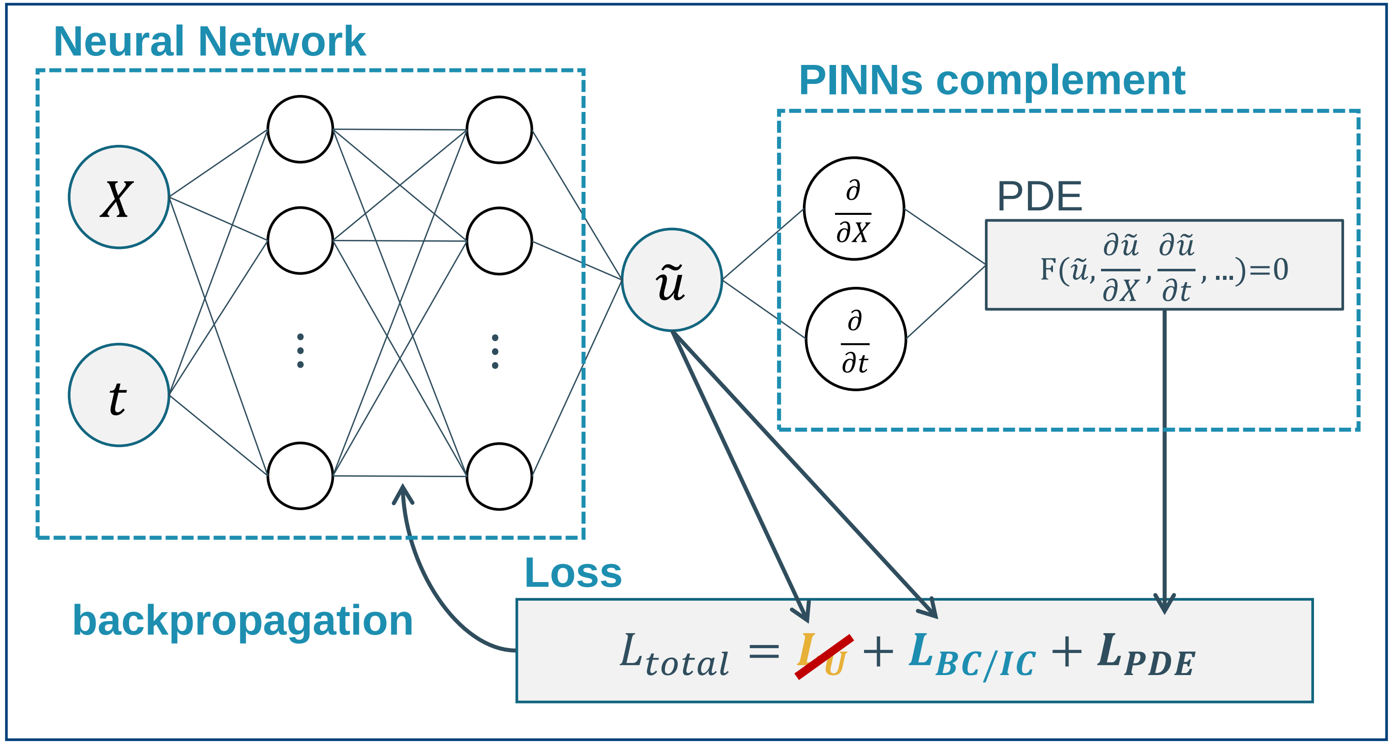
- *PDE evaluation :* $L_{PDE} = \sum_{X \in \Omega} \left\| F(\tilde{u}, \frac{\partial \tilde{u}}{\partial X}, \frac{\partial \tilde{u}}{\partial t}, \dots) \right\|$
- *BC value :* $L_{BC/IC} = \sum_{X \in \partial\Omega} \| N(X) - u_{BC}(X) \|$
- *Ground truth data :* $L_U = \sum_{X \in \Omega} \| N(X) - u \|$



performed using **A**utomatic **D**ifferentiation

Applying chain rule throw the network

PINNs for a boundary value problem



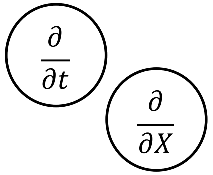
$$L_{PDE} = \sum_{X \in \Omega} \left\| F\left(\tilde{u}, \frac{\partial \tilde{u}}{\partial X}, \frac{\partial \tilde{u}}{\partial t}, \ldots\right) \right\|$$

$$L_{BC/IC} = \sum_{X \in \partial\Omega} \| N(X) - u_{BC}(X) \|$$

~~$$L_U = \sum_{X \in \Omega} \| N(X) - u \|$$~~

Forward problem :

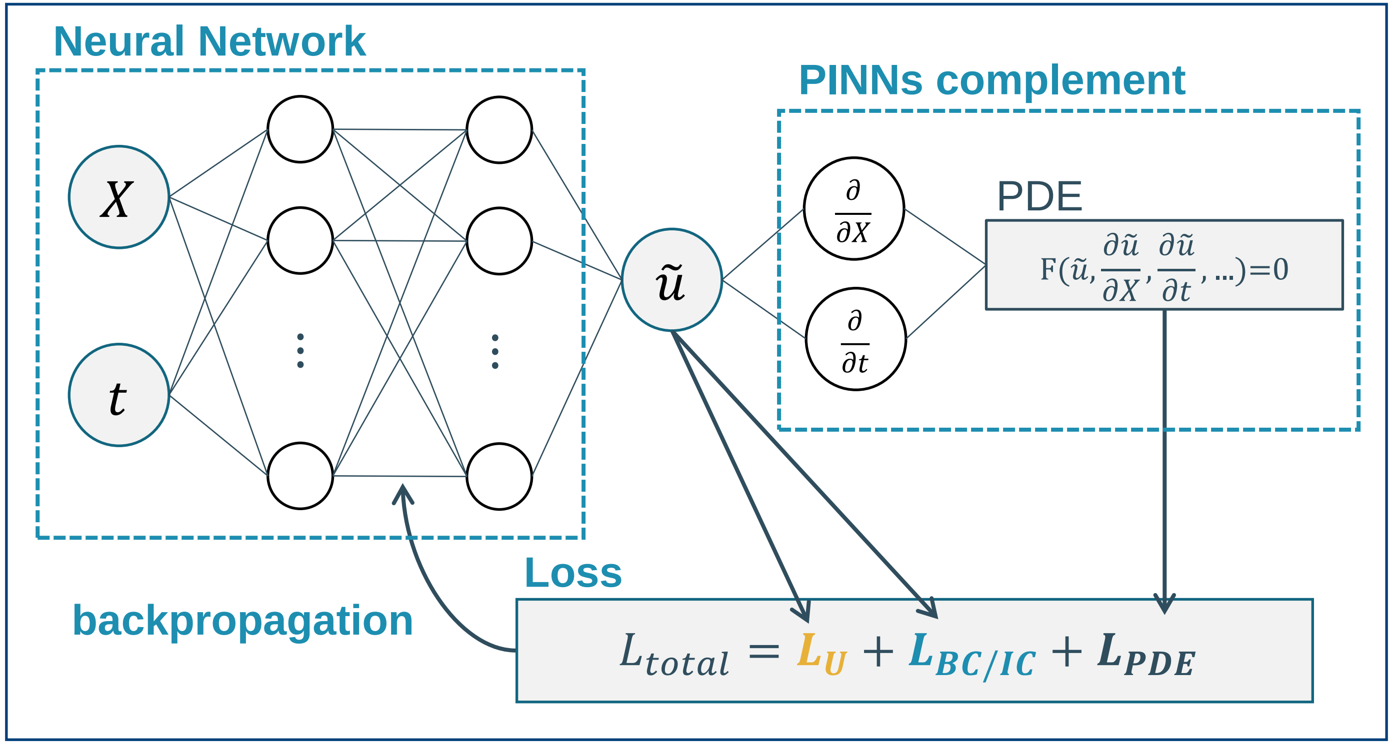
➡ *no need for labeled data*



➡ performed using **A**utomatic **D**ifferentiation

Applying chain rule throw the network

PINNs for a boundary value problem



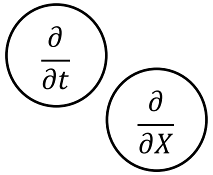
$$L_{PDE} = \sum_{X \in \Omega} \left\| F(\tilde{u}, \frac{\partial \tilde{u}}{\partial X}, \frac{\partial \tilde{u}}{\partial t}, \ldots) \right\|$$

$$L_{BC/IC} = \sum_{X \in \partial\Omega} \| N(X) - u_{BC}(X) \|$$

$$L_U = \sum_{X \in \Omega} \| N(X) - u \|$$

Forward problem :

➡ *no need for labeled data*



➡ performed using **A**utomatic **D**ifferentiation

Applying chain rule throw the network

Inverse problem :

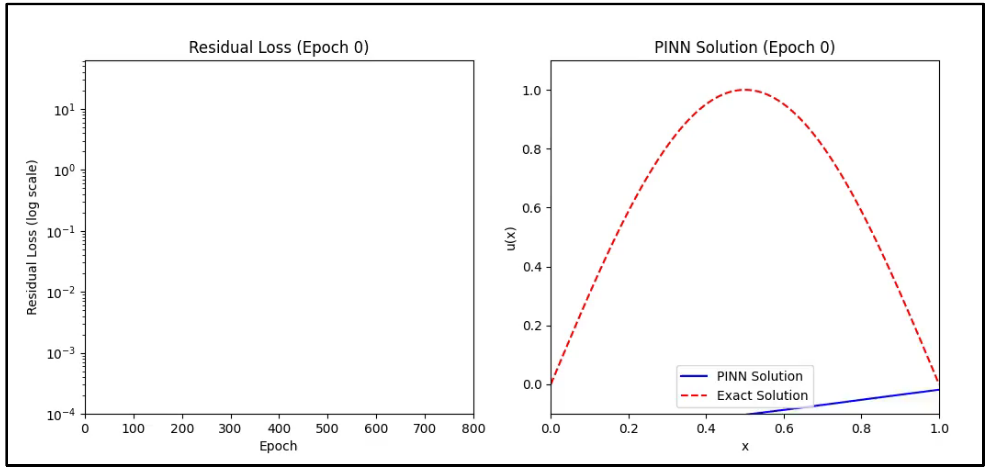
Determining PDE parameters

$F_{p_i}(\tilde{u}, \frac{\partial \tilde{u}}{\partial X}, \frac{\partial \tilde{u}}{\partial t}, \ldots)$ p_i : *model parameters*

PINN to solve 1D Poisson equation

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} = \pi^2 \sin(\pi x), \quad x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

Exact solution : $u(x) = \sin(\pi x)$



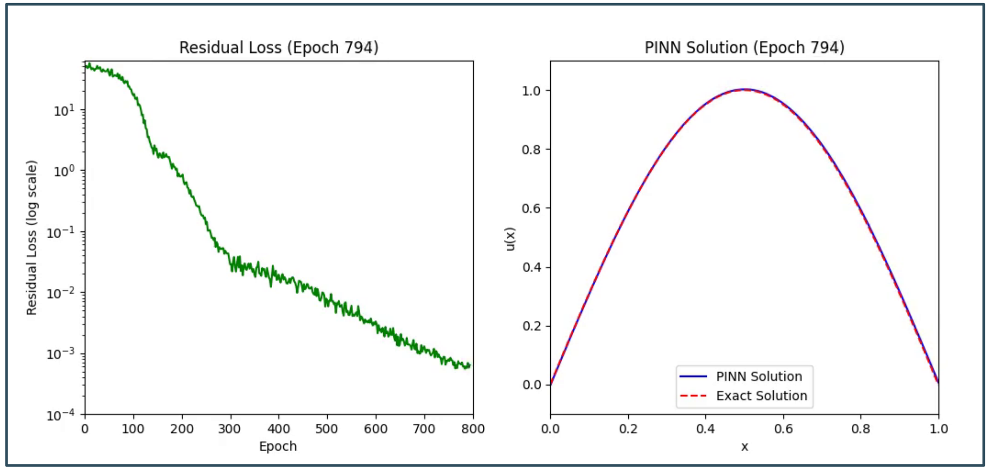
PyTorch

```python
import torch
import torch.nn as nn
import torch.optim as optim

class PINN(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(1, 20), nn.Tanh(),
            nn.Linear(20, 20), nn.Tanh(),
            nn.Linear(20, 1)
        )

    def forward(self, x):
        return self.net(x)

def poisson_residual(x, model):
    u = model(x)
    u_x = torch.autograd.grad(u, x, torch.ones_like(u), create_graph=True)[0]
    u_xx = torch.autograd.grad(u_x, x, torch.ones_like(u_x), create_graph=True)[0]
    f = torch.pi**2*torch.sin(torch.pi * x)
    return (-u_xx - f).pow(2).mean()  # Residual loss

def boundary_loss(model):
    return model(torch.tensor([[0.0]]))**2 + model(torch.tensor([[1.0]]))**2  # Enforce u(0)=u(1)=0

# Initialize model and optimizer
model = PINN()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
for epoch in range(1000):
    x = torch.rand(100, 1, requires_grad=True)
    loss = poisson_residual(x, model) + boundary_loss(model)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if epoch % 100 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item()}")
```

PINN to solve 1D Poisson equation

$$\begin{cases} -\dfrac{\partial^2 u}{\partial x^2} = \pi^2 \sin(\pi x), \quad x \in [0,1] \\ u(0) = u(1) = 0 \end{cases}$$

Exact solution : $u(x) = \sin(\pi x)$



DeepXDE

```python
import deepxde as dde
import torch

def pde(x, y):
    dy_xx = dde.grad.hessian(y, x)
    return -dy_xx - np.pi ** 2 * torch.sin(np.pi * x)

def boundary(x, on_boundary):
    return on_boundary

def func(x):
    return np.sin(np.pi * x)

geom = dde.geometry.Interval(-1, 1)
bc = dde.icbc.DirichletBC(geom, func, boundary)
data = dde.data.PDE(geom, pde, bc, 16, 2, solution=func, num_test=100)

layer_size = [1] + [50] * 3 + [1]
activation = "tanh"
initializer = "Glorot uniform"
net = dde.nn.FNN(layer_size, activation, initializer)

model = dde.Model(data, net)
model.compile("adam", lr=0.001, metrics=["l2 relative error"])

losshistory, train_state = model.train(iterations=10000)
```

PINN software

Python

- ***DeepXDE*[1]**
- *SciANN*
- *NeuroDiffEq*
- *IDRLnet*
- …

Julia

- *NeuralPDE.jl*

Nvidia

- *Modulus*

DeepXDE

A library for scientific machine learning and physics-informed learning

- ➢ Multi-backend : Tensorflow, Pytorch, JAX…
- ➢ Simplified implementation, lot of features
- ➢ Very active community, latest research implemented
- ➢ Well documented with a lot of examples

[1] Lu, Lu, Xuhui Meng, Zhiping Mao, et George E. Karniadakis. 2021. « DeepXDE: A deep learning library for solving differential equations ». *SIAM Review* 63 (1): 208-28. https://doi.org/10.1137/19M1274067

Literature review of PINN

Review papers

- **2021** Physics-informed machine learning
- **2022** Scientific Machine Learning through Physics-Informed Neural Networks: Where we are and What's next
- **2024** From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning

Github repository

bitzhangcy/Neural-PDE-Solver

1000+ papers by category

- **Application papers**
  __Mechanics__, Chemistry, Robotics,…
- **Network architecture**
  Convolution, Graph Network, __Separable-PINN__, KAN…
- **New implementation**
  Variational form, __Mixed PINN__…
- **Extension**
  __Uncertainty Quantification__…
- **Improving training**
  Sampling strategy, Fourier features, Loss balance…

⚠ *Convergence issues*

[1] Karniadakis, George Em, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. "Physics-Informed Machine Learning." Nature Reviews Physics 3, no. 6 (June 2021): 422–40. https://doi.org/10.1038/s42254-021-00314-5.
[2] Cuomo, Salvatore, Vincenzo Schiano di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. "Scientific Machine Learning through Physics-Informed Neural Networks: Where We Are and What's Next." arXiv, June 7, 2022. https://doi.org/10.48550/arXiv.2201.05624
[3] Toscano, Juan Diego, Vivek Oommen, Alan John Varghese, Zongren Zou, Nazanin Ahmadi Daryakenari, Chenxi Wu, and George Em Karniadakis. "From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning." arXiv, October 22, 2024. https://doi.org/10.48550/arXiv.2410.13228.

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

Techniques from the literature

An Expert's Guide to Training Physics-informed Neural Networks

Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris

- Non-dimensionalization
- Fourier features
- Causal/curriculum training
- Loss weighting strategies

Problem implementation

- Maximizing hard constraints
- Mixed-PINN formulation

Network architecture

- **S**eparable **PINN**
- **K**olmogorov-**A**rnold **N**etwork

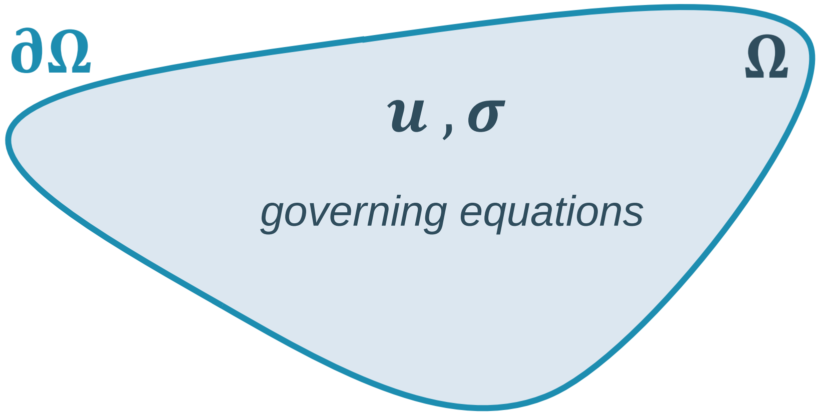
Optimization algorithm

- Adam + LBFGS
- Adaptative sampling

Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-Informed Neural Networks*, 2023.

PINNs framework for continuum mechanics

Boundary **V**alue **P**roblem (**BVP**):



Boundary conditions :

$\partial\Omega_u \,:\, u = u_{BC}$	*in displacement*
$\partial\Omega_f \,:\, \sigma.\vec{n} = \vec{F}_{BC}$	*in stress*

Fields :

u	*displacement*
ε	*strain tensor*
σ	*stress tensor*

→ *Which output for the network ?*

Equations :

$\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$	*small deformation*
$\sigma_{ij} = f(\varepsilon_{ij})$	*material law*
$\sigma_{ij,j} + f_i = 0$	*momentum balance*

$$\rightarrow \quad L_{PDE} = \sum_{\tilde{\sigma}\in\Omega} \left\| \widetilde{\sigma_{ij,j}} + f_i \right\|$$

PINNs framework for continuum mechanics

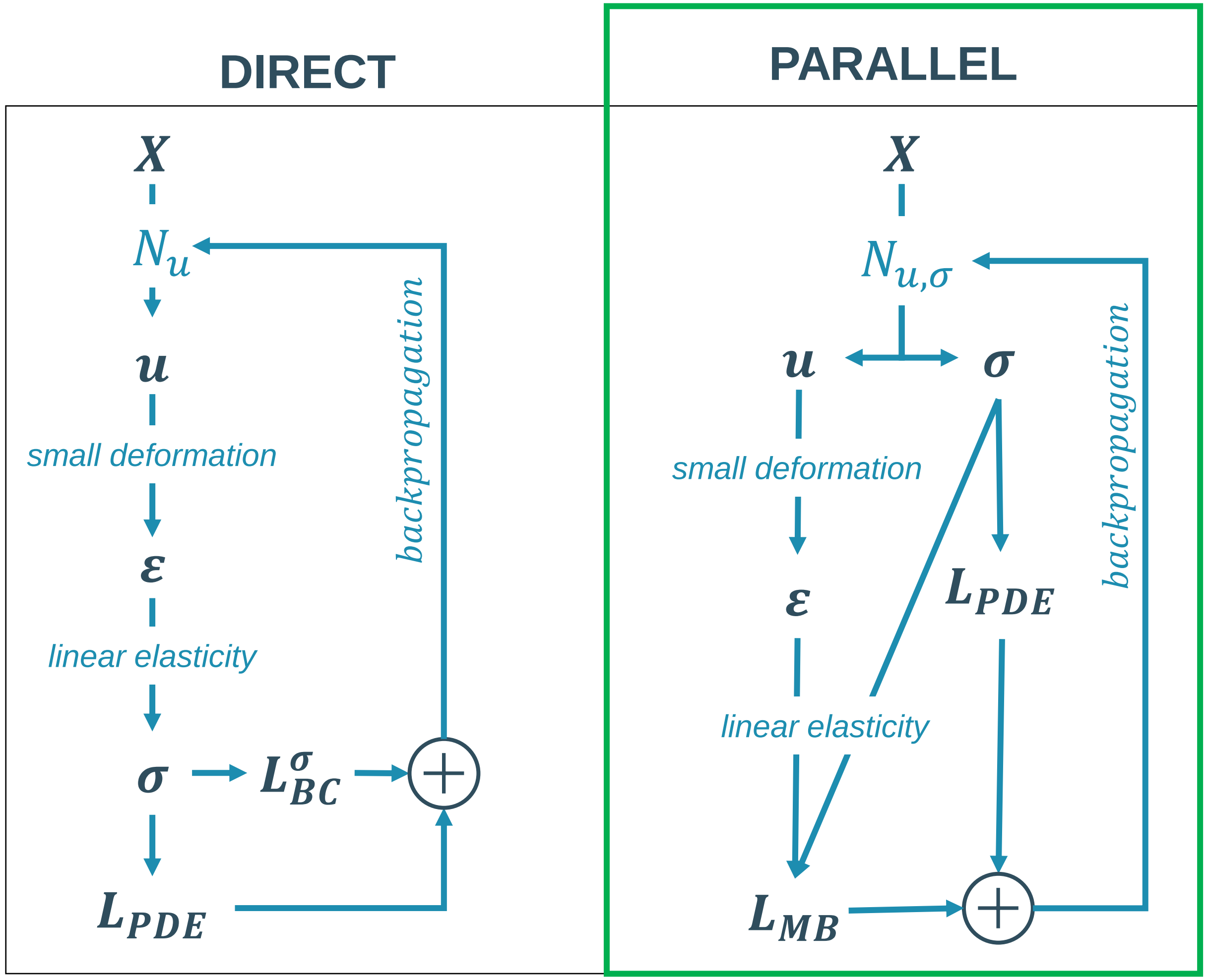
Which output for the network ? $\boldsymbol{u}$? $\boldsymbol{\sigma}$? $\boldsymbol{both}$?

Goal : maximise the number of **hard constraint**

Hard boundary condition only on the output of a neural network

	DIRECT	PARALLEL
$\boldsymbol{BC_u}$	Hard	Hard
$\boldsymbol{BC_\sigma}$	Soft	Hard
Material law	Hard	Soft

+ limit the order of the derivatives



Techniques from the literature

An Expert's Guide to Training Physics-informed Neural Networks

Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris

- non-dimensionalization
- Fourier feature
- causal/curriculum training
- loss weighting strategies

Problem implementation

- Maximizing hard constraints
- Mixed-PINN formulation

Network architecture

- **S**eparable **PINN**
- **K**olmogorov-**A**rnold **N**etwork

Optimization algorithm

- Adam + LBFGS
- Adaptative sampling

Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-Informed Neural Networks*, 2023.

Techniques from the literature

An Expert's Guide to Training Physics-informed Neural Networks

Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris

- non-dimensionalization
- Fourier feature
- causal/curriculum training
- loss weighting strategies

Problem implementation

- Maximizing hard constraints
- Mixed-PINN formulation

Network architecture

- **S**eparable **PINN**
- **K**olmogorov-**A**rnold **N**etwork

Optimization algorithm

- Adam + LBFGS
- Adaptative sampling

Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-Informed Neural Networks*, 2023.

Introduction to Separable-PINN

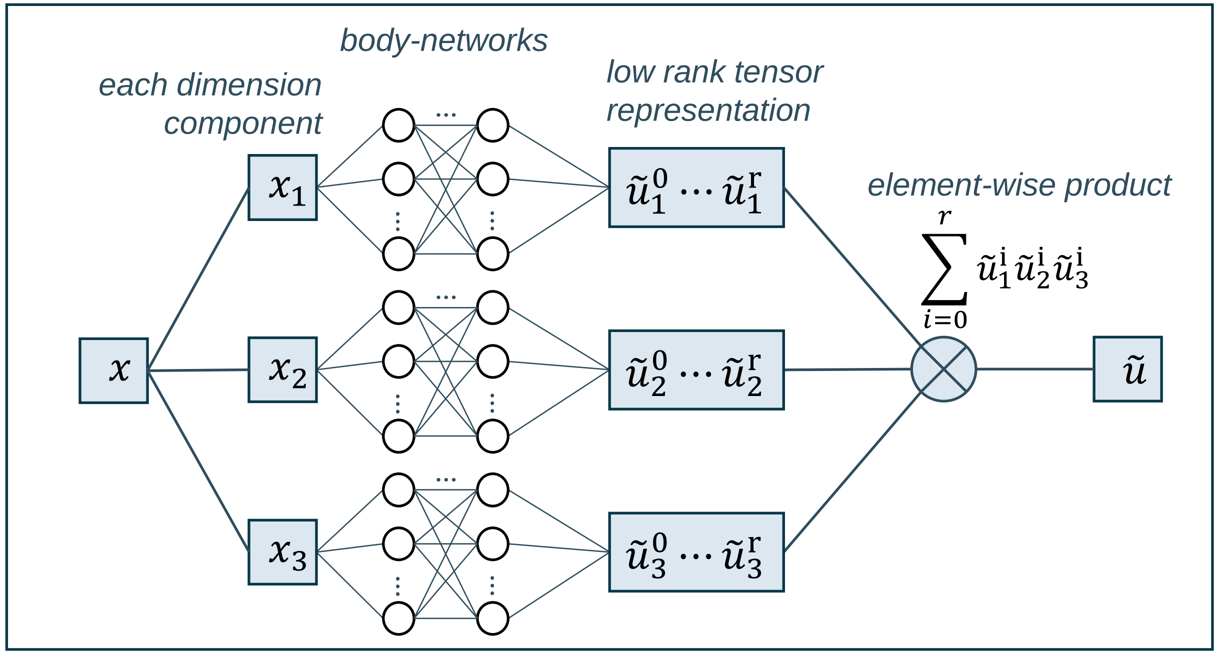


Fig. 1: SPINN architecture for a 3D problem

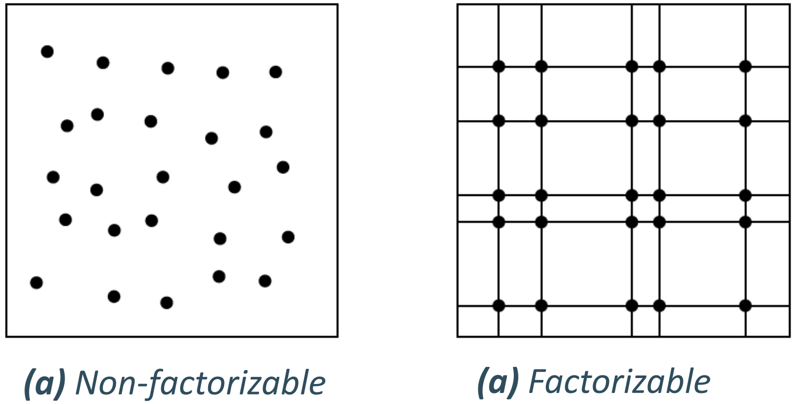


Fig. 2: Illustrative example of factorizable-coordinates constraint required for SPINN sampling

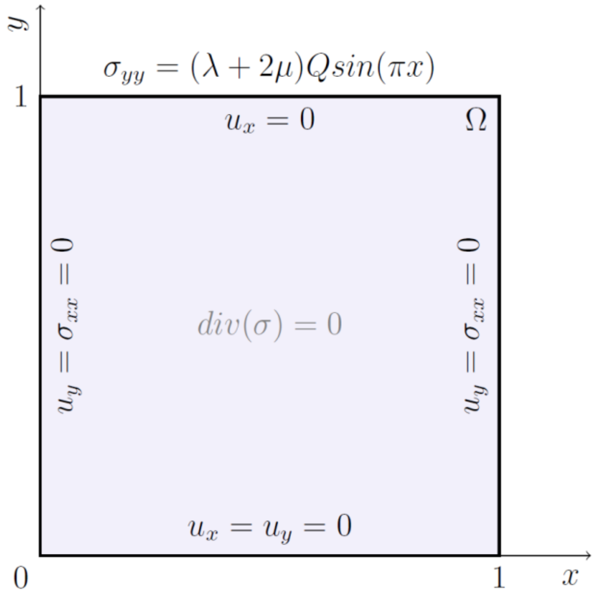
- A different network for each dimension
- Low-rank tensor approximation
- Expressive enough

- Faster computation
- N^d points for the price of $N \times d$
- Bring limitations in the geometry

[1] Cho, Junwoo, Seungtae Nam, Hyunmo Yang, Seok-Bae Yun, Youngjoon Hong, et Eunbyung Park. 2023. « Separable Physics-Informed Neural Networks ». arXiv. https://doi.org/10.48550/arXiv.2306.15969.

PINN for continuum mechanics : example from literature[1]

Domain and boundary condition :



Volumic forces :

$$f_x = \lambda \left[4\pi^2 \cos(2\pi x)\sin(\pi y) - \pi \cos(\pi x) Q y^3\right] + \mu \left[9\pi^2 \cos(2\pi x)\sin(\pi y) - \pi \cos(\pi x) Q y^3\right]$$

$$f_y = \lambda \left[-3\sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x)\cos(\pi y)\right] + \mu \left[-6\sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x)\cos(\pi y) + \pi^2 \sin(\pi x) Q y^4/4\right].$$

Exact solution :

$$u_x(x,y) = \cos(2\pi x)\sin(\pi y),$$
$$u_y(x,y) = \sin(\pi x) Q y^4/4.$$

Parameters :

$\lambda = 1$
$\mu = 0{,}5$

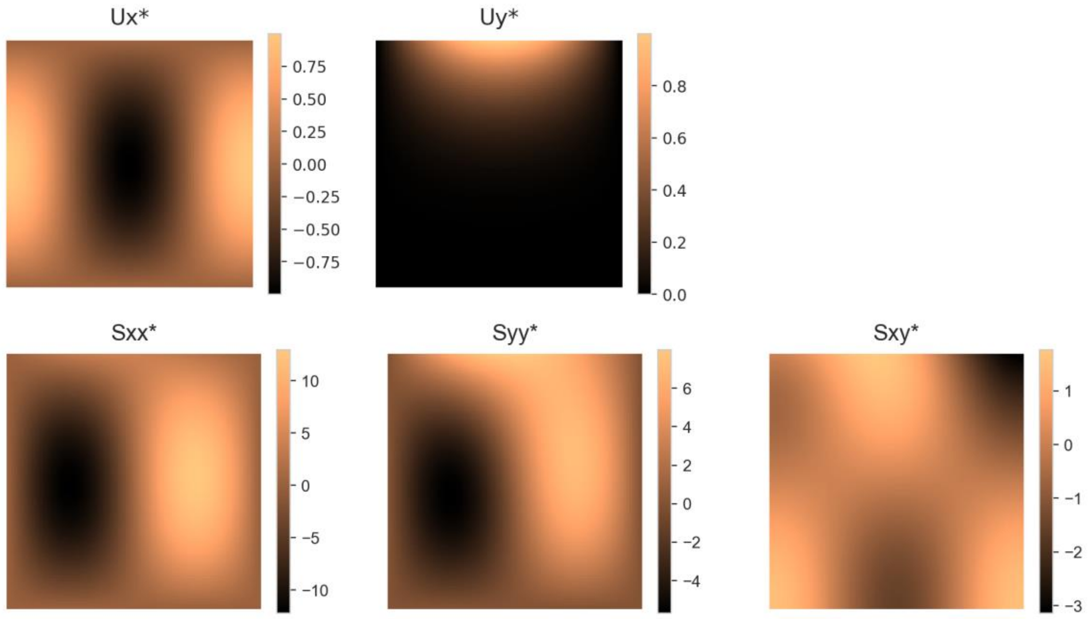
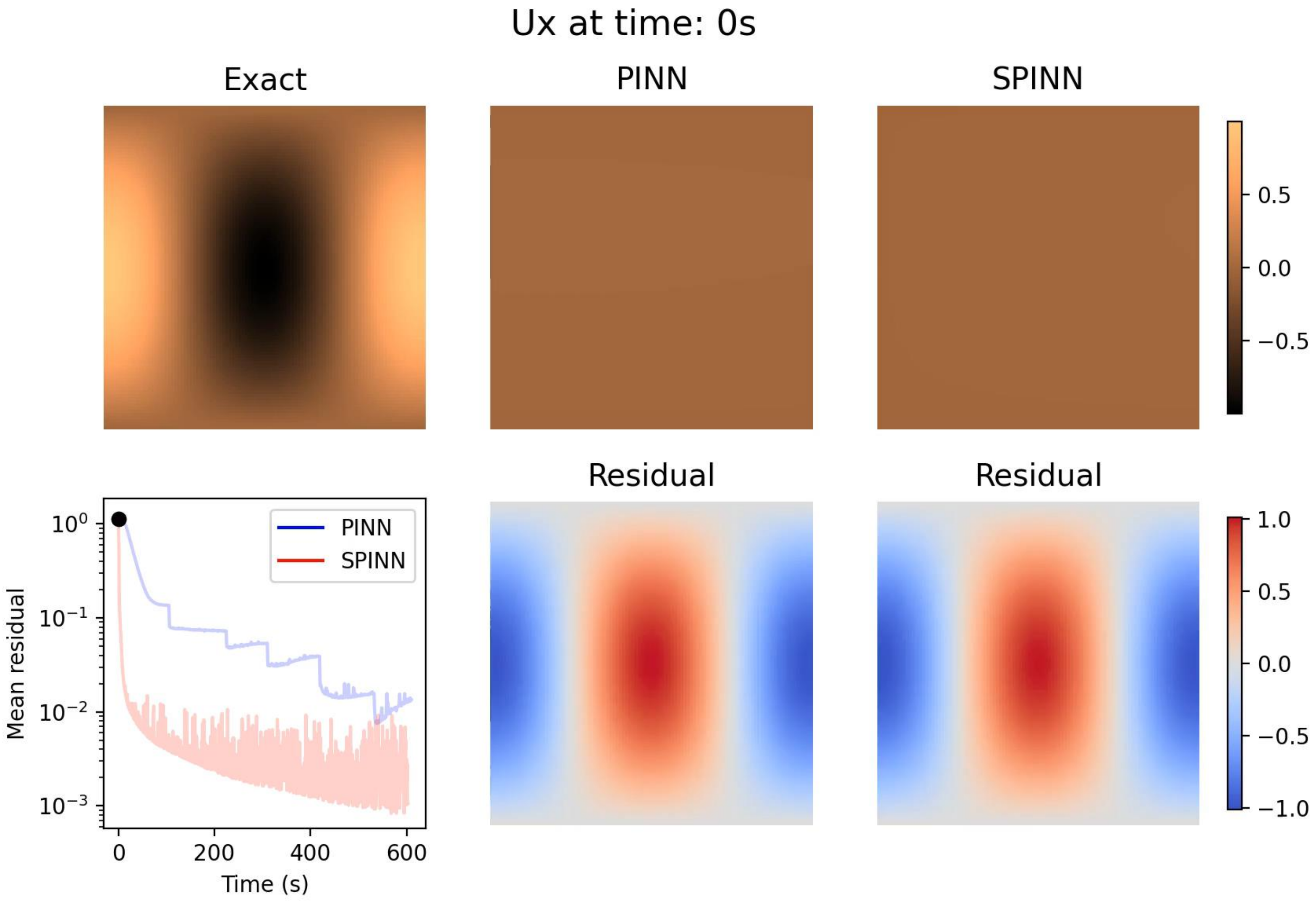


Fig. 2 : Exact displacement and stress solution of the problem

[1] Haghighat, Ehsan, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. “A Deep Learning Framework for Solution and Discovery in Solid Mechanics.” *ArXiv:2003.02751 [Cs, Stat]*, May 6, 2020.

PINN vs SPINN on a continuum mechanics example



SPINN outperforms in both speed and accuracy

KU LEUVEN

Techniques from the literature

An Expert's Guide to Training Physics-informed Neural Networks
Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris

- non-dimensionalization
- Fourier feature
- causal/curriculum training
- loss weighting strategies

Problem implementation

- Maximizing hard constraints
- Mixed-PINN formulation

Network architecture

- **S**eparable **PINN**
- **K**olmogorov-**A**rnold **N**etwork

Optimization algorithm

- Adam + LBFGS
- Adaptative sampling

Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-Informed Neural Networks*, 2023.

Techniques from the literature

An Expert's Guide to Training Physics-informed Neural Networks

Sifan Wang, Shyam Sankaran, Hanwen Wang, Paris Perdikaris

- non-dimensionalization
- Fourier feature
- causal/curriculum training
- loss weighting strategies

Problem implementation

- Maximizing hard constraints
- Mixed-PINN formulation

Network architecture

- **S**eparable **PINN**
- **K**olmogorov-**A**rnold **N**etwork

Optimization algorithm

- Adam + LBFGS
- Adaptative sampling

Wang, Sifan, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. *An Expert's Guide to Training Physics-Informed Neural Networks*, 2023.

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

Resources

Separable Physics-Informed Neural Networks for Robust Inverse Quantification in Solid Mechanics

Damien Bonnet-Eymard, Augustin Persoons, Matthias Faes, and David Moens

Presented at ISRERM 2024 conference - Available on ResearchGate

Code and results are available online :

www.github.com/bonneted/ISRERM2024

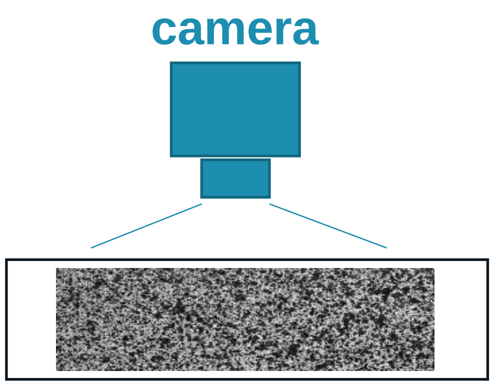
relies on

SPINN branch of DeepXDE

- bonneted/deepxde at SPINN (github.com)
- still under development

Inverse quantification from full field measurements

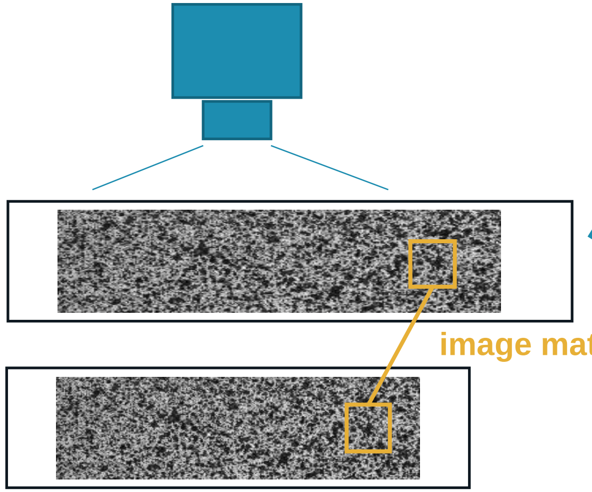
Digital **I**mage **C**orrelation



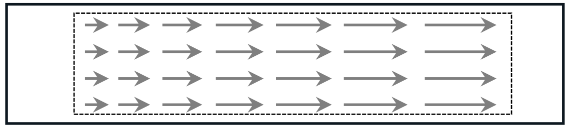
Inverse quantification from full field measurements

Digital **I**mage **C**orrelation

camera



Full field measurement



Inverse quantification :

- ***V**irtual **F**ield **M**ethod*
- ***F**inite **E**lement **M**ethod **U**pdating*
- …

Can struggle on complex geometry or material behavior

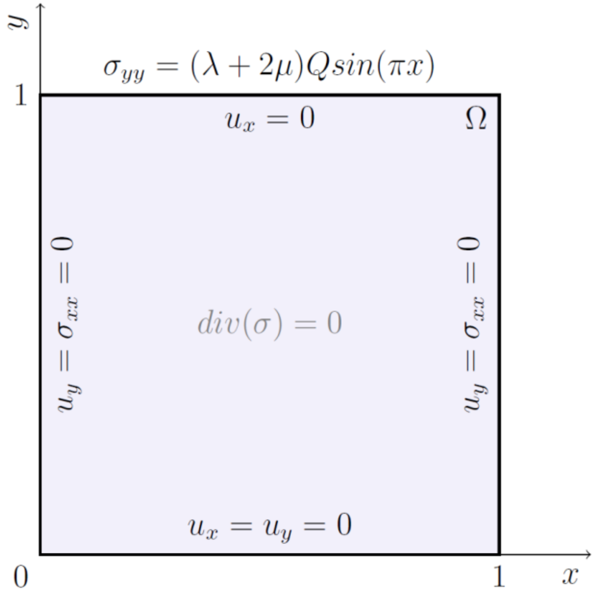
Use **P**hysics **I**nformed **N**eural **N**etworks as an alternative inverse method

Material properties

?

PINNs for continuum mechanics : case study from literature[1]

Domain and boundary condition :



Volumic forces :

$$
\begin{aligned}
f_x &= \lambda \left[4\pi^2 \cos(2\pi x)\sin(\pi y) - \pi \cos(\pi x) Q y^3\right] \\
&\quad + \mu \left[9\pi^2 \cos(2\pi x)\sin(\pi y) - \pi \cos(\pi x) Q y^3\right] \\
f_y &= \lambda \left[-3\sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x)\cos(\pi y)\right] \\
&\quad + \mu \left[-6\sin(\pi x) Q y^2 + 2\pi^2 \sin(2\pi x)\cos(\pi y) + \pi^2 \sin(\pi x) Q y^4/4\right].
\end{aligned}
$$

Exact solution :

$$
\begin{aligned}
u_x(x,y) &= \cos(2\pi x)\sin(\pi y), \\
u_y(x,y) &= \sin(\pi x) Q y^4/4.
\end{aligned}
$$

Parameters :

$\lambda = 1$
$\mu = 0{,}5$

Find elasticity parameters

+ measurements

ground truth displacement

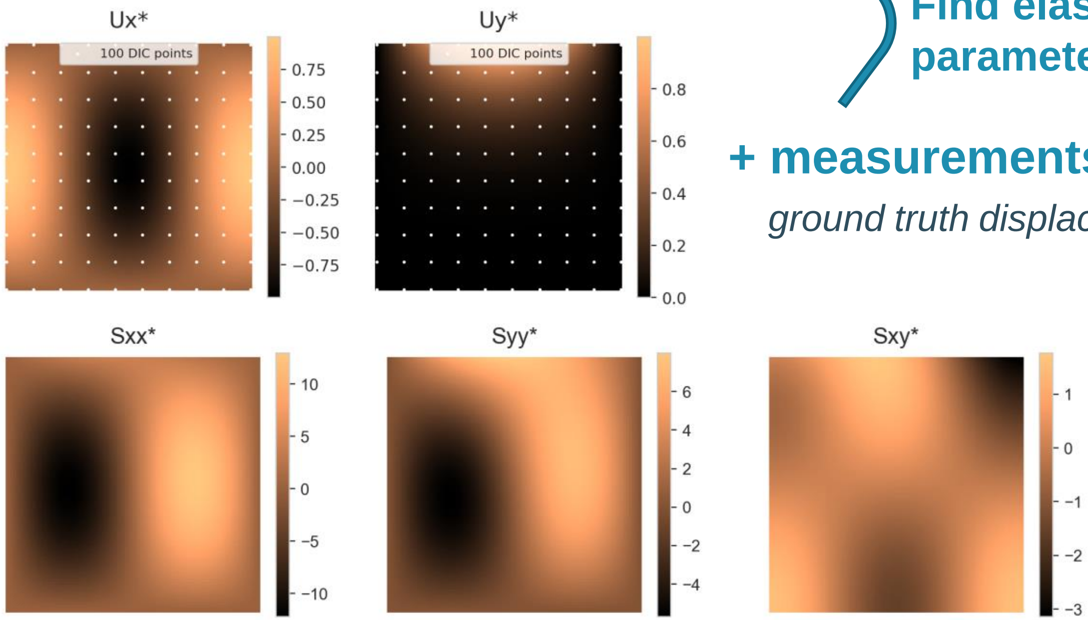
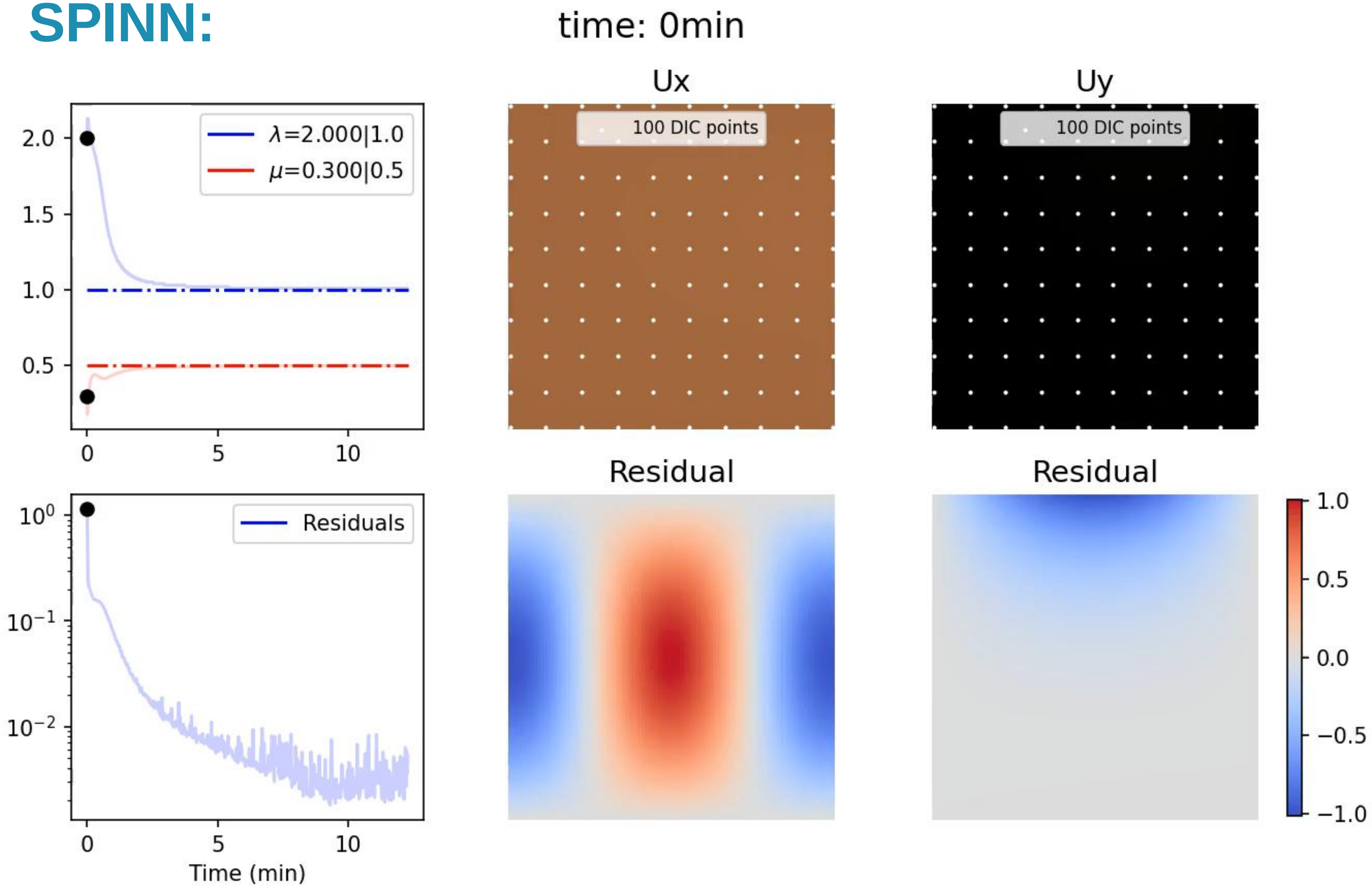


Fig. 2 : Exact displacement and stress solution of the problem

[1] Haghighat, Ehsan, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. "A Deep Learning Framework for Solution and Discovery in Solid Mechanics." *ArXiv:2003.02751 [Cs, Stat]*, May 6, 2020.

Inverse quantification of elasticity parameters : SPINN

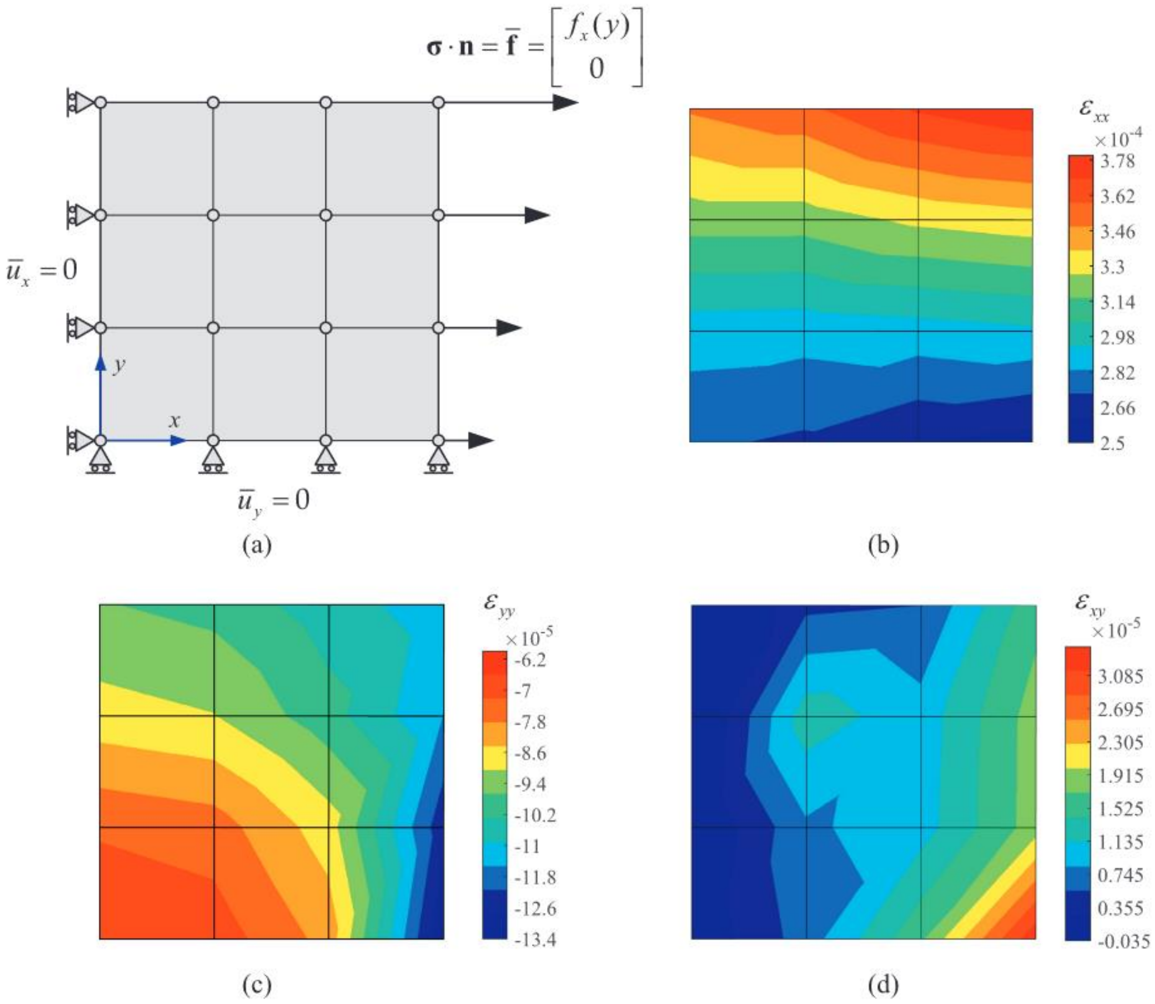
SPINN:



Converge directly in a few minutes

More realistic example ?

Inverse quantification benchmark from the literature[1]



16 simulated measurements

Using FEM as the reference

Strain corrupted with noise

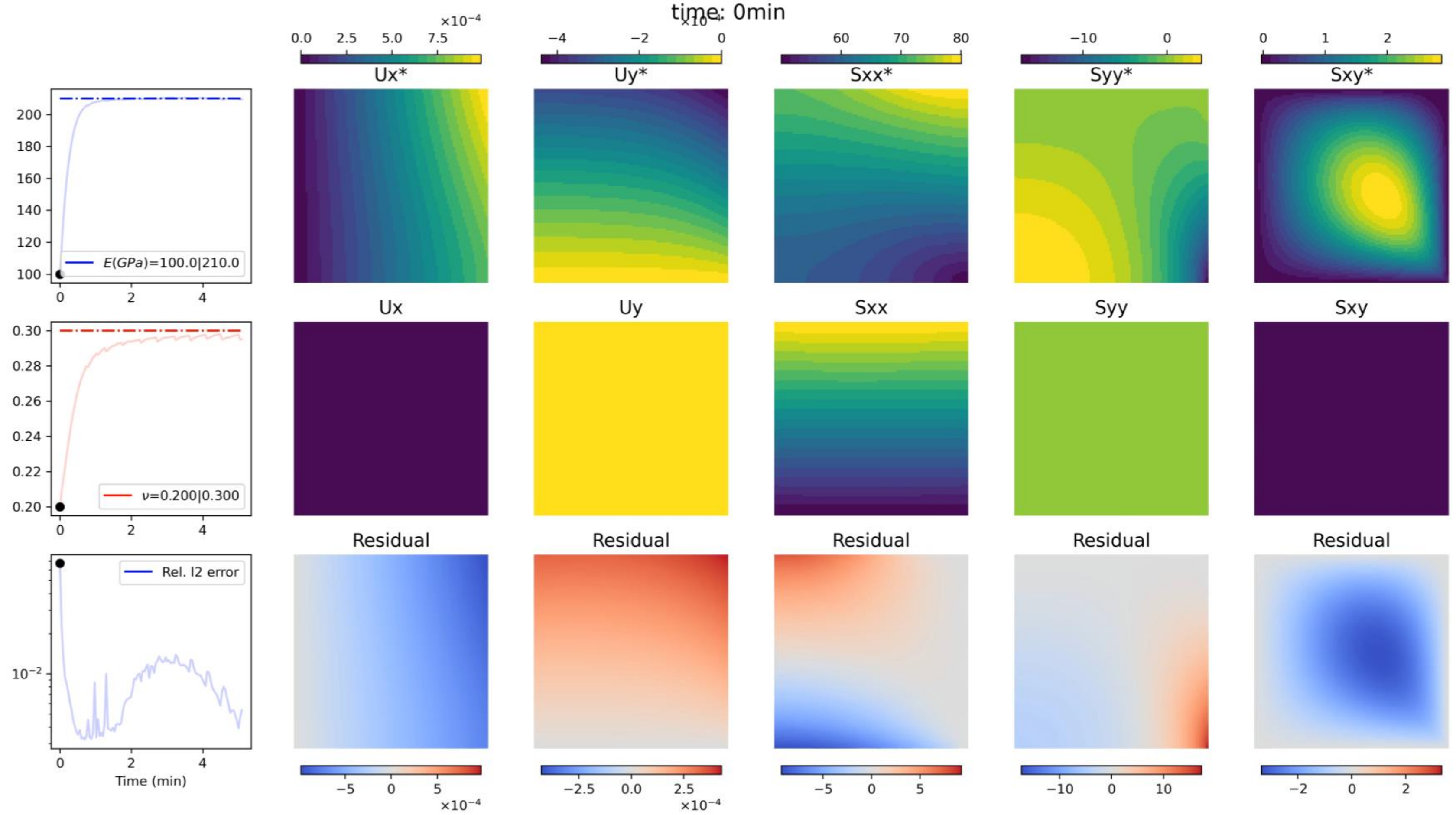
1$\mu\varepsilon$ gaussian noise (≈10% of std.)

	E (GPa)	ν	E - Error(%)	ν - Error(%)
Reference	210.00	0.3000		
FEMU	203.90	0.2706	2.90	9.789
CEGM	204.55	0.2728	2.59	9.058
EGM	195.10	0.2356	7.09	21.436
VFM	205.14	0.2753	2.31	8.207

→ **Compare with SPINN**

[1] Martins, J.M.P., António Andrade-Campos, et Sandrine Thuillier. 2018. « Comparison of inverse identification strategies for constitutive mechanical models using full-field measurements ». *International Journal of Mechanical Sciences* 145 (septembre): 330-45.

Benchmark from the literature: SPINN results



16 simulated measurements

Using FEM as the reference

Strain corrupted with noise

1$\mu\varepsilon$ gaussian noise (≈10% of std.)

	E (GPa)	ν	E - Error(%)	ν - Error(%)
Reference	210.00	0.3000		
FEMU	203.90	0.2706	2.90	9.789
CEGM	204.55	0.2728	2.59	9.058
EGM	195.10	0.2356	7.09	21.436
VFM	205.14	0.2753	2.31	8.207
SPINN	**209.5**	**0.2952**	**0.24**	**1.667**

→ **SPINN 10x more accurate**

Expected: regression model dealing with an unbiased noise

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

Resources

Physics-Informed Neural Networks to propagate random field properties of composite materials

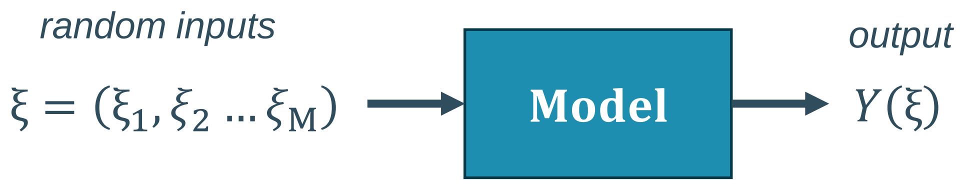
D. Bonnet-Eymard, A. Persoons, P. Gavallas, M. GR Faes, G. Stefanou, D. Moens

Presented at USD 2024 conference - Available on ResearchGate

Code and results are available online :

www.github.com/bonneted/USD2024

Polynomial Chaos Expansion (PC)



$$Y(\xi) \approx \sum_{\alpha \in \mathcal{A}} y_\alpha \Psi_\alpha(\xi)$$

multivariate polynomials of the random inputs

$\mathcal{A} \subset \mathbb{N}^M$, *multi-indices that define the truncation*

PCE coefficients to be found

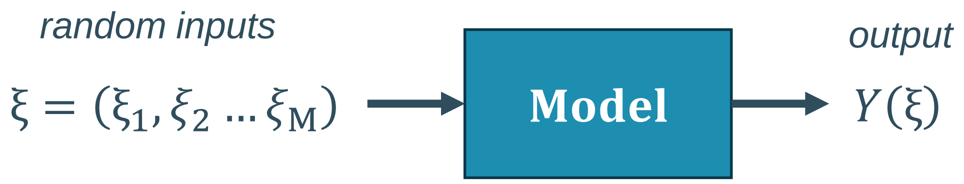
Resolution (determining y_α) :

- sampling
- y_α determined using least squares, least angle…

Space dependent output : $Y(\xi, x)$?

- discretization of the output (PCA…)
- Spatially dependant coefficients

Space dependent Polynomial Chaos Expansion



$$Y(\xi, x) \approx \sum_{\alpha \in \mathcal{A}} y_\alpha(x) \Psi_\alpha(\xi)$$

$\mathcal{A} \subset \mathbb{N}^M$, *multi-indices that define the truncation*

multivariate polynomials of the random inputs

Space dependent PCE coefficients to be found

Resolution (determining y_α) :

- sampling
- y_α determined using least squares, least angle…

Space dependent output : $Y(\xi, x)$?

- discretization of the output (PCA…)
- **Spatially dependant coefficients**

➢ Use a Neural Network as the approximator :

$$y_\alpha(x) = NN(x) \Rightarrow \textbf{PINN-PC}$$

PINN-PC[1]

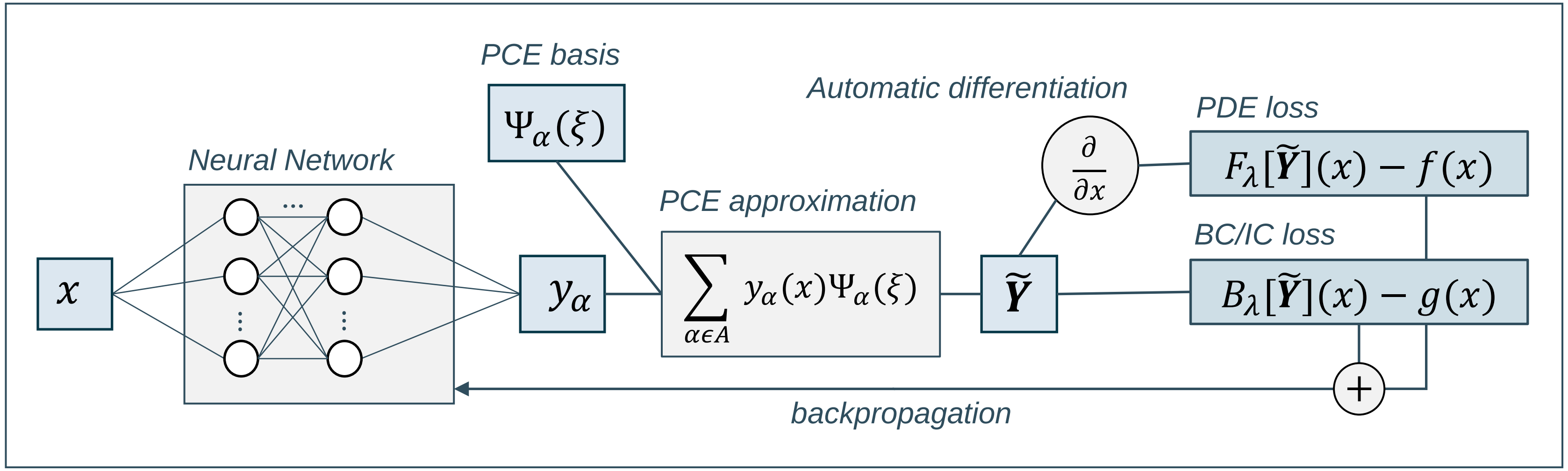


Fig. 1 : Schematic representation of the PINN-PC framework.

Several coefficients to predict :

- grouping them by their polynomial degree
- using a separate neural network for each group

⚠ **Increase the computational cost**

by a factor N = the number of samples

[1] Zhang, Dongkun, Lu Lu, Ling Guo, et George Em Karniadakis. « Quantifying Total Uncertainty in Physics-Informed Neural Networks for Solving Forward and Inverse Stochastic Problems ». *Journal of Computational Physics* 397 (15 novembre 2019): 108850. https://doi.org/10.1016/j.jcp.2019.07.048.

Poisson equation : reference solution

Problem setup :

Poisson equation

$$-\frac{d^2}{dx^2}u = f(x;\omega), \quad x \in [-1,1] \text{ and } \omega \in \Omega,$$
$$u(-1) = u(1) = 0.$$

Forcing term

$$f(x;\omega) \sim \mathcal{GP}(f_0(x), \mathrm{Cov}(x,x'))$$

$$f_0(x) = 10\sin(\pi x)$$

$$\mathrm{Cov}(x,x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{l_c^2}\right),$$

Finite difference solution :

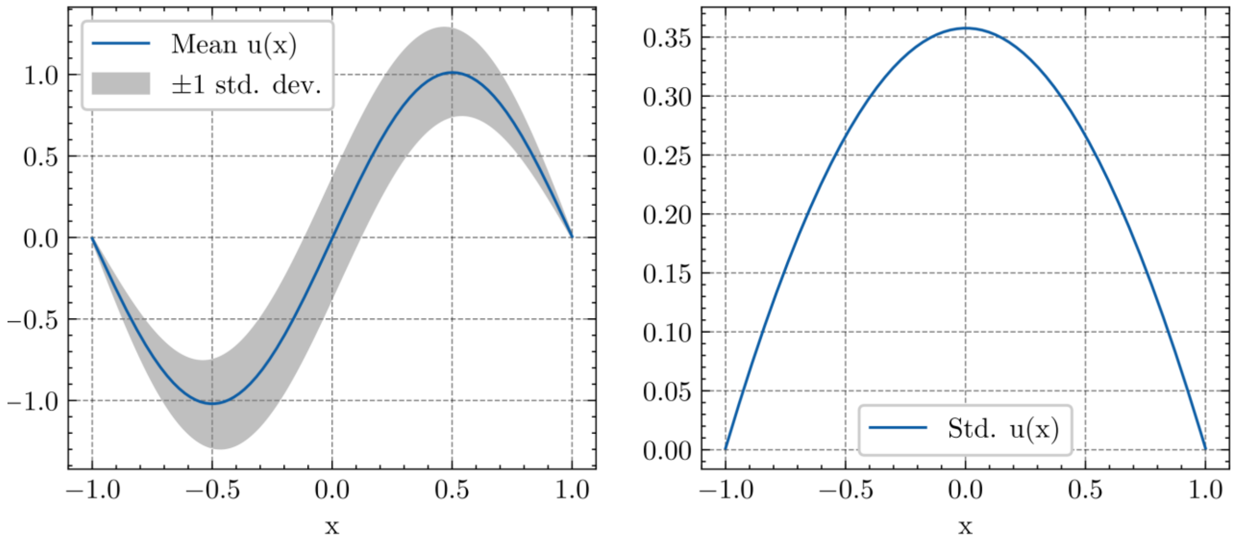


Fig. 2: Mean and standard deviation of the solution computed using 10^6 Monte Carlo simulations.

Poisson equation : random field discretization

Discretization of $f(x, \omega)$:

Forcing term

$$f(x;\omega) \sim \mathcal{GP}(f_0(x), \mathrm{Cov}(x,x'))$$

$$f_0(x) = 10\sin(\pi x)$$

$$\mathrm{Cov}(x,x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{l_c^2}\right),$$

Karhunen–Loève expansion

$$f(x) = f_0(x) + \sum_{n=1}^{\infty} \sqrt{\lambda_n}\, \phi_n(x)\, \xi_n$$

On a discrete space :

$Cov(x,x') = Mcov$

K-L $\leftrightarrow$ Spectral decomposition **of** $\boldsymbol{Mcov}$

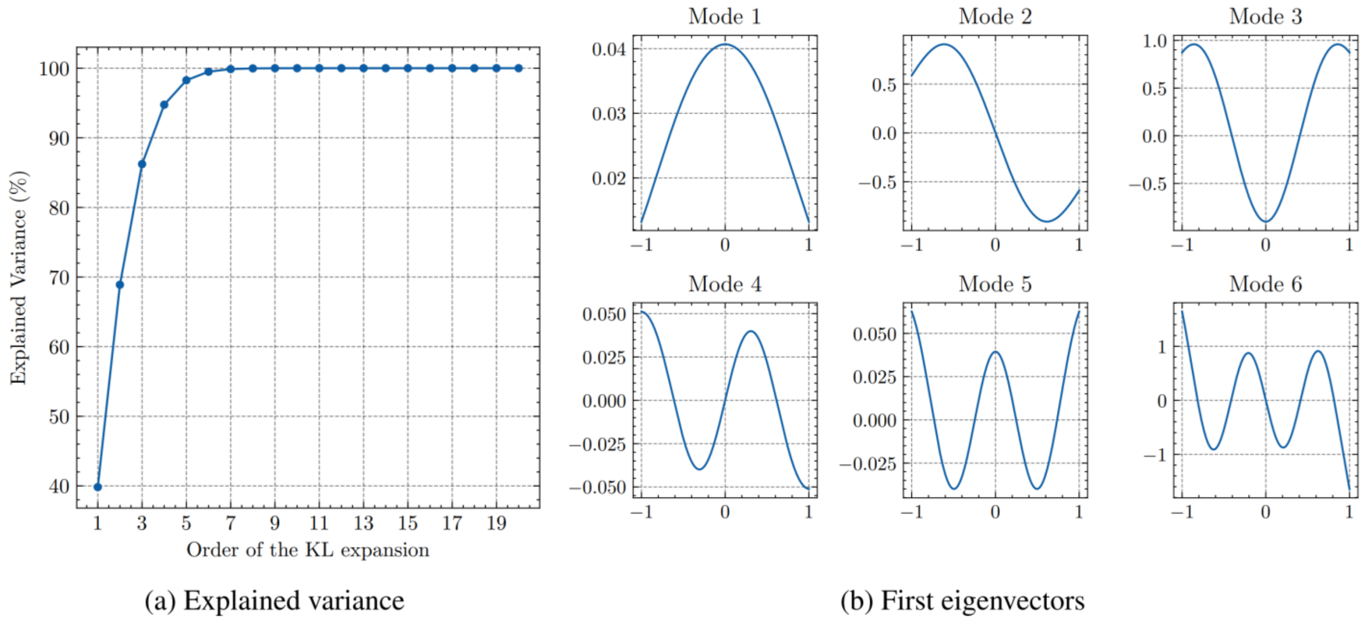


Fig. 3: Eigendecomposition of the covariance matrix to construct the K-L expansion.

Keep **6 order** to have **99%** explained variance

- *GP represented by 6 variables :* $(\xi_1, \xi_2 \dots \xi_6)$
- *1 order PCE : only 6 polynomials* $\Psi_i(\xi) = \xi_i$

Poisson equation : implementation

Neural Network

- MeanNN: [1, 4, 4, 1], to approximate $u_0(x)$
- CoeffNN: [1, 36, 36, 36, 36, 6], to approximate $y_\alpha(x)$
- Tanh activation function

Training :

- 1000 samples of f
- 13 training points in [-1, 1]
- Adam (20000 epochs, lr = 1e-3)

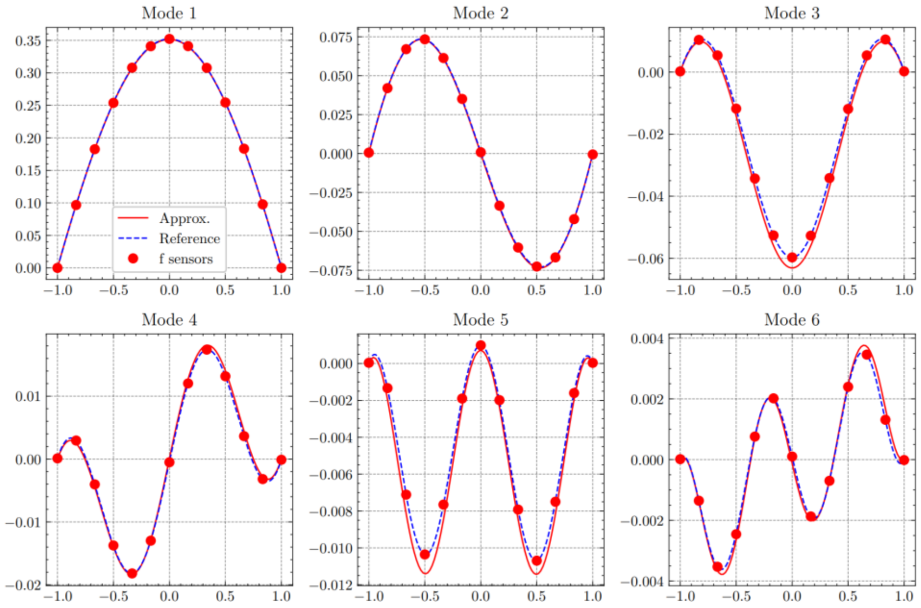


Fig. 4: approximated PCE coefficients

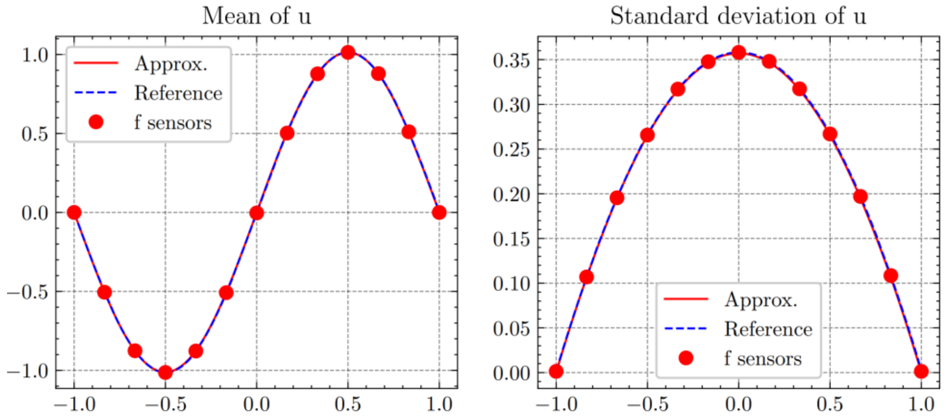


Fig. 5: Mean and standard deviation of the solution

I. Introduction to Physics-Informed Neural Networks

II. Improving the convergence of PINN

III. PINN for inverse quantification of material parameters

IV. PINN to propagate uncertainty

Conclusion

PINN: a powerful tool that is becoming increasingly mature

Idea : use an **artificial neural network** to approximate the solution of a **boundary value problem** defined by a partial differential equation (**PDE**) and boundary conditions (**BCs**)

Physics-Informed loss function : $L_{total} = L_U + L_{BC/IC} + L_{PDE}$

- L_{PDE} **:** PDE calculated through automatic differentiation
- $L_{BC/IC}$ **:** Residual between PINN approximation and BCs values
- L_U **:** Residual between PINN approximation and measurements

loss minimized → **compliance to the PDE, BCs and data**

⚠ The training (i.e., finding the network parameters that minimize the loss) rely on stochastic optimization and can struggle to converge

Techniques to improve convergence

- Hard constraints
- Mixed formulation
- Separable-PINN
- Adaptative sampling

Possible applications

- Inverse quantification
- Uncertainty propagation

Damien Bonnet-Eymard
PhD student at KU Leuven

damien.bonnet-eymard@kuleuven.be

The project leading to this application has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 955393.

GREYDIENT project
Marie Sklodowska-Curie Actions

www.greydient.eu