Leveraging knowledge to design machine learning despite the lack of data using Physical Informed Machine Learning models.

Mathilde Mougeot[1,4], Thi Nguyen Khoa Nguyen [1,2,3]

Centre Borelli, ENS Paris-Saclay, CNRS, Université Paris-Saclay[1]
CEA[2], Michelin[3],
ENSIIE[4]

October 2023

Motivation

Leveraging knowledge to design machine learning despite the lack of data using PINNS models and Transfer Learning.

In recent years, considerable progresses have been made in the implementation of decision support procedures based on machine learning methods through the exploitation of very large databases and the use of learning algorithms.
In many research or production environments, the available databases are rarely so voluminous and the question arises as to whether in this context it is reasonable to use machine learning methods.
This lesson introduces transfer learning and hybrid models that use knowledge from related application or from physics to implement efficient models with an economy of data. Several achievements will be presented that successfully use these learning approaches to design machine learning for industrial small data regimes and to develop powerful decision support tools even in cases where the initial data volume is limited.
Practical sessions will help to handle such models in python environments.

Most of this work have been done thanks to the Industrial Data Analytics and Machine Learning chair of Centre Borelli/ ENS Paris-Saclay (CEA/ Michelin, SNCF).

Outline

Outline

Backbone of Supervised Machine learning

1. Input (X) /output (Y) provided by the target application
   ex : input :output : (space/time) / velocity

2. The data set : $\mathcal{D}_n = \{(x_i, y_i),\ 1 \leq i \leq n, x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$

3. The model f and its parameters θ : $f_\theta(x)$
   f_θ may represent a neural network, a decision tree, a boosting model...

4. The cost/ loss function measures how well f_θ predicts y.
   Regression task : $\ell(y, z) = (y - z)^2$

5. The learning/ optimization Goal "learn"
   $f \in \mathcal{H} = \{\text{measurable functions } \mathcal{X} \to \mathcal{Y}\}$ s.t. to minimize the (empirical) risk

$$\mathcal{R}(f) = \mathbb{E}_{(X,Y)\sim P}[\ell(Y, f_\theta(X))]$$

6. The calibrated ready-to-use Machine learning model : $\hat{f}_\theta$

$\hat{f}_\theta$ depends on f_θ, $\mathcal{D}_n$, ℓ and on several (optimization) parameters...

Warm-up

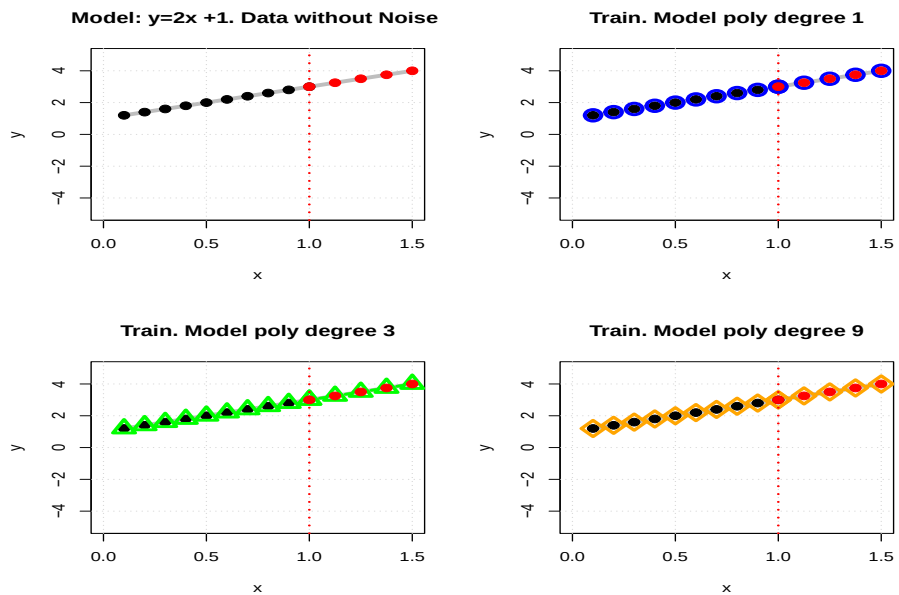
Choosing a model in the Machine Learning framework.

- $f_\alpha(x, \theta_\alpha)$ a family of functions :
  α represents *the model class* used to model the data and make prediction without knowing the function $f(x)$.
  - $x \to z = (1, x, x^2, x^3, \ldots, x^p)$,
    z provides a multivariate transformation of the initial x.
    z is usually denoted by **features** (feature vector of size p).
    Different p parameters provide *different complexities*.
  - Illustration with the polynomial regression families of order p :
    ML Model : $f_{\theta_p(x)} = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_p x^p$
- DataSet of observations : $\mathcal{D}_n = \{(x_i, y_i),\ 1 \leq i \leq n,\ x_i \in \mathbb{R},\ y_i \in \mathbb{R}\}$.
- **Learning step.**
  Optimization procedure : minimisation of the empirical risk on Training data.
  $\hat{\theta} = \arg\min_{\theta_p} E(\mathcal{D}_n, f_{\theta_p}) = \sum_i (y_i - f_{\theta_p}(x_i))^2$
- **Predictive power.** The effectiveness of the model is evaluated on a different dataset, the test dataset.

Modeling issues

Illustration. Observations and ML models...

Impact of the choice of the ML model (here polynomial).
We consider an unknown function : $y = 2x + 1$, [Mehta *et al.*, 2019]
Training 10 obs. with no noise (black points). Test data set (5 red points). 3 ML models
$x \to z = (1, x, x^2, x^3, \ldots, x^p)$ with $p = 1$, $p = 3$, $p = 9$

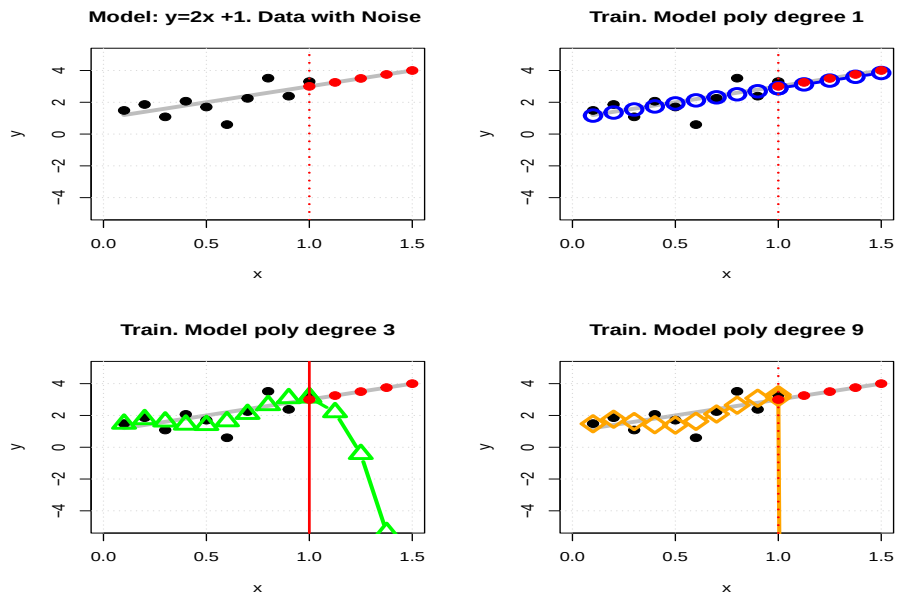


$\to$ With no noise, all the ML models perform well on the Test data.

Modeling issues

Illustration. Observations with noise and same previous ML models...

Impact of the choice of the ML model and the training data set.
Unknown function : $y = 2x + 1$,
3 ML models $x \to z = (1, x, x^2, x^3, \ldots, x^p)$ with $p = 1$, $p = 3$, $p = 9$
Training 10 obs. with noise (black points). Test data set (5 red points).

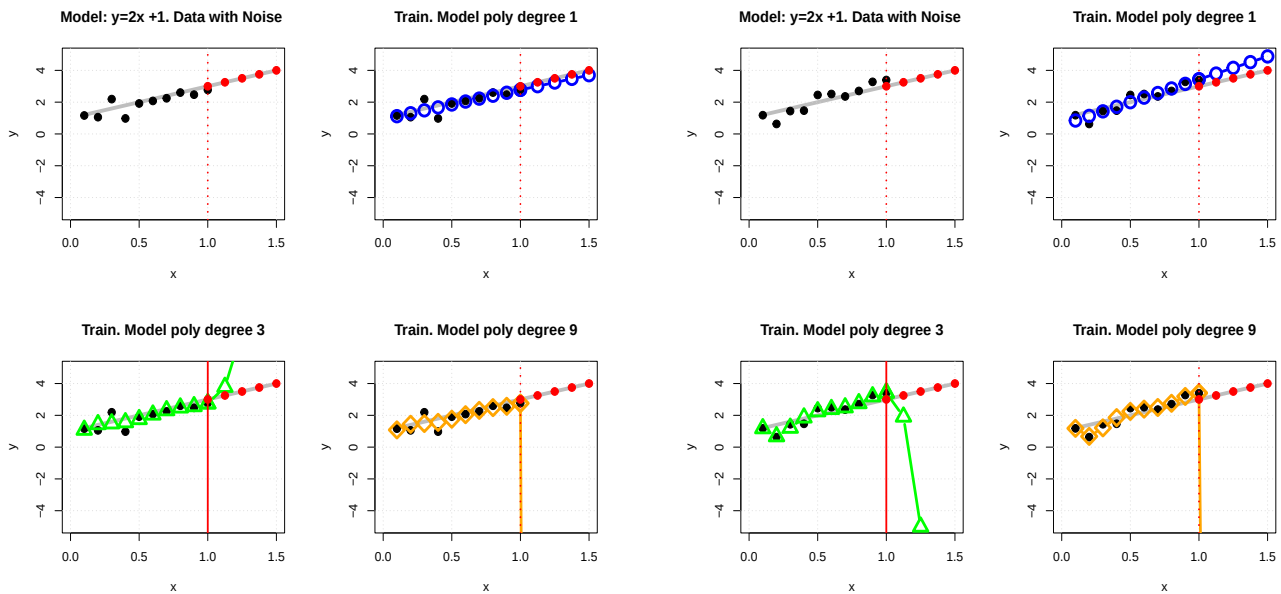


$\to$ With noise in the training, the less complex/ contrained ML model performs much better for new data (outside the learning domain).

Modeling issues

Illustration. Observations with noise and ML models...

Impact of the choice of the ML model and the training data set.
Unknown function : $y = 2x + 1$, Training 10 obs. with noise (black points).
3 ML models $x \to z = (1, x, x^2, x^3, \ldots, x^p)$ with $p = 1$, $p = 3$, $p = 9$
Test data set (5 red points).

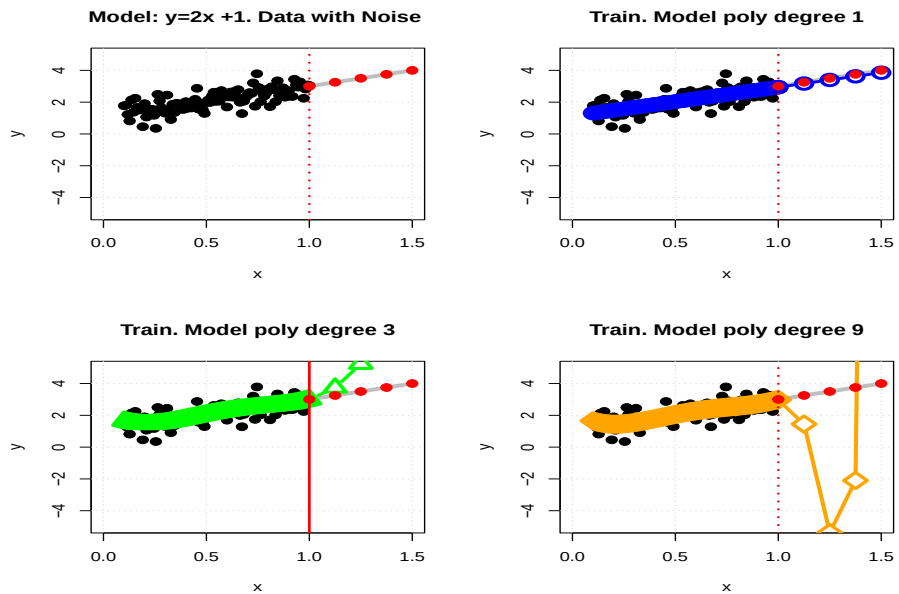


$\to$ With noise in the training, the less complex/ contrained ML model performs much better for new data (outside the learning domain).

Modeling issues

Illustration. Observations with noise and ML models...

Impact of the choice of the ML model, the size of the training data set.
Unknown function : $y = 2x + 1$, Training 100 observations with noise (black points).
3 ML models $x \to z = (1, x, x^2, x^3, \ldots, x^p)$ with $p = 1$, $p = 3$, $p = 9$
Test data set (5 red points).

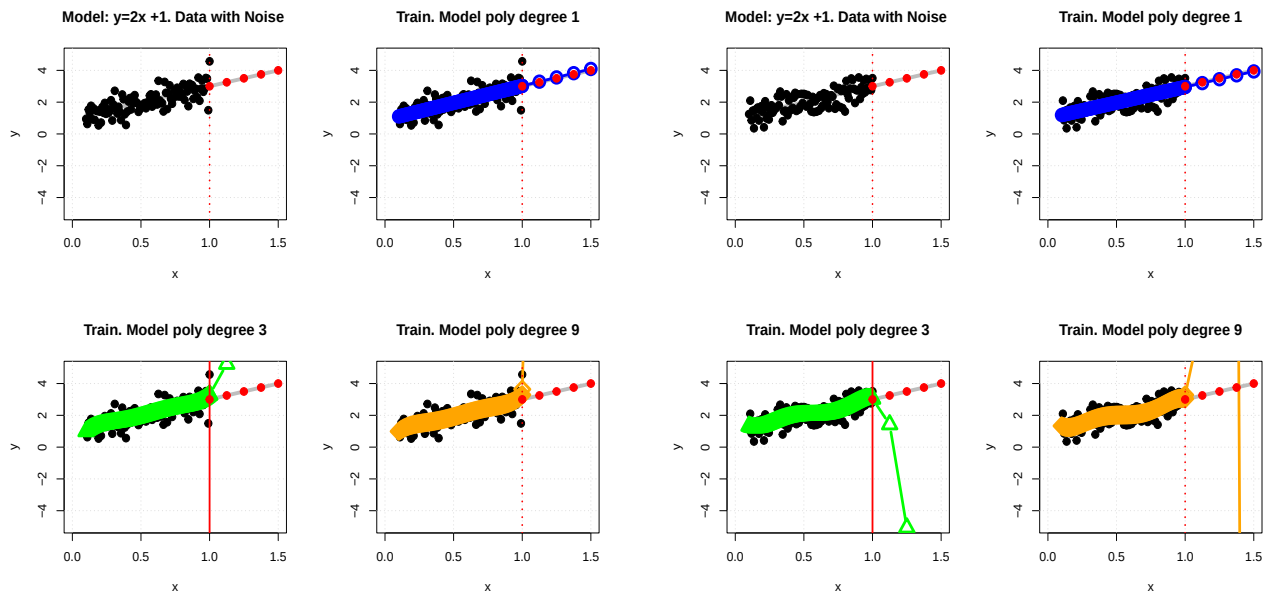


$\to$ With more data in the training set, the parameters of the correct model are better estimated (less variations for various training data set)

Modeling issues

Illustration. Observations with noise and ML models...

Impact of the choice of the ML model, the size of the training data set.
Unknown function : $y = 2x + 1$, Training 100 obs. with noise (black points).
3 ML models $x \to z = (1, x, x^2, x^3, \ldots, x^p)$ with $p = 1$, $p = 3$, $p = 9$
Test data set (5 red points).



$\to$ With more data in the training set, the parameters of the correct model are better estimated (less variations for various training data set)

Modeling issues

Knowledge in data...

- Fitting is not predicting. Fitting existing data well is fundamentally different from making predictions about new data.
  At "small" sample sizes, noise can create fluctuations in the data that look like genuine patterns.

- Using a complex model can result in overfitting. Overfitting degrade the predictive performance of the model, the ability to predict on new data.

- As simple models (like a linear function) cannot represent complicated patterns in the data, so they are forced to ignore the fluctuations and to focus on the larger trends.
  Simple models can be better at prediction than complex models due to the bias-variance tradeoff.
  It takes less data to train a simple model than a complex one.

- Interpolation vs extrapolation. It is difficult to generalize beyond the situations encountered in the training data set

Modeling issues

The biais and variance trade-off...

General setting for binary classification :

- $\mathcal{H} = \{\text{measurable functions } \mathbb{R}^d \to \{0, 1\}\}$
- Best solution : $g^* = \arg\min_{g \in \mathcal{H}} \mathcal{R}(g)$
- Class $h \subset \mathcal{H}$ of functions
- Ideal target in h : $g_h^* = \arg\min_{g \in h} \mathcal{R}(g)$
- Estimate in h : $\hat{g}_h$ obtained with some procedure thanks to data

Approximation error and estimation error (Bias/Variance)

$$\mathcal{R}(\hat{g}_h) - \mathcal{R}(g^*) = \mathcal{R}(g_h^*) - \mathcal{R}(g^*) + \mathcal{R}(\hat{g}_h) - \mathcal{R}(g_h^*)$$

- $\mathcal{R}(g_h^*) - \mathcal{R}(g^*)$: Approximation error can be large if the model h is not well chosen
- $\mathcal{R}(\hat{g}_h) - \mathcal{R}(g_h^*)$: Estimation error can be large if the model is complex!

Machine learning design for small industrial data regimes

Starting from the observation :

- The cost of labelled data acquisition may be prohibitive.
- In this *small data regime*, the vaste majority of state-of-the-art machine learning techniques are lacking robustness.
- Industrial applications faced with the challenge of drawing conclusions and making decision under partial information.

A vast amount of prior knowledge can act as a regularization agent that constrains the space of admissible solutions to a manageable size.

- Transfer learning/ domain adaptation given source domains
- Physics Informed modeling

Knowledge in ML modeling

Big data ...
$\rightarrow$ Without extra information, first knowledge lies in the observation data.
Large amounts of training data combined with deep neural architecture gave birth to solutions outperforming previously dominating methods.
Examples :

- Image net classification with deep convolutional neural networks, Krizhevsky
- Plant net application
- DeepL

ML models for physics..
- Observations data ought to reflect the underlaying (physical) properties principles that dictate their generation.[Rich data].
- The input/output relation mimics the (physics) phenomena.
- No direct mechanism let to embed the (physical) principles into a ML model during its training phase.

Knowledge in ML modeling

Data augmentation and tailored knowledge...

Extra Prior knowledge can provide rich information not existing or hard to extract in limited training data and helps improve the data efficiency, the ability to generalize, and the plausibility of resulting models.

- Data augmentation.

Easy for Image classification tasks (symmetry, rotation, texture transformations).

- Tailored knowledge [Features, architecture, function properties]

1. **Feature engineering.** $x_{raw} \to x \to f_\theta(x) \sim y$
   Ex. Wavelet based scattering transform, Fourier transform.
   Ex : sounds classification for Delphin challenge classification (frequential data).
2. **Design of specialized NN architecture** associated with a given predictive task. Symmetry groups as rotation, homothety, translation may implement an intrinsic geometry of f_θ
   $x \to f_\theta(x) \sim y$
   Ex : Convolutional NN [CNN] by craflty respecting invariance along the groups of symmetries
3. **Multi-Task Learning**
   Introduction of knowledge in the cost function, in the optimisation process. Ex : Physical Informed Neural Network

Outline

From Physics

to Machine learning

models

Physics models- Numerical methods-Solvers

The study of a given phenomena may be described thanks to a system of equations, based of Physics knowledge related for example in solid, fluid mechanics....

Evolution of a continuous value $v(x, t)$ using a generic form of PDE :

$$\frac{\partial v}{\partial x} = F(t, x, v, \frac{\partial v}{\partial x_i}, \frac{\partial^2 v}{\partial x_i \partial x_j}, \ldots)$$

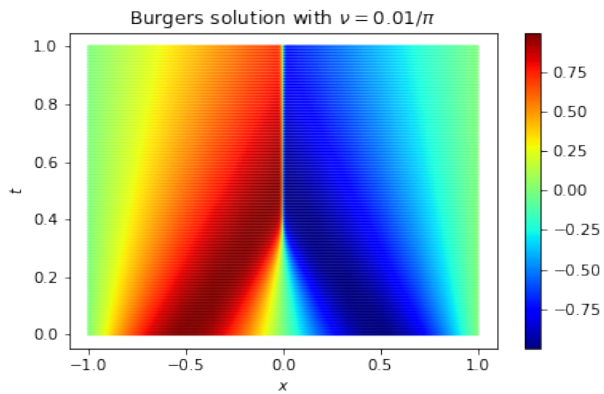
- Computational methods solve these differential equations using numerical methods with *discretization techniques* like finite volume method mostly used in computational fluid dynamics.
- Depending on the complexity of the problem, these methods tend to be *very computationnaly expensive* and require a high level of methodological understanding in the generation of the discretization (*meshing*).
- Simulation softwares have been developped
  - Opensource : OpenFOAM open source software for computational fluid dynamics (CFD), OpeFEM : A free and open source software to solve partial differential equations (PDE) using the Finite Element Method (FEM)....
  - Proprietary simulation software for conception, design, monitoring...

Physics models

Example. Seminal Burger's equation.
In computational fluid dynamics, the seminal Burger equation describes the speed of a fluid (u) on a given domain in space (x) and time (t) characterized by a viscosity parameter (ν).
and occurs in various areas of applied mathematics, such as fluid mechanics.

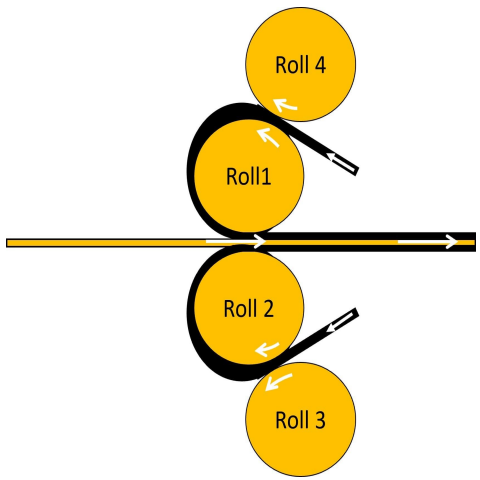
The Burger's equation :
in one space dimension, t along with the Dirichlet boundary condition (ν fixed) :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0$$
$$u(0, x) = -sin(\pi x)$$
$$u(t, -1) = u(t, 1) = 0$$



Physics models

Example. A more complex industrial model : the Rubber calendering modeling.
In the industry of tire manufacturing, calendering is a mechanical process used to create and assemble thin tissues of rubber [Nguyen *et al.*, 2022].
From the physical point of view, the rubber is assimilated as an incompressible non-Newtonian fluid flow :



The governing system of PDEs :

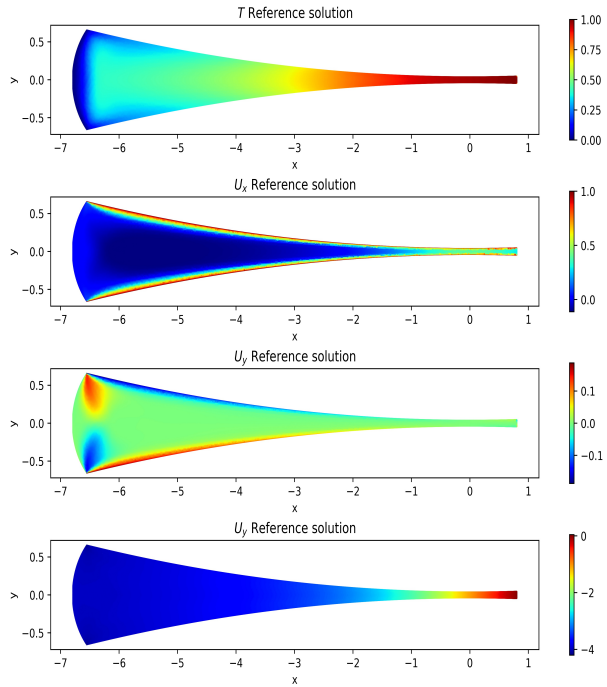
$$\nabla.(2\eta(\vec{\boldsymbol{u}}, T)\bar{\bar{\epsilon}}(\vec{\boldsymbol{u}})) - \vec{\nabla}p = \vec{0}$$
$$\nabla.\vec{\boldsymbol{u}} = 0$$
$$\vec{\boldsymbol{u}}.\vec{\nabla}T = \frac{\lambda}{\rho C_p}\nabla^2 T + \frac{1}{\rho C_p}\eta(\vec{\boldsymbol{u}}, T)|\gamma(\vec{\boldsymbol{u}})|^2$$

Parameters : $\vec{\boldsymbol{u}} = (u_x, u_y)^T$ is velocity vector, p pressure, T temperature.
Conditions : $\lambda = 20\ W.m^{-1}.K^{-1}$ thermal conductivity, $\rho = 1000\ kg.m^{-3}$ rubber density, $C_p = 1000\ J.K^{-1}$ heat capacity at 25°C.

Physics models

Example. A more complex industrial model : the Rubber calendering modeling. Numerical solutions computed thanks to a private Software

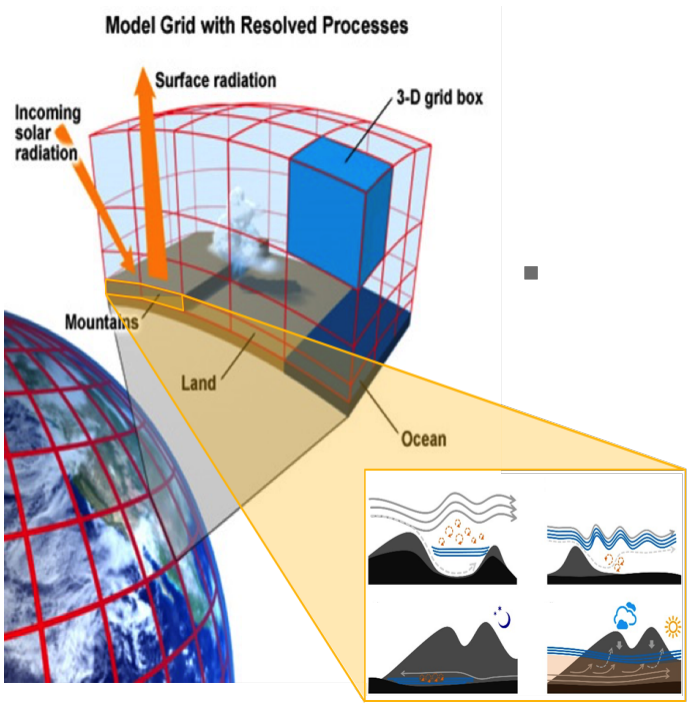
To generate reference High-Fidelity (HF) solutions of temperature (1) velocity (2,3), and pressure (4), fields,
the equations are discretized and solved
using an in-house generic and multi-purpose finite element solver named MEF++
co-developed by Laval University and Michelin..



Physics models

Example. A more complex Meteorological model around a montain

Physical parameterizations play a key role in modeling important physical processes in numerical weather and climate prediction models.



The parameterization as PDEs

Linearization around a BL flow $u_0(z) = \mathrm{D}\tanh(z/\mathrm{D})$ yields the 2D Boussinesq **dimensionless** equations:

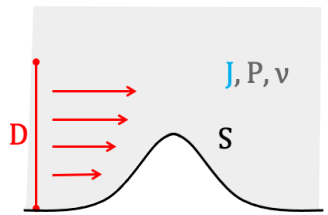
$$u_0\partial_x u + u_{0z} w = -\partial_x p + \nu\partial_z^2 u$$
$$u_0\partial_x w = -\partial_z p + b + \nu\partial_z^2 w$$
$$u_0\partial_x b + \mathrm{J} w = \mathrm{P}^{-1}\nu\partial_z^2 b$$
$$\partial_x u + \partial_z w = 0$$

where u, w, p and b are the horizontal and vertical velocity, the pressure and the buoyancy, respectively.
Boundary conditions (BC) at ground level are given by:

$$h(x) + u(x,h) = w(x,h) = \mathrm{J}h(x) + b(x,h) = 0$$

Dimensionless parameters

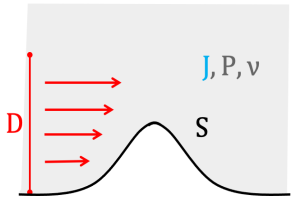
	Dimensionless number
J	Richardson number
P	Prandlt number
S	Slope parameter
D	Boundary layer depth
ν	Inverse Reynolds number



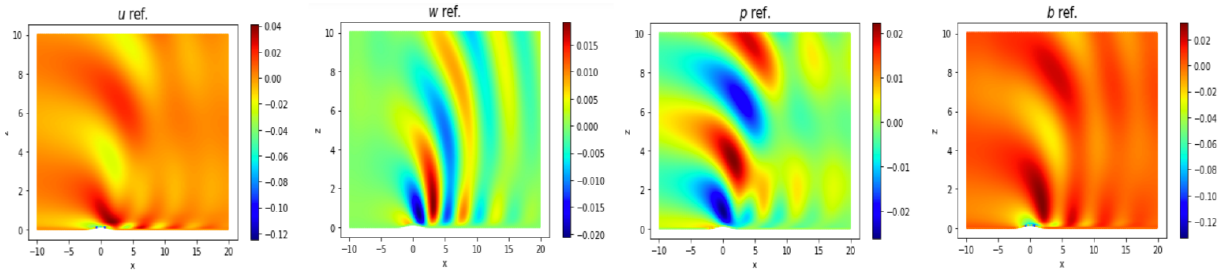
Physics models.

Example. A more complex Meteorological model around a mountain.

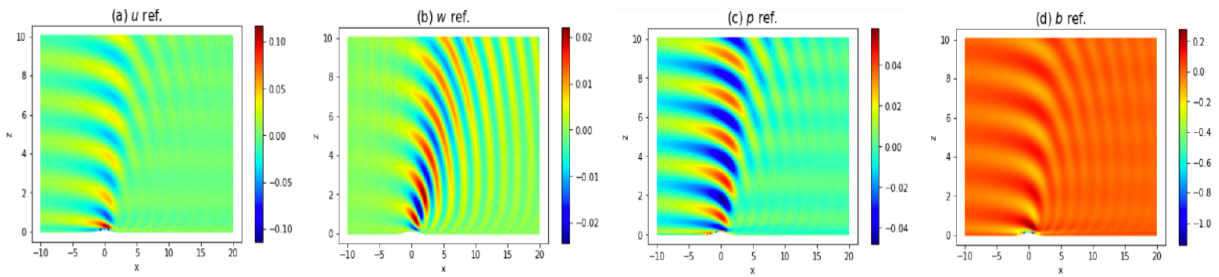
Example of Meteorological model for a given mountain height
Numerical solutions provided by CEA & LMD .



Numerical solution $J = 1$



Numerical solution $J = 9$



[Nguyen *et al.*, 2023b]

Physics models & Numerical methods

undergoing issues

- Design costs : physics modeling
- Discretization of the domain/ meshing ...
- Implementation , support, evolution, cost...
- Computation cost : simulation time may be prohibitive, Experimental design to choice smart configurations
- Generalization, approximation issues : convergence of the solver, choice of parameters, downscaling...

Surrogate models

Machine Learning models for physics

Surrogate models

Surrogate/ Meta models approximate the input/output relation.
Several techniques have been developed such as :

- Reduced Order Models (ROM) which reduce the order of the model in an unsupervised manner like Principal Component Analysis (PCA).
- Data fit models which create a fit between input and output models based on simulation data as for example polynomial basis, radial basis function, Gaussian Process, stochastic polynomial chaos expansion.
- Machine learning, Deep Neural Networks models which are known to need large data set.

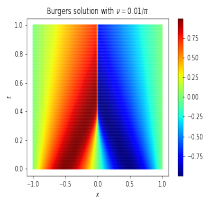
The supervised data driven approach :

1. Data set provided by experimental design or sampling.
   $\mathcal{D}_n = \{(x_i, y_i),\ 1 \leq i \leq n,\ \text{input } x_i \in \mathcal{X},\ \text{output} y_i \in \mathcal{Y}\}$
2. The Model $f_\theta : f_\theta(x) \to y$
   Parametric model : Gaussian Process....
   Non Parametric models : NN, RF, GradBoost...
3. Calibration by optimisation given the data $\mathcal{D}_n$
   Example : ℓ_2 Loss function $\mathcal{L}_{\text{data}} = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} (f_\theta(x_i) - y_i))^2$

ML surrogate model

Illustration with Deep Neural Network on a toy example.

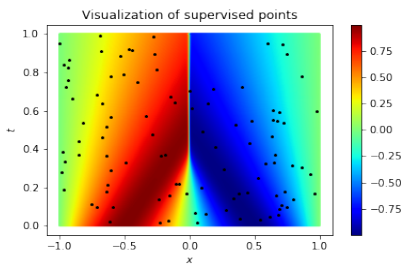
Burger's equation :input (x space , t *time*), output : speed of the fluid given parameter : ν viscosity. Dirichlet boundary condition : $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0; u(0,x) = -sin(\pi x); u(t,-1) = u(t,1) = 0$



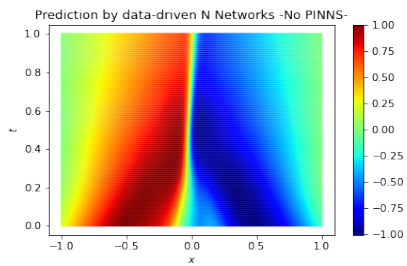
Neural network Surrogate model.

f_θ trained with $n = 100$ supervised observations. NN architecture : 2-4 (x50)-1.
Evaluation on a Test data set, regular grid of points (256 (x) , 100 (t)).
$E(u,\hat{u}) = ||u-\hat{u}||_2^2/||u||_2^2$ on a grid (256 (x) , 100 (t))

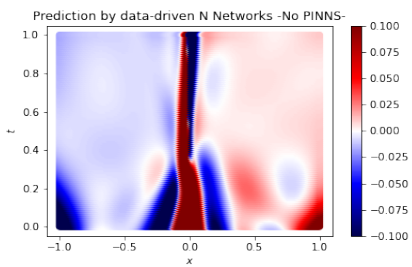
Supervised points



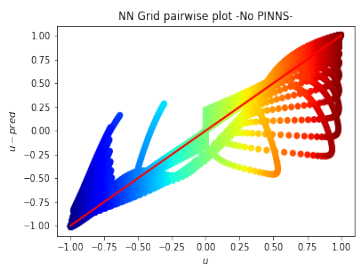
Surrogate model $E(u,\hat{u}) = 0.17$



Error



Pairwise Plot

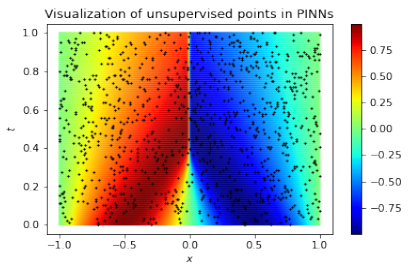


ML Surrogate model

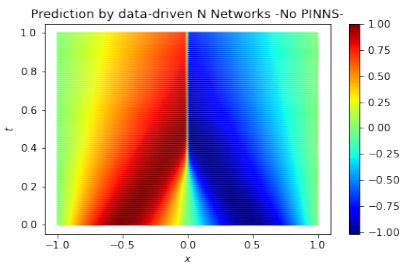
Illustration with Deep Neural Network on a toy example.

$n = 1000$ supervised observations.
Model f_θ : neural network. NN architecture : 2-4 (x50)-1.
Evaluation on a Test data set, regular grid of points (256 (x) , 100 (t)).

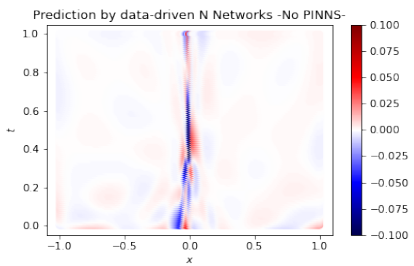
Supervised points



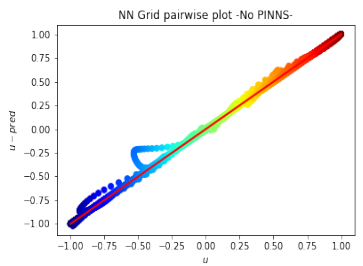
Surrogate model $E(y, \hat{y}) = 0.02$



Error

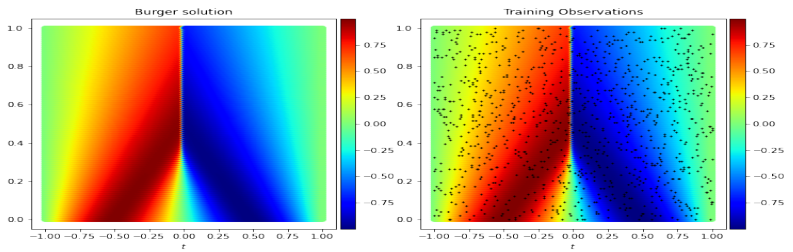


Pairwise Plot

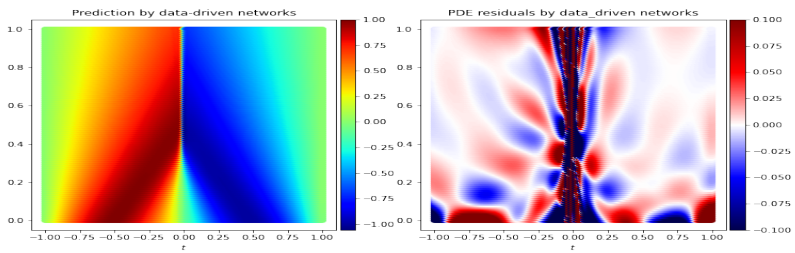


ML Surrogate model

Illustration with Deep Neural Network on a toy example.
Focus on the Pde errors/residuals computed for Burger model (left/bottom figure) :
$N_{\text{data}} = 1000$, $N_{\text{colloc}=0}$, NN architecture : 2-4 (x50)-1. 50 000 epoch, Adam optimizer.
$E(u, \hat{u}) = ||u - \hat{u}||_2^2 / ||u||_2^2$ on a grid (256 (x) , 100 (t))



PINNs model $E(u, \hat{u}) = 0.0163$



Conclusions :
$\rightarrow$The pde constraints are not respected...
$\rightarrow$ The NN model mimics the input/output relation but the underlaying physics is not caught.

ML Surrogate model

First conclusions :

- Especially, in a "small data regime", the vast majority of state-of-the-art machine learning techniques are lacking robustness and fail to "model" the underlaying physics phenomena (pde constraints not respected).
- The cost of data acquisition may be prohibitive.
  Experimental design are proposed to chose the observations.
- We are inevitably faced with the challenge of drawings conclusions and making decision under partial information.

Outline

Physics Informed models

Physical Informed Neural Networks

Introduced by Raissi et al. in 2019 [Raissi *et al.*, 2019, Cuomo *et al.*, 2022]
Physics-informed neural networks-PINNs- are neural networks that are trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear partial differential equations.

Key objectives of Physics-informed Machine Learning :

- Mitigate the shortage of training data
- Ensure the physical plausibility of results
- Increase models' generality

[Karniadakis *et al.*, 2021], [Meng *et al.*, 2022], [Hao *et al.*, 2022]...
[Raissi *et al.*, 2019] claim to solve to main class of problem **data-driven solution** and **data-driven discovery of PDE**.

Physical Informed Neural Netwotks

Since 2019, many applications have been investigating with Pinns, a collection of classical problems in fluid, quantum mechanics, reaction-diffusion systems, propagation of nonlinear shallow-water waves...

FIGURE – Various equations have been investigated since 2017. Shrodinger problems -SE- , Euler equations -EE-, Navier-Stokes equations -NSE-; Heat flow convections -HE-.Credit .... [Cuomo *et al.*, 2022]

Physical Informed Neural Netwotks

Physics-informed neural networks- are neural networks that are trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear partial differential equations
Introduced by Raissi et al. in 2019 [Raissi *et al.*, 2019].

Key ingredients :

- A deep neural network **model** : $h_\theta \in \mathcal{H}$
- A **supervised data set** of input (x) /output (y) observations $\mathcal{D}_n = \{(x_i, y_i)\ 1 \leq i \leq n,\ x \in \mathcal{X},\ y \in \mathcal{Y},\}$
- **Equations*** describing the physics (physical parameters, physical constants...)
- **A appropriate loss function** $\mathcal{L}(h_\theta, \mathcal{D}_n,$ physics equations$)$.

With this approach [Raissi *et al.*, 2019] claimed to solve to main class of **data-driven solution** problem and **data-driven discovery of PDE**.

(*) many physics have been investigated for forward, inverse, ill problems....

PINNS framework

PINNs [Raissi *et al.*, 2019] can address problems that are described by few data or noisy experiment observations and *solve* differential equations expressed as :

- $\mathcal{F}(u(z), \gamma) = f(z) \qquad z \in \Omega$
- $\mathcal{B}(u(z)) = g(z) \qquad z \in \partial\Omega$

Notations :

- $z := [x_1, x_2, \ldots, x_{d-1}, t]$ the space-time coordinate vector
- $u()$ the unknown solution in the domain $\Omega \subset \mathbb{R}^d$ with $\partial\Omega$ boundary.
- γ the parameters related to the context physics
- $f()$ the function identifying the data on the problem
- $g()$ the boundary function.
- $\mathcal{F}$ the non linear differential operator.
- $\mathcal{B}$ the operator indicating arbitrary or boundary conditions to the problems

This framework addresses

- forward - finding $u()$ for every z where γ are specified parameters-
- as *inverse* problems when γ should be also computed from the data.

Physical Informed Neural Networks

Thanks to an optimization/ training procedure, the NN learns to approximate the differential equations through finding θ that defined the *trained NN* by minimizing a loss function that depends on :

- some known data ($\mathcal{L}_{\text{data}}$),
- the boundary conditions ($\mathcal{L}_{\mathcal{B}}$), and
- the differential equation ($\mathcal{L}_{\mathcal{F}}$).

$$\theta^* = \underset{\theta}{\text{argmin}}(w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}} + w_{\mathcal{B}}\mathcal{L}_{\mathcal{B}} + w_{\text{data}}\mathcal{L}_{\text{data}})$$

PINNS framework

In the PINN framework $u(z)$ is computed thanks to a (deep)- NN, parameterized by a set of parameters θ, giving rise to an approximation : $\hat{u}_\theta(z) \sim u(z)$. where $\hat{u}_\theta$ denotes the NN approximation with θ parameter.

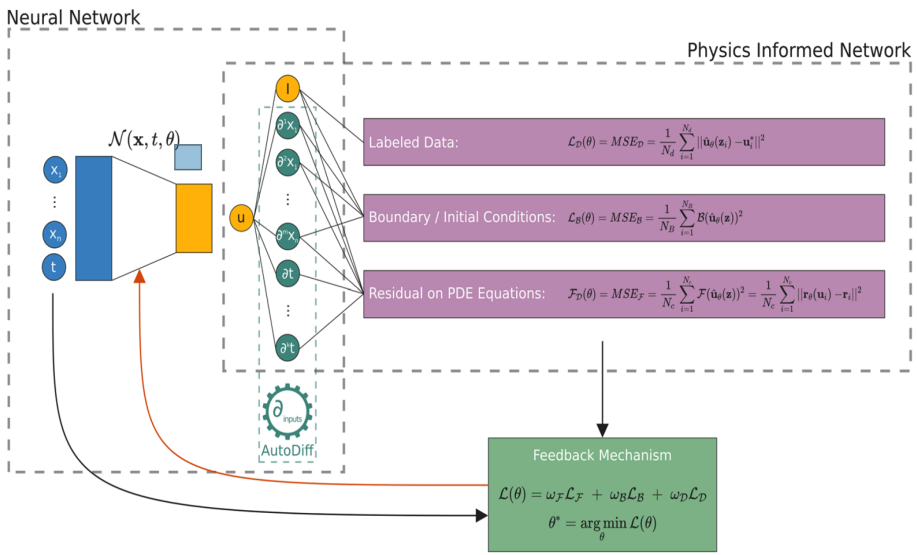


FIGURE – Credit [Cuomo *et al.*, 2022]

Note : the same framework let to analyse forward, inverse problems ($\theta = (\theta_{NN}, \gamma)$)

Physical Informed Neural Netwotks

For PINNs models, the loss function $\mathcal{L}()$ contains physics information described by partial differential equation, tailored to modeled underlaying differential operator. The loss function is a weighted sum of the two losses given

$$\mathcal{L}(\theta) = w_{\text{data}}\mathcal{L}_{\text{data}} + w_{\text{pde}}\mathcal{L}_{\text{pde}}$$

- Supervised (domain, boundary) data which needs label (input/output)
  $\mathcal{L}_{\text{data}}(\theta, \mathcal{D}_{\text{data}}) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} (f_\theta(x_i) - y_i))^2$
  f_θ is the NN chosen model, with θ parameter.

- Unsupervized or "cheap" collocation points to enforce the Physical constrains
  $\mathcal{L}_{\text{pde}}(\theta, \mathcal{D}_{\text{colloc}}) = \frac{1}{N_{\text{pde}}} \sum_{i=1}^{N_{\text{pde}}} (R^{\text{pde}}(i))^2$

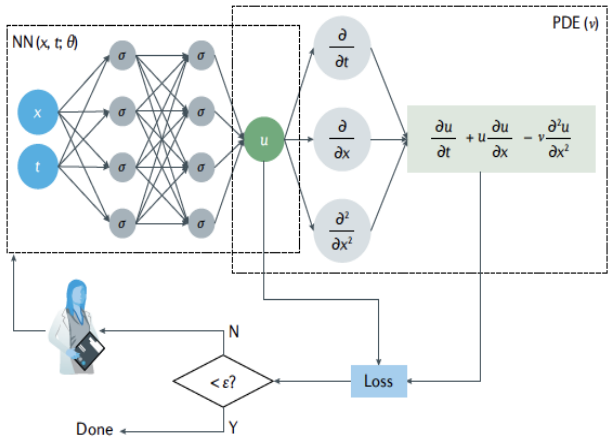
  R_{pde} are the residuals computed based on the pde information.
- The loss functions are optimized using gradient-based optimization algorithm (full batch or stochastic depending of the size of the data set).
  w_{data}, w_{pde} weights to balance the two losses.

Physical Informed Neural Netwotks

[Raissi *et al.*, 2019]

$f_\theta(t,x)$ is defined by : $f_\theta := u_t + \mathcal{N}[u]$

- $u(t,x)$ is approximated by a deep NN.
- Automatic differentiation helps to minimize $R(t,x)$ applying the chain rule for differentiating compositions of functions.



The shared parameters θ between the NN $f_\theta(t,x)$ and $R_\theta^{\text{pde}}(t,x)$ can be learned by minimizing the mean squared error.

$$\text{MSE} = w_{\text{data}}\text{MSE}_{\text{data}} + w_{\text{pde}}\text{MSE}_{\text{pde}}$$

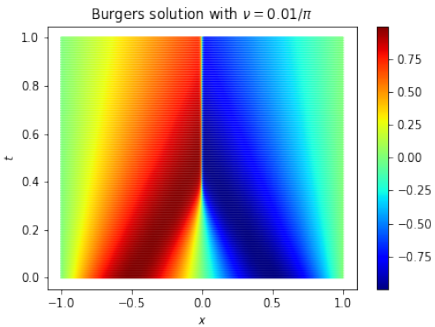
- $\text{MSE}_{\text{data}} = \frac{1}{N_{\text{data}}}\sum_{i=1}^{N_{\text{data}}}[f_\theta(t_u^i, x_u^i) - u^i]^2$ for chosen training data (initial and boundary).
- MSE_{pde} = is the cost of pde residuals on random collocation points. No mesh!

PINNS model on a toy model

Illustration. Seminal Burger's equation

in one dimension (x), in time t with the Dirichlet boundary condition (ν fixed) :

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0$$
$$u(0, x) = -sin(\pi x)$$
$$u(t, -1) = u(t, 1) = 0$$



Let supposed known the parametrized, nonlinear partial differential equations (pde) :

$$u_t + \mathcal{N}[u], \ x \in \Omega, t \in [0, T]$$

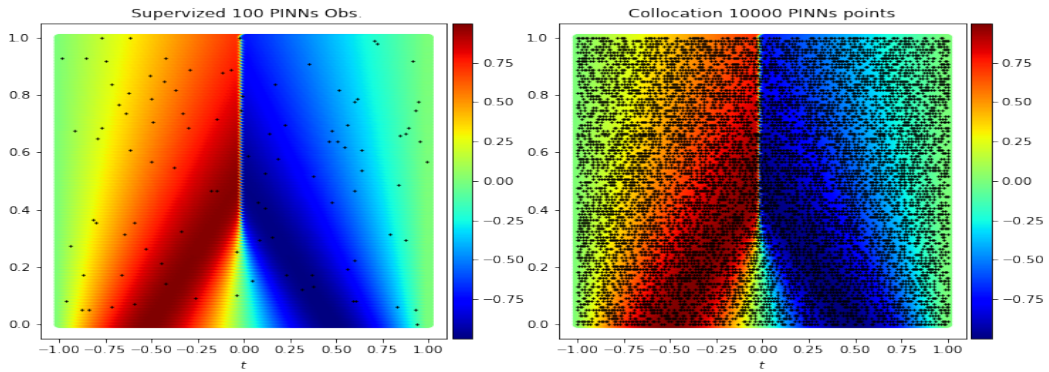
$u(t, x)$ denotes the latent solution,
$\mathcal{N}[.]$ a non linear differential operator, Ω a subset of $\mathbb{R}^p$

Introduced by Raissi et al. in 2019 [Raissi *et al.*, 2019].

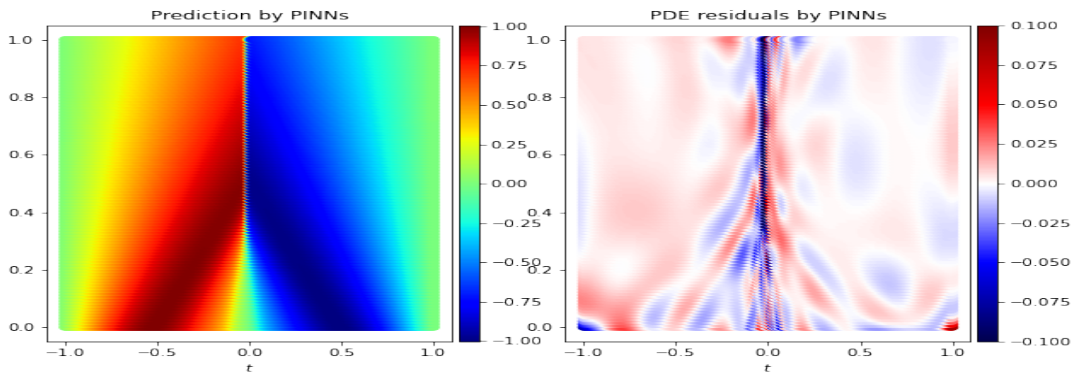
PINNS model on a toy model

Importance of the Training observations (only in the domain) and impact on the Pde errors computed for Burger model :

$N_{\text{data}} = 100$, $N_{\text{colloc}=10000}$, NN architecture : 2-4 (x50)-1. 50 000 epoch, Adam optimizer.



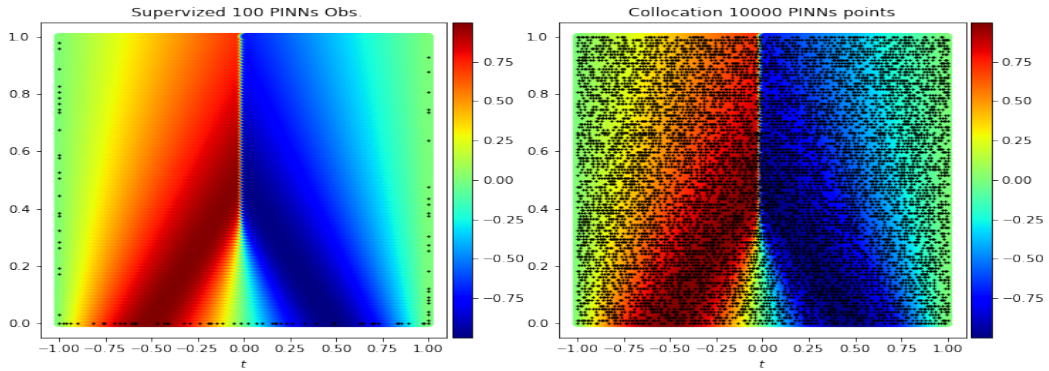
PINNs model $E(u, \hat{u}) = 0.0157$



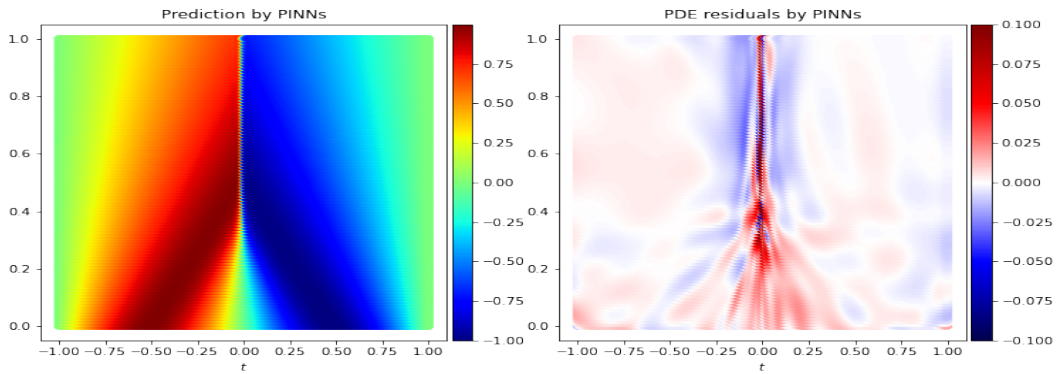
PINNS model on a toy model

Supervised data only for boundary conditions :
Illustration of the pde errors after training :

$N_{\text{data}} = 100$, $N_{\text{colloc}=10000}$, NN architecture : 2-4 (x50)-1. 50 000 epoch, Adam optimizer.



PINNs model $E(u, \hat{u}) = 0.05$

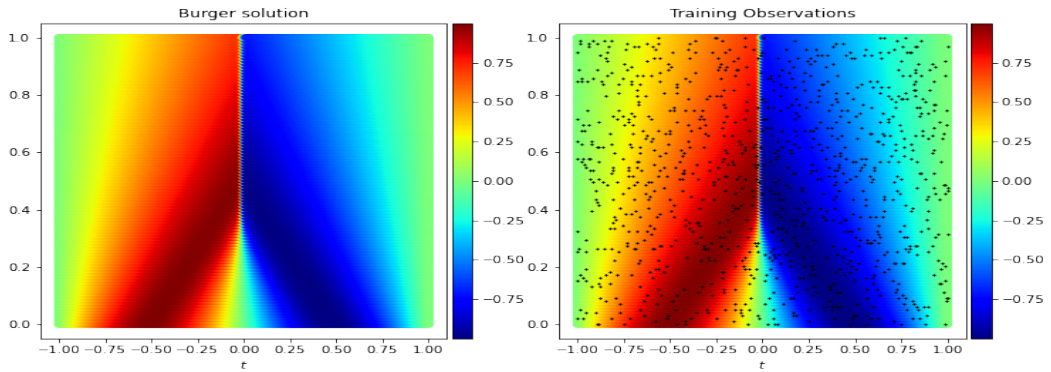


$E(u, \hat{u}) = ||u - \hat{u}||_2^2 / ||u||_2^2$ on a grid (256 (x) , 100 (t))

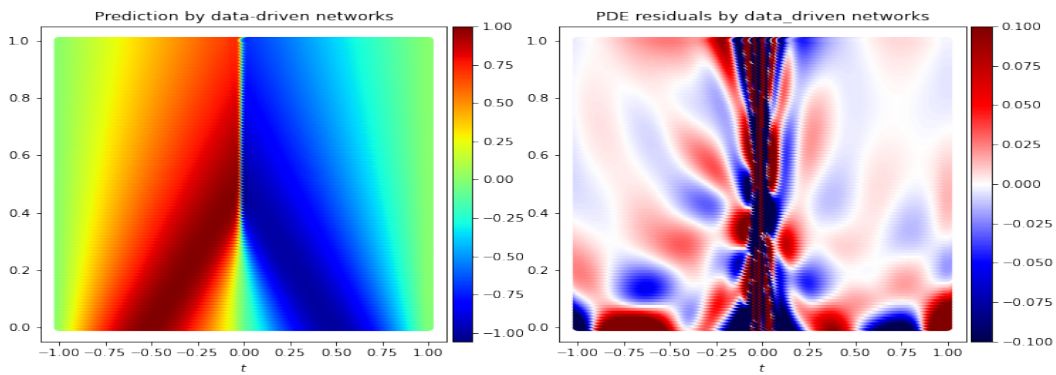
Back to a Surrogate model on a toy model

Pde errors computed for Burger model :

$N_{\text{data}} = 1000$, $N_{\text{colloc}=0}$, NN architecture : 2-4 (x50)-1. 50 000 epoch, Adam optimizer.



PINNs model $E(u, \hat{u}) = 0.0163$



$E(u, \hat{u}) = ||u - \hat{u}||_2^2 / ||u||_2^2$ on a grid (256 (x) , 100 (t))

Physical Informed Neural Networks issues

Open questions which appeared to have an important impact for PINNs models...

- **data** *Supervised data.* Observations (number, localisation, boundary or inside the domain...); *Unsupervised data.* Collocation points : number, localization, choice at random, on a mesh grid ...
- **Loss** Trade-off between the two data and residual pde losses :
  $\text{MSE} = w_{\text{data}}\text{MSE}_{\text{data}} + w_{\text{pde}}\text{MSE}_{\text{pde}}$
- **NN** Choice of architecture (full connected, deep vs shallow, convolutional networks), activation functions, ...
- **Optimisation** ...
- PDE with given parameters (ν) are easily learned by PINNs.
  What about parameterized PDE ?
- Impact of the geometry of the design. ($\rightarrow$ GAPINNs)

Industrial PINNs Applications

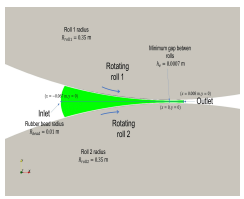
- Surrogate modeling. Data-driven solution of pde. Fluid mechanics [Raissi *et al.*, 2019],... , Solid mechanics,...
  Application to rubber calendering process (non-Newtonian fluid thermo-mechanical problems) [Nguyen *et al.*, 2022]
- Identification of unknown parameters. Data discovery of pde.
  [Raissi *et al.*, 2019], [Nguyen *et al.*, 2022]
- ill-posed problems.

Outline

The rubber calendering process

Tire manufacturing application : the Rubber calendering modeling.
Calendering is a mechanical process used to create and assemble thin tissues of rubber. From the physical point of view, the rubber is assimilated as an incompressible non-Newtonian fluid flow. [Nguyen *et al.*, 2022].

The quantities e_1, e_2, e_3, e_4 correspond to the dimensionless PDEs residuals defined in the Physical equations.



$$e_1 = \frac{\partial}{\partial x}\left(2\eta(\vec{\hat{\boldsymbol{u}}}, \hat{T})\frac{\partial \hat{u}_x}{\partial x}\right) + \frac{\partial}{\partial y}\left(\eta(\vec{\hat{\boldsymbol{u}}}, \hat{T})(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x})\right) - \frac{\partial \hat{p}}{\partial x}$$

$$e_2 = \frac{\partial}{\partial x}\left(\eta(\vec{\hat{\boldsymbol{u}}}, \hat{T})(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x})\right) + \frac{\partial}{\partial y}\left(2\eta(\vec{\hat{\boldsymbol{u}}}, \hat{T})\frac{\partial \hat{u}_y}{\partial y}\right) - \frac{\partial \hat{p}}{\partial y}$$

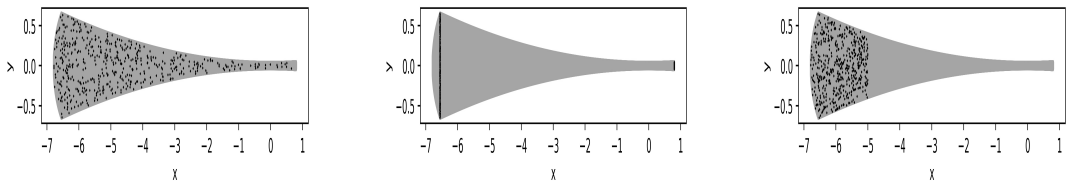
$$e_3 = \frac{\partial \hat{u}_x}{\partial x} + \frac{\partial \hat{u}_y}{\partial y}$$

$$e_4 = \hat{u}_x\frac{\partial \hat{T}}{\partial x} + \hat{u}_y\frac{\partial \hat{T}}{\partial y} - \frac{1}{Pe}(\frac{\partial^2 \hat{T}}{\partial x^2} + \frac{\partial^2 \hat{T}}{\partial y^2}) - \frac{Br}{Pe}\eta(\vec{\hat{\boldsymbol{u}}}, \hat{T})|\gamma(\vec{\hat{\boldsymbol{u}}})|^2$$

where $\vec{\boldsymbol{u}} = (u_x, u_y)^T$ velocity vector; p is the pressure; T temperature, $\lambda = 20\ W.m^{-1}.K^{-1}$ thermal conductivity, $\rho = 1000\ kg.m^{-3}$ rubber density, $C_p = 1000\ J.K^{-1}$ is heat capacity at 25°C.

Inferring physical fields of interest from sensor measurements

The goal is to infer the pressure, velocity, and temperature fields at all points in the domain given only some measurements of the temperature available from virtual sensors.



case 1 case 2 case 3

Location of sensors for the geometry of the rubber calender

PINNS Cost Functions [Nguyen *et al.*, 2022] :

$L = L_{data} + L_{pde}$.

$$L = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \omega_T (\hat{T}^i - T^{i*})^2 + \frac{1}{N_{\text{colloc}}} \sum_{i=1}^{N_{\text{colloc}}} (\omega_1 e_1^2 + \omega_2 e_2^2 + \omega_3 e_3^2 + \omega_4 e_4^2)$$

$\hat{T}$ are the predictions obtained by the neural network and T^* are the supervised sensor data for the temperature. The quantities e_1, e_2, e_3, e_4 correspond to the dimensionless PDEs residuals defined in the equations.

No data for the velocity and the pressure in the cost function except their relation with the temperature given by the system of PDEs.

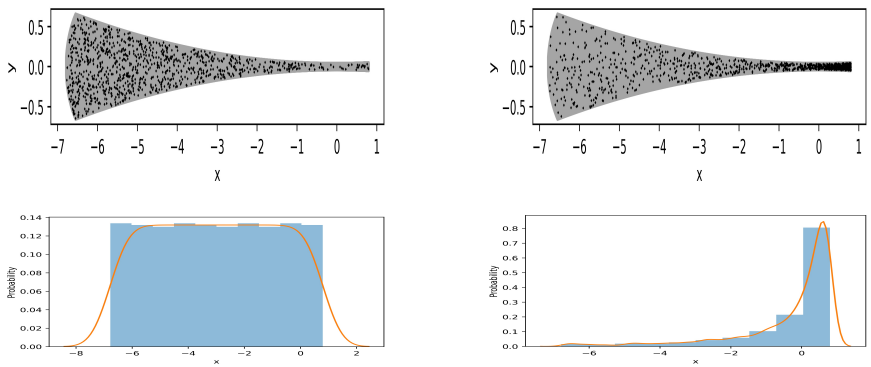
The boundary conditions of the problem are supposed to be not completely defined.

No information on the pressure field is given, but only its gradient in the PDE residual is. (pressure only identifiable up to a constant).

Inferring physical fields of interest from sensor measurements

- Sampling collocation point issues

Mesh grid (1, 000 random points) compared to 1, 000 randomly uniformly distributed points inside the domain.



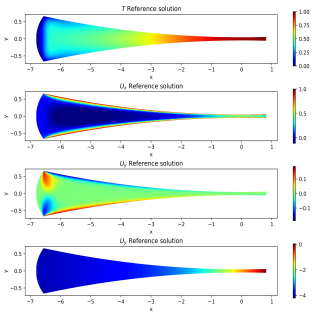
random mesh (case 1) finite element mesh (case 2)

- Knowledge brings by the mesh grid

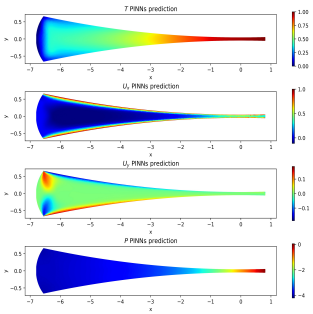
We point out that when the points are taken from the finite element mesh, the density of points is higher at the output than at the input of the calender. This is because the finite element mesh is refined to take into account the very thin gap between the rolls close to the domain outlet.

Inferring physical fields of interest from sensor measurements

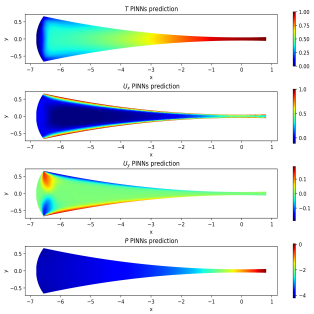
- Reference solution and PINNs prediction when using in additional the boundary condition for the velocity and the temperature as learning data.



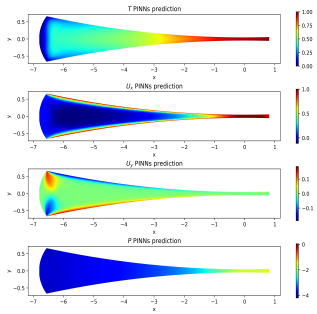
Reference solution



PINNs prediction : case 1



PINNs prediction : case 2



PINNs prediction : case 3

- PDE residuals

Error	ϵ_T (1)	(2)	ϵ_{u_x} (1)	(2)	ϵ_{u_y} (1)	(2)	ϵ_p (1)	(2)
Case 1	14.3	**7.86**	17.8	**2.56**	5.59	**5.41**	18.9	**0.29**
Case 2	13.9	**10.1**	49.9	**6.90**	10.3	**5.04**	0.95	**0.39**
Case 3	14.8	**5.83**	125	107	5.76	**5.44**	55.1	47.5

- PINNs are only able to approximate accurately the temperature in the zones where there are sensors and fail to predict the velocity and pressure.
- Providing PINNs information on the roll boundaries only and velocity let to good solutions.

Inverse problems

Identifying some unknown physical parameters. In the industrial context, it is often the case that some physical parameters are not defined but we observe some measurements of the solution.

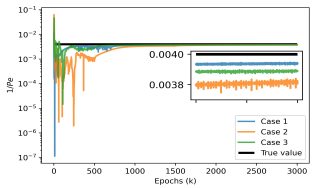
Application to the rubber calendering process.
Use of PINNs to identify an unknown physical parameter, the thermal conductivity of the rubber λ from some observed measurements.
use of PINNs to define the two unknown parameters (Péclet and Brinkman number) in the dimensionless PDE and then deduce the value of the conductivity.

No information on the boundary is given.

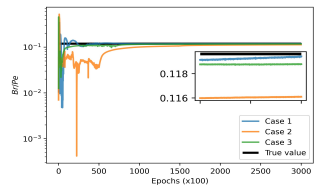
Inverse problems

Identifying some unknown physical parameters.

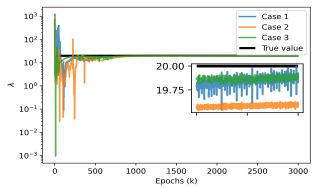
• Prediction for unknown parameters during the training process while considering different cases of the location of sensors.
The small plots show the results at the last 100,000 epochs.



$1/Pe$



Br/Pe



λ

•Prediction for unknown parameters at the end of PINNs training process.

	$1/Pe$	Br/Pe	λ
High-fidelity model	4.00e-03	1.20e-01	2.00e+01
Case 1	3.94e-03	1.20e-01	1.99e+01
Case 2	3.81e-03	1.16e-01	1.96e+01
Case 3	3.89e-03	1.19e-01	1.98e+01

In the calendering application, PINNs succeeds of identifying an unknown physical parameter (namely the thermal conductivity of the rubber λ) from some observed measurements.

Outline

Adaptive Sampling strategies

- As PINNs integrate the PDEs constraints by minimizing the PDE residuals on a set of collocation points during the training process, it has been shown that the location of these collocation points has a great impact on the performance of PINNs [Daw *et al.*, 2022].
- Lu et al. (2021) proposed a first work that showed the improvement of PINNs performance by modifying the set of collocation points [Lu *et al.*, 2021].
  RAR. The Residual-based Adaptive Refinement (RAR) proposes to add new training collocation points to the location where the PDE residual errors are large. RAR has been proven to be very efficient to increase the accuracy of the prediction but however leads to an uncontrollable amount of collocation points and computational cost at the end of the training process.
- **RAD and RAR-D** proposed by [Wu *et al.*, 2023]. For the Residual-based Adaptive Distribution (RAD), All the training collocation points are randomly sampled according to a probability density function which is based on the PDE residuals.
  For the Residual-based Adaptive Refinement with Distribution (RAR-D) only a few new points are sampled and then added to the training data set according to a probability density function which is based on the PDE residuals.

Adaptive Sampling strategies

[Nguyen *et al.*, 2023a] proposes the
Fixed-Budget Online Adaptive Learning (FBOAL) method :

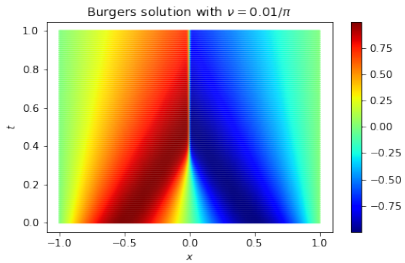
- to **control** the number of training points potentially added by the method
- to **control** the budget of collocation points .
- to consider a **set of sub-domains** in order to capture not only the global extrema but also the local extrema of the PDEs residuals.

The added points tend to be placed at nearly the same location corresponding to the global maximum value of the PDE residual.

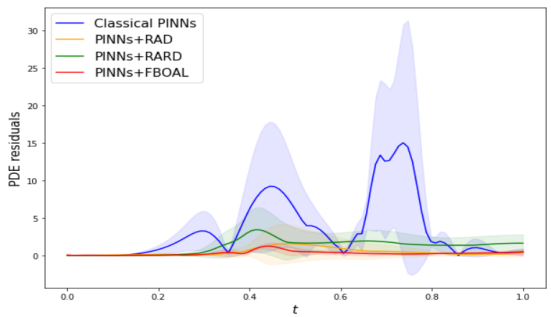
Adaptive Sampling Strategies for PINNS

help to decrease the PDE residuals.

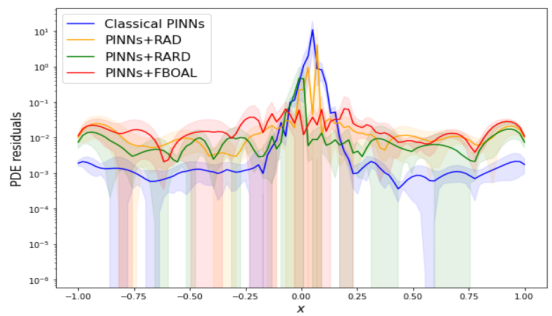
Illustration for Burgers equation :
$u_t + uu_x - \nu u_{xx} = 0 \quad \text{for } x \in [-1, 1], t \in [0, 1]$;
$u(x, 0) = -\sin(\pi x)$
$u(-1, t) = u(1, t) = 0.$



Comparison of vanilla PINNS with several adaptive strategies (RAD, RAR-D, FBOAL)



On the line $x = 0$



At instant $t = 1$

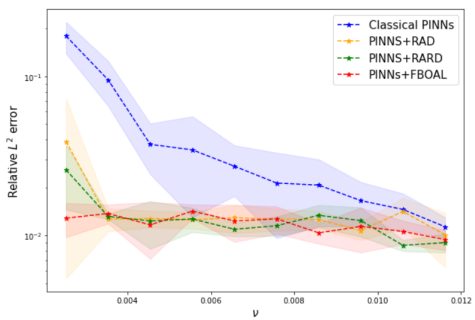
FIGURE – Burgers equation : Absolute value of PDE residuals for $\nu = 0.0025$ after the training for different approaches.

Adaptive Sampling Strategies for PINNS

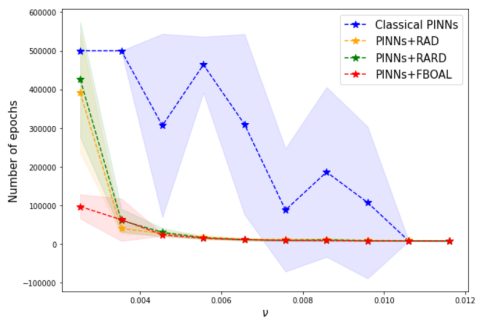
help to decrease the Global PDE residual Error and Training Time.

Comparison of vanilla PINNS with several adaptive strategies (RAD, RAR-D, FBOAL)
To assess the overall performance of PINNs, the errors of the prediction are evaluated on a 256×100 spatio-temporal mesh (validation mesh).

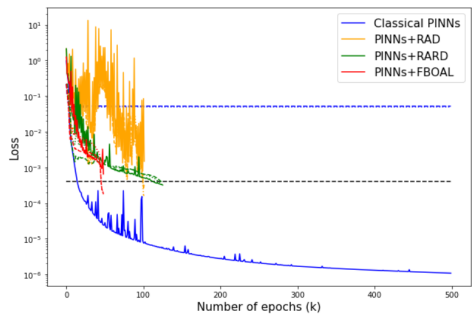
- Performances



Relative $\mathcal{L}^2$ error



Nb of training iterations



Loss for $\nu = 0.0025$

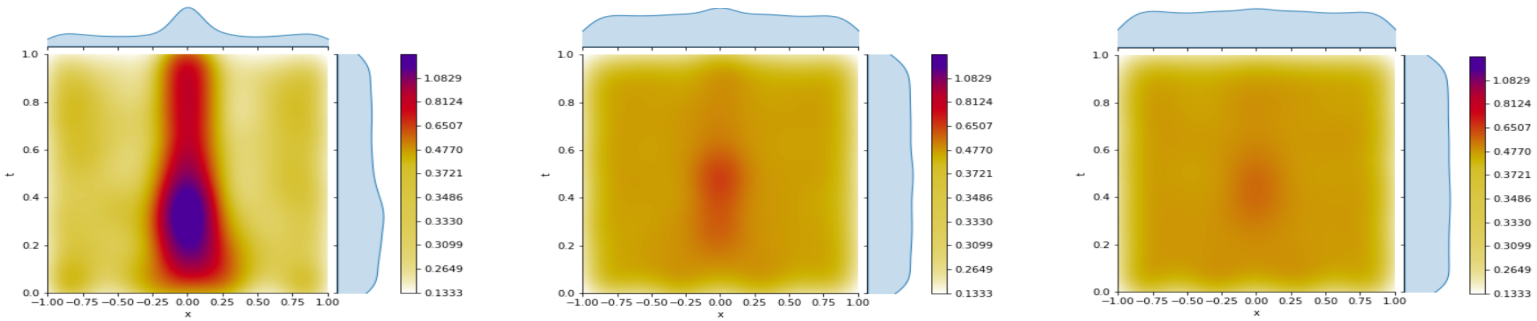
- Training time

	Classical	RAR-D	RAD	FBOAL
Training time (minutes)	33.7 ± 1.5	41.0 ± 5.8	38.8 ± 2.3	$\mathbf{21.5 \pm 2.7}$
Number of resampling	0 ± 0	201 ± 75	210 ± 77	$\mathbf{48 \pm 2}$

TABLE – Burgers equation : Training time and the number of resampling for $\nu = 0.0025$. The training is effectuated on an NVIDIA V100 GPU card.

Adaptive Sampling Strategies for PINNS

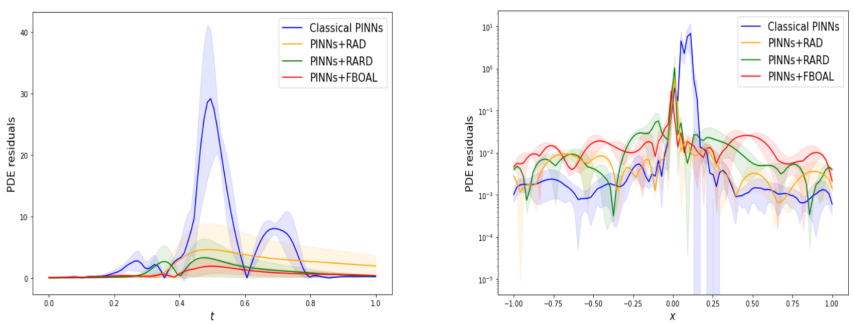
provide an adaptive collocation point "mesh".



$\nu = 0.0025$ $\nu = 0.0076$ $\nu = 0.0116$

FIGURE – Burgers equation. Density of collocation points after FBOAL training. One PINNS model for each ν *value*

Absolute value of PDE residuals after training process by different approaches



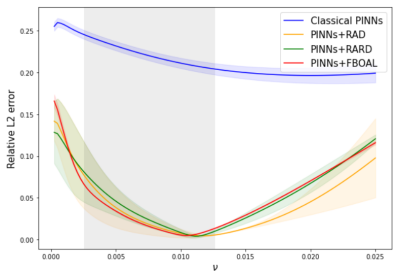
On the line $x = 0$ At instant $t = 1$

FIGURE – Burgers equation $\nu = 0.0025$.

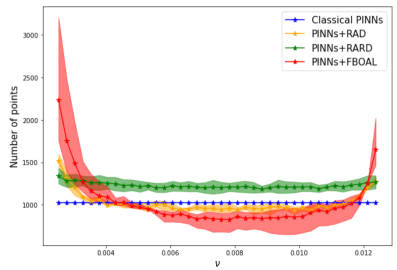
Adaptive Sampling Strategies for PINNS

automatically provide an adaptive collocation point "mesh" given the ν parameter.

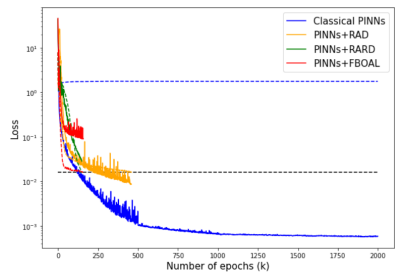
Illustration : *ParametricPINNS* - ν input of the NN for Burger equation *bullet*



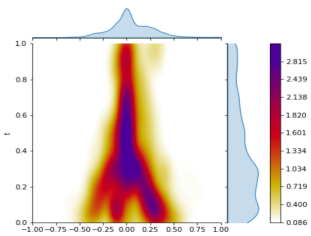
Relative $\mathcal{L}^2$ error



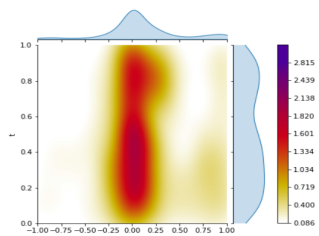
Nb of collocation points



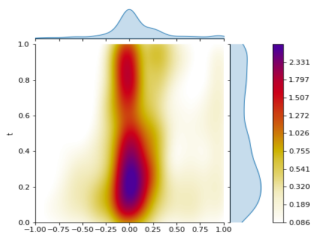
Training cost function



$\nu = 0.0025$



$\nu = 0.0076$



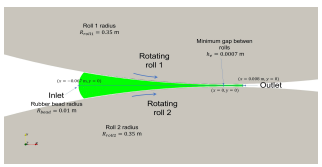
$\nu = 0.0116$

- Training time

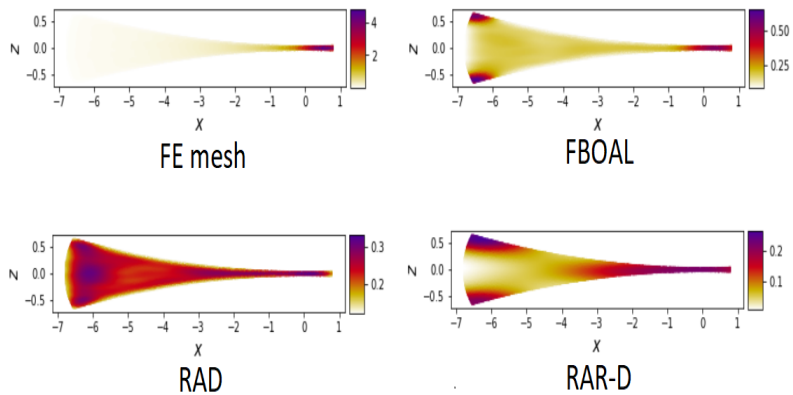
	Classical	RAR-D	RAD	FBOAL
Training time (hours)	13.2 ± 0.0	$\mathbf{1.4 \pm 0.3}$	9.1 ± 3.5	7.8 ± 1.9
Number of resampling	0 ± 0	$\mathbf{34 \pm 8}$	204 ± 71	173 ± 25

Adaptive Sampling Strategies for PINNS

Application to the calendering process.
The Final density of collocation points is related to the region with the highest gradients (with no supervised points in the domain.)



Geometrical setup



Density of collocation points by different methods

→ Sampling on a previous Expert mesh is valuable!

	ϵ_T	ϵ_{u_x}	ϵ_{u_y}	ϵ_p
Random points	13.6 ± 1.17	24.1 ± 3.77	15.6 ± 2.73	11.9 ± 1.33
FE mesh	**8.54 ± 1.39**	**14.7 ± 2.49**	18.0 ± 2.61	**1.41 ± 0.17**
FBOAL	10.5 ± 1.82	19.3 ± 1.85	**9.63 ± 3.24**	5.13 ± 0.72
RAD	10.2 ± 1.41	20.5 ± 1.96	**9.71 ± 2.55**	5.08 ± 0.76
RAR-D	12.3 ± 1.74	18.7 ± 1.65	11.1 ± 2.90	4.87 ± 0.69

TABLE – Relative $\mathcal{L}^2$ errors between reference solution and PINNs prediction with different cases of collocation points.

Geometry aware physics informed

[Oldenburg *et al.*, 2022] introduced Geometry aware physics informed neural network surrogate (GAPINN) to generate surrogates which may take into account different domain geometry boundaries.

The GAPINN framework involves three network types.

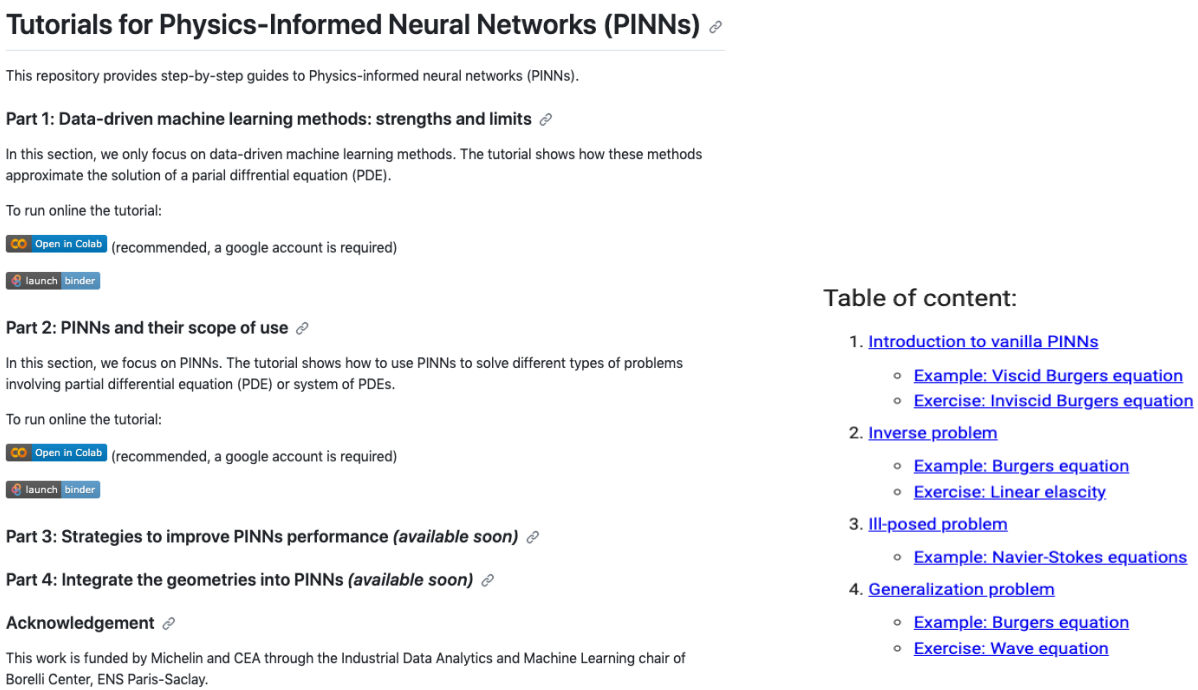
1. The first network reduces the dimensions of the irregular geometries to a latent representation, by using Variational AutoEncodeur (VAE)
2. The previous latent representation is combined with the spatial coordinates define PINNs input.
3. Using PINNS framework, a surrogate model is trained by the minimisation of the residuals of the underlaying PDE for irregular non-parametric geometries.

Outline

Softwares & Tutorials

- Several Softwares packages
  DeepXDE, PyDEns, NeuroDiffEq PyTorch-based library, Modulus....
  see [Cuomo *et al.*, 2022]

- PINNS Tutorial provided by Khoa Nguyen. More to come soon....



Tutorials for Physics-Informed Neural Networks (PINNs)

This repository provides step-by-step guides to Physics-informed neural networks (PINNs).

Part 1: Data-driven machine learning methods: strengths and limits

In this section, we only focus on data-driven machine learning methods. The tutorial shows how these methods approximate the solution of a parial diffrential equation (PDE).

To run online the tutorial:

Open in Colab (recommended, a google account is required)

launch binder

Part 2: PINNs and their scope of use

In this section, we focus on PINNs. The tutorial shows how to use PINNs to solve different types of problems involving partial differential equation (PDE) or system of PDEs.

To run online the tutorial:

Open in Colab (recommended, a google account is required)

launch binder

Part 3: Strategies to improve PINNs performance *(available soon)*

Part 4: Integrate the geometries into PINNs *(available soon)*

Acknowledgement

This work is funded by Michelin and CEA through the Industrial Data Analytics and Machine Learning chair of Borelli Center, ENS Paris-Saclay.

Table of content:

https ://github.com/nguyenkhoa0209/pinns_tutorial

a joint work on Physical Informed Neural Networks

thanks to the Industrial Data Analytics and Machine Learning chair

Industrial Data Analytics & Machine Learning

with

- Thi Nguyen Khoa Nguyen , Centre Borelli, CEA, Michelin,
- Thibault Dairay, Michelin
- Raphaël Meunier, Michelin
- Christophe Millet, CEA

Thank you!

References

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M. et Piccialli, F. (2022).

Scientific machine learning through physics–informed neural networks : Where we are and what's next.
Journal of Scientific Computing, 92(3):88.

Daw, A., Bu, J., Wang, S., Perdikaris, P. et Karpatne, A. (2022).

Rethinking the importance of sampling in physics-informed neural networks.
arXiv preprint arXiv :2207.02338.

Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H. et Zhu, J. (2022).

Physics-informed machine learning : A survey on problems, methods and applications.
arXiv preprint arXiv :2211.08064.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S. et Yang, L. (2021).

Physics-informed machine learning.
Nature Reviews Physics, 3(6):422–440.

Lu, L., Meng, X., Mao, Z. et Karniadakis, G. E. (2021).

Deepxde : A deep learning library for solving differential equations.
SIAM Review, 63(1):208–228.

Mehta, P., Bukov, M., Wang, C.-H., Day, A. G., Richardson, C., Fisher, C. K. et Schwab, D. J. (2019).

A high-bias, low-variance introduction to machine learning for physicists.
Physics reports, 810:1–124.

Meng, C., Seo, S., Cao, D., Griesemer, S. et Liu, Y. (2022).

When physics meets machine learning : A survey of physics-informed machine learning.
arXiv preprint arXiv :2203.16797.

References

Nguyen, T. N. K., Dairay, T., Meunier, R., Millet, C. et Mougeot, M. (2023a).

Fixed-budget online adaptive learning for physics-informed neural networks. towards parameterized problem inference.
In International Conference on Computational Science, pages 453–468. Springer.

Nguyen, T. N. K., Dairay, T., Meunier, R. et Mougeot, M. (2022).

Physics-informed neural networks for non-newtonian fluid thermo-mechanical problems : An application to rubber calendering process.
Engineering Applications of Artificial Intelligence, 114:105176.

Nguyen, T. N. K., Millet, C., Dairay, T., Meunier, R. et Mougeot, M. (2023b).

Using self-adaptive physics-informed learning to estimate orographic gravity waves.
Bulletin of the American Physical Society.

Oldenburg, J., Borowski, F., Öner, A., Schmitz, K.-P. et Stiehm, M. (2022).

Geometry aware physics informed neural network surrogate for solving navier–stokes equation (gapinn).
Advanced Modeling and Simulation in Engineering Sciences, 9(1):8.

Raissi, M., Perdikaris, P. et Karniadakis, G. E. (2019).

Physics-informed neural networks : A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational physics, 378:686–707.

Wu, C., Zhu, M., Tan, Q., Kartha, Y. et Lu, L. (2023).

A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks.
Computer Methods in Applied Mechanics and Engineering, 403:115671.