Convergence acceleration of a nonlinear solver using statistical learning

Rémy VALLOT [1-2]

Thibault DAIRAY [1-2] Florian DE VUYST [3] Mathilde MOUGEOT [2-4]

1. Michelin 2. Centre Borelli UMR 9010, Ecole Normale Supérieur Paris-Saclay
3. BMBI UMR 7338, Université de Technologie Compiègne 4. ENSIIE

SMAI société de mathématiques appliquées et industrielles

2-6 June 2025

Presentation outline

Context

Methods

Applications

Results

Perspectives

Numerical simulations in industry : crucial to reduce costly testing, conception, decision making, comprehension
Traditional iterative algorithms : convergence heavily dependant on initialization

Problem definition

Nonlinear Partial Differential Equation (PDE) problem discretization :
High-dimensional system of nonlinear algebraic equations, solved using the Quasi-Newton method.

The problem is defined as :

Finding $U^\star$ such that $F(U^\star, \lambda) = 0$, where λ is a vector of parameters

Strategy

The Quasi-Newton algorithm can take a numerous number of iterations to converge.
This algorithm is reliant on the initialization.

Strategy

The Quasi-Newton algorithm can take a numerous number of iterations to converge.
This algorithm is reliant on the initialization.

Use machine learning to find a new initialization for the nonlinear solver generating less iterations to converge.

Following the work of Jin et al. [1] and Aghili et al. [2]

[1] Jin, T., Maierhofer, G., Schratz, K., and Xiang, Y. (2025). A fast neural hybrid newton solver adapted to implicit methods for nonlinear dynamics.

[2] Aghili, J., Franck, E., Hild, R., Michel-Dansac, V., and Vigon, V. (2025). Accelerating the convergence of newton's method for nonlinear elliptic pdes using fourier neural operators. Communications in Nonlinear Science and Numerical Simulation,140:108434.

Strategy

The Quasi-Newton algorithm can take a numerous number of iterations to converge.
This algorithm is reliant on the initialization.

Use machine learning to find a new initialization for the nonlinear solver generating less iterations to converge.

Following the work of Jin et al. [1] and Aghili et al. [2]

The solver should be kept :

- to guarantee the physic of the solution
- better for users of the existing solver

[1] Jin, T., Maierhofer, G., Schratz, K., and Xiang, Y. (2025). A fast neural hybrid newton solver adapted to implicit methods for nonlinear dynamics.

[2] Aghili, J., Franck, E., Hild, R., Michel-Dansac, V., and Vigon, V. (2025). Accelerating the convergence of newton's method for nonlinear elliptic pdes using fourier neural operators. Communications in Nonlinear Science and Numerical Simulation,140:108434.

Data

Consider a set of vectors $\{\lambda_k\}$ of parameters, $k \in [\![1, n]\!]$, n number of observation snapshots

a set of associated discretized solutions $\{U_k^\star\}$, $U_k^\star := \begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix}_k^\star \in \mathbb{R}^d$ (computed with a numerical solver)

Data

Consider a set of vectors $\{\lambda_k\}$ of parameters, $k \in [\![1, n]\!]$, n number of observation snapshots

a set of associated discretized solutions $\{U_k^\star\}$, $U_k^\star := \begin{bmatrix} u_1 \\ \vdots \\ u_d \end{bmatrix}_k^\star \in \mathbb{R}^d$ (computed with a numerical solver)

Baseline strategies

Classical initialization : constant field set to boundary value.

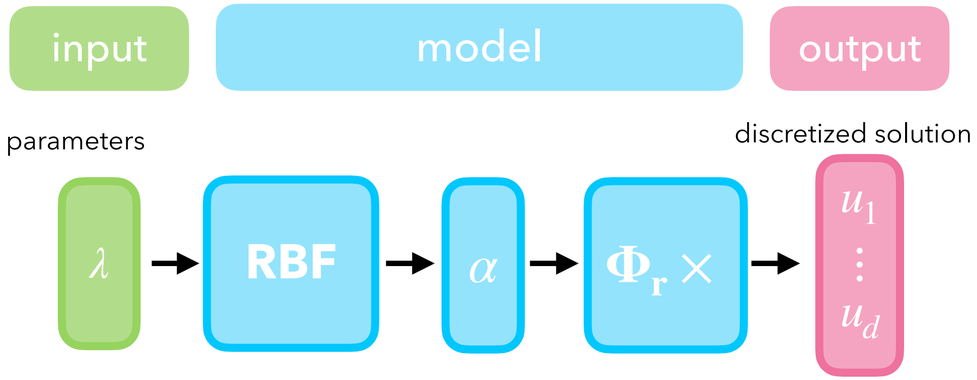
Nearest initialization : a naive strategy that use the solution from the closest parameter in the database

Proper Orthogonal Decomposition (POD) *[Lumley 1967]*

Consider a snapshot matrix $U = \begin{bmatrix} | & | & & | \\ U_1^\star & U_2^\star & \dots & U_n^\star \\ | & | & & | \end{bmatrix}$.

Compute truncated SVD ($U = \Phi_r \Sigma_r V_r^T$) to obtain Φ_r the orthonormal reduced basis between projection coefficients α_k and solution U_k.

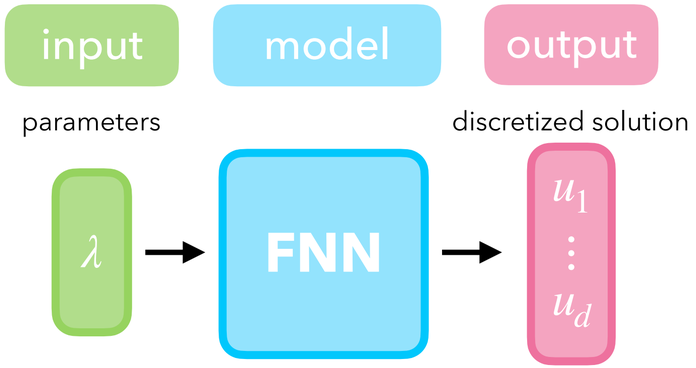
Define interpolation method like Radial Basis Function (RBF) between parameters and projection coefficients.



POD surrogate architecture

Lumley, J. L. (1967). The structure of inhomogeneous turbulent flows. Atmospheric Turbulence and Radio Wave Propagation, pages 166–178

Neural Networks (NN)



FNN surrogate architecture

The weights of the neural network are computed by optimizing a loss function that minimizes the mean squared error between the predicted outputs and the true values for the parameter λ_k given.

$$\min_{\theta} \mathscr{L}(\theta) = \frac{1}{n} \sum_{k=1}^{n} \left\| \text{FNN}_\theta(\lambda_k) - U_k \right\|^2$$

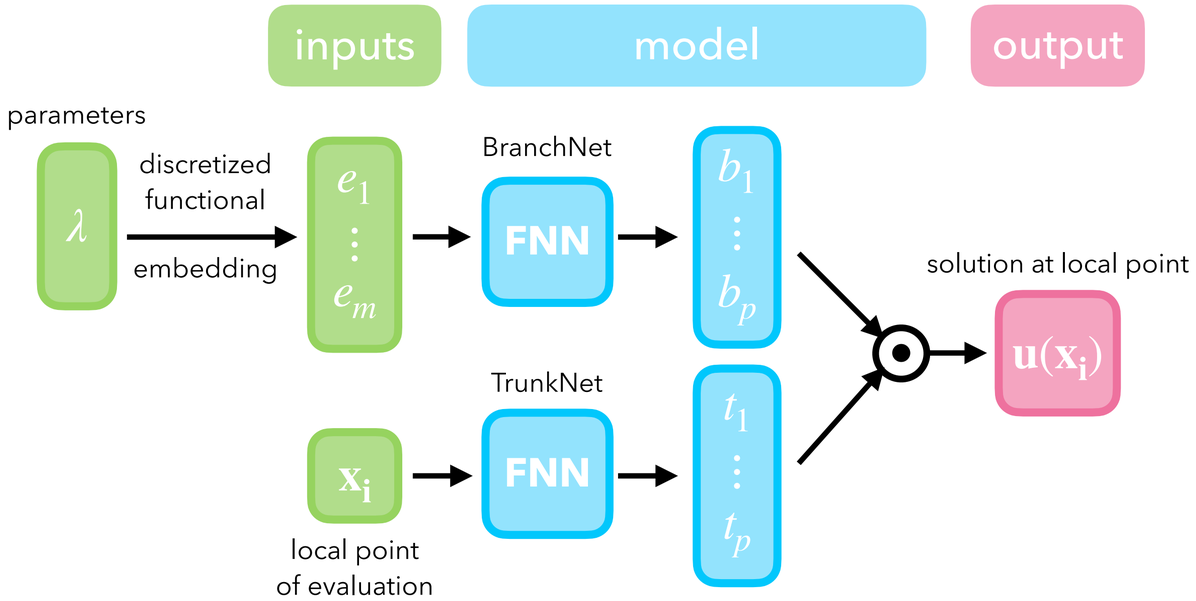
This architecture is **poorly scalable** : as many outputs as discretization points

The prediction points are fixed

Deep Operator Network (DeepONet) *[Lu 2019]*

Neural operator architecture to approximate mappings between functions

Supported by the **universal approximation theorem for operators** *[Chen & Chen 1995]*



DeepONet surrogate architecture

$i \in [\![1, d]\!]$

m : number of given sensors

p : size of reduced space

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. (2021). Learning non-linear operators via deeponet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229.

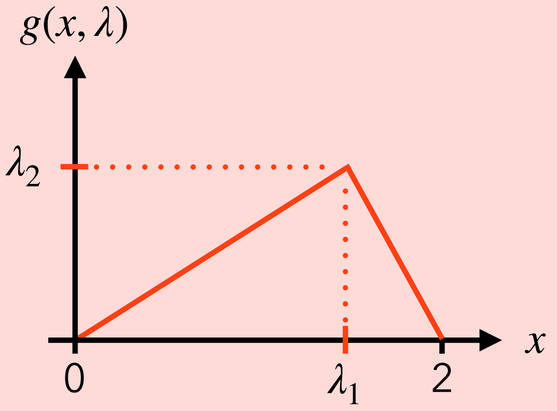
1D toy example

Let Ω be an open domain of $\mathbb{R}$. The nonlinear Poisson equation is formalized as follows :

$$\begin{cases} -\dfrac{\partial}{\partial x}\left[q(u)\dfrac{\partial u}{\partial x}\right] = g(x,\lambda) & \text{in } \Omega \\ u = u_D & \text{on } \partial\Omega \end{cases}$$

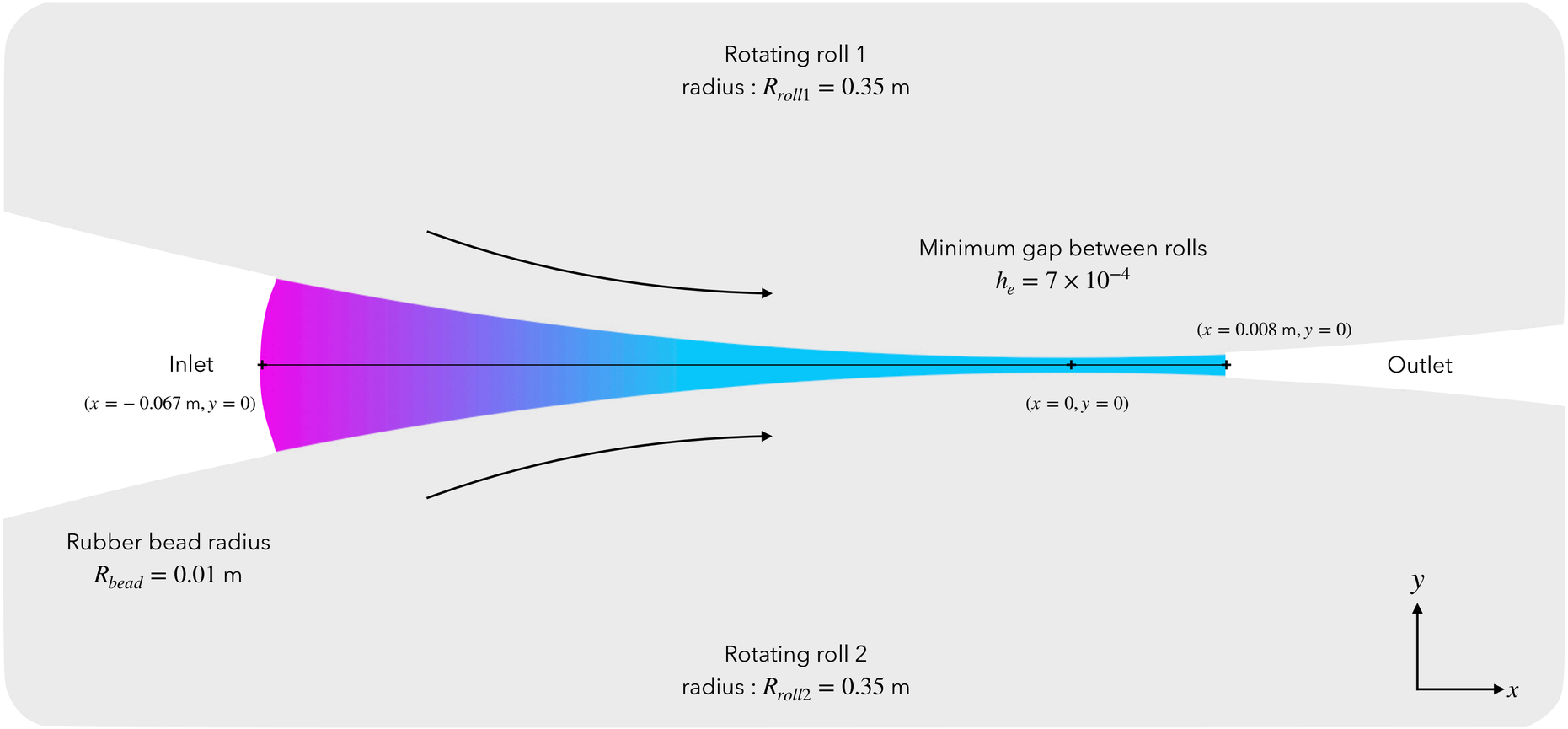
where $\Omega = [0,2]$; $q(u) := 1 + u^2$; $\lambda := \begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix}$;

$$g(x,\lambda) := \begin{cases} \dfrac{\lambda_2}{\lambda_1}x & \text{if } x \in [0,\lambda_1] \\ \dfrac{\lambda_2}{\lambda_1 - 2}(x-2) & \text{if } x \in [\lambda_1, 2] \end{cases}$$



The Quasi-Newton solver is created using Python and GetFEM library for finite element solving.

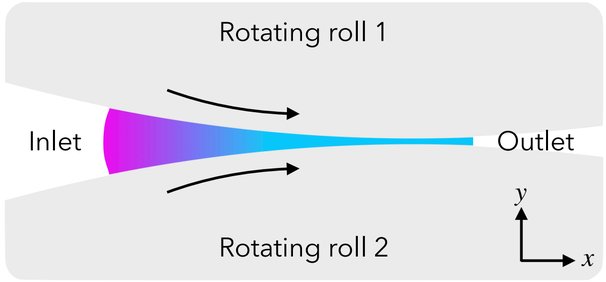
2D industrial example : calendering process



Sketch of the 2D calendering process, the domain of interest is the colored one

2D industrial example : calendering process

Focus on modeling the velocity field. Thermal effects are not considered in this study.



Governing equations:

Mass conservation (incompressible flow):

$$\nabla \cdot (\rho \vec{u}) = 0$$

Momentum conservation (neglecting body forces):

$$\nabla \cdot (2\mu \bar{\bar{\dot{\varepsilon}}}) = \nabla p$$

with :

$\bar{\bar{\dot{\varepsilon}}} = \frac{1}{2}\left(\nabla \vec{u} + (\nabla \vec{u})^T\right)$ and the truncated power-law viscosity model : $\mu_{\text{eff}}(\dot{\gamma}) = \begin{cases} \mu_{\min} & \text{if } \dot{\gamma} < \dot{\gamma}_{\min} \\ C_T \cdot \mathbf{K} \cdot \dot{\gamma}^{\mathbf{n}-1} & \text{if } \dot{\gamma}_{\min} \leq \dot{\gamma} \leq \dot{\gamma}_{\max} \\ \mu_{\max} & \text{if } \dot{\gamma} > \dot{\gamma}_{\max} \end{cases}$

The varying parameters of this case study are $\mathbf{K}$ and $\mathbf{n}$.

The solver used is MEF++ and is Quasi-Newton.

Brain refresh

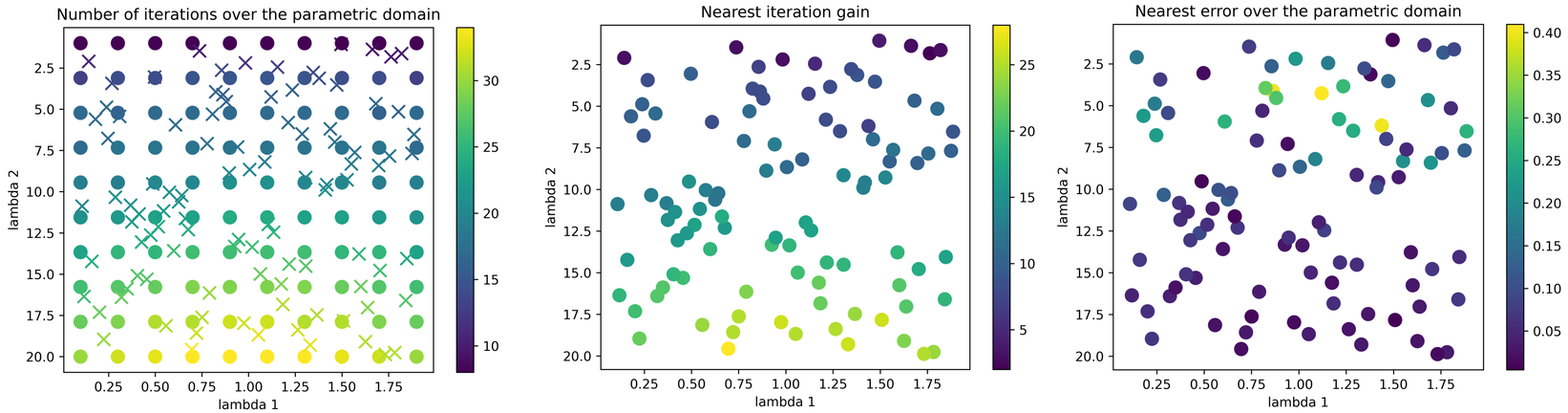
Nonlinear Poisson equation $-\frac{\partial}{\partial x}\left[q(u)\frac{\partial u}{\partial x}\right] = g(x,\lambda)$ in Ω

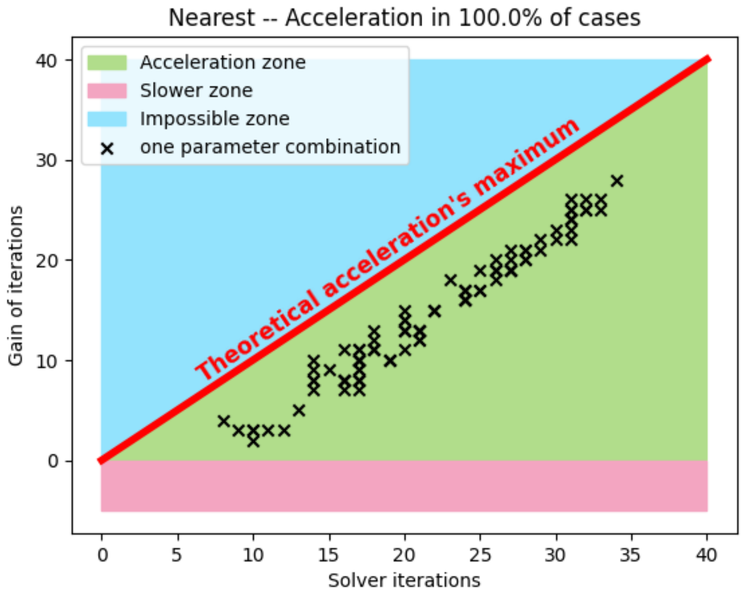
1D toy example

Varying parameters : λ_1, λ_2.

Baseline strategy : Nearest parameter solution

percentage gain mean : 63.47% – median : 66.67%





Sampling

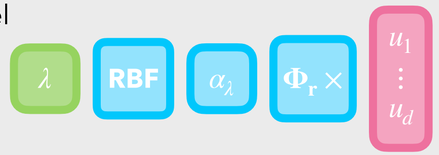
Train: 10×10 *regular grid* $\lambda_1 \in [0.1, 0.9]$, $\lambda_2 \in [1, 30]$

Test: 100 pairs via Latin Hypercube Sampling in the same intervals

Brain refresh

Nonlinear Poisson equation $-\frac{\partial}{\partial x}\left[q(u)\frac{\partial u}{\partial x}\right] = g(x,\lambda)$ in Ω

POD model

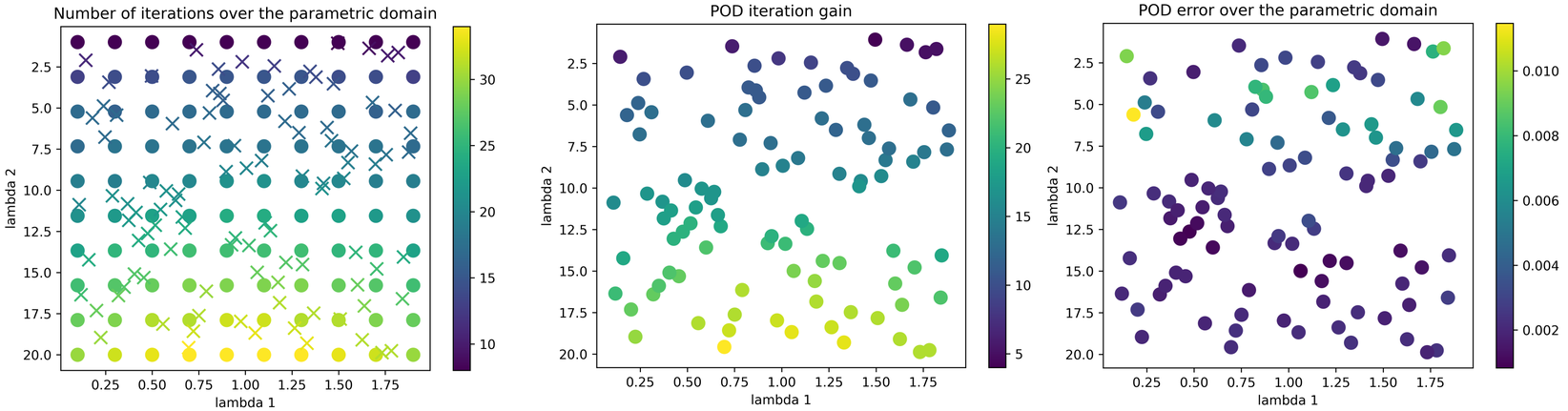


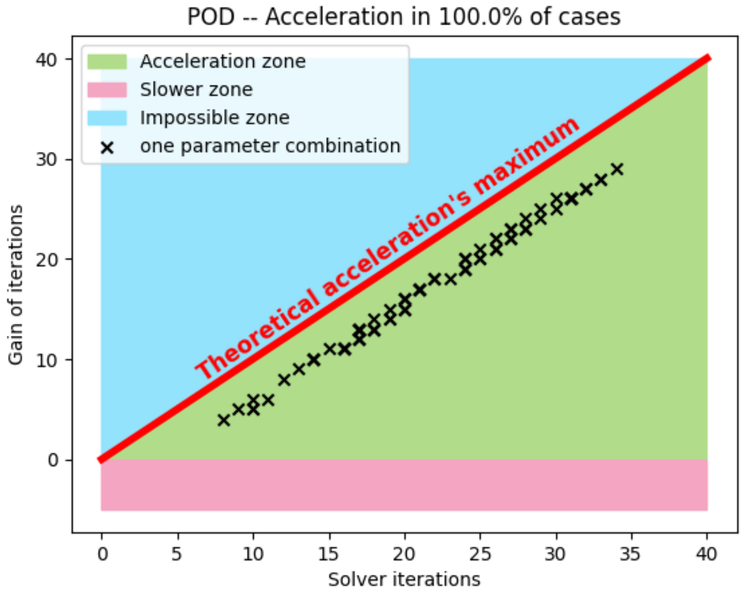
1D toy example

Varying parameters : λ_1, λ_2.

Proper Orthogonal Decomposition (POD)

percentage gain mean : 77.13% – median : 80.00%





Sampling

Train: 10×10 *regular grid* $\lambda_1 \in [0.1, 0.9]$, $\lambda_2 \in [1, 30]$

Test: 100 pairs via Latin Hypercube Sampling in the same intervals

Brain refresh

Nonlinear Poisson equation $-\frac{\partial}{\partial x}\left[q(u)\frac{\partial u}{\partial x}\right] = g(x,\lambda)$ in Ω

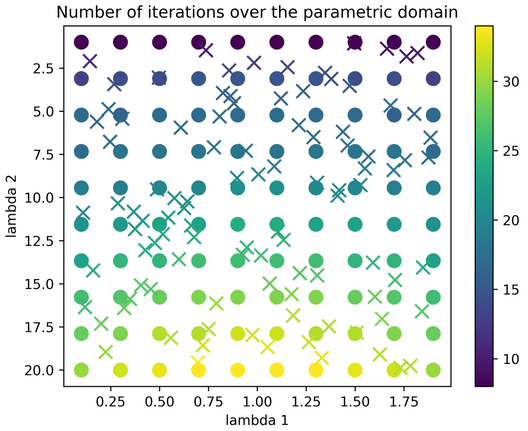
NN model

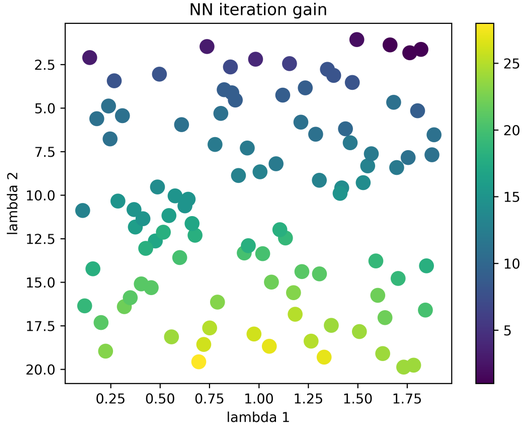
1D toy example

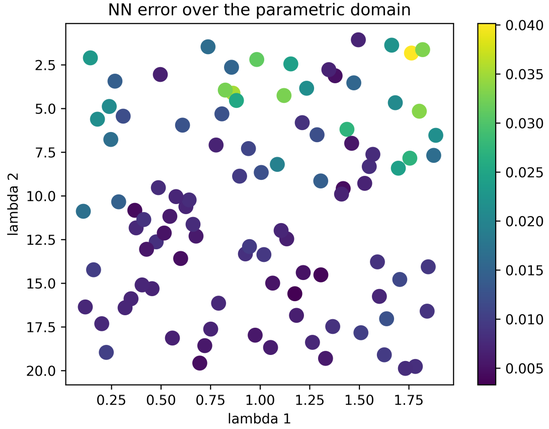
Varying parameters : λ_1, λ_2.

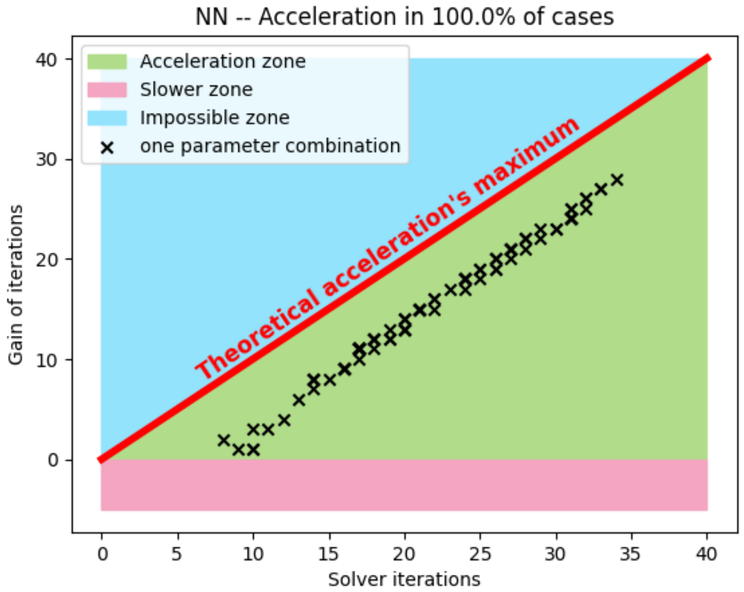
Neural Network (NN)

percentage gain mean : 66.58% – median : 71.43%









Sampling

Train: 10×10 *regular grid* $\lambda_1 \in [0.1, 0.9]$, $\lambda_2 \in [1, 30]$

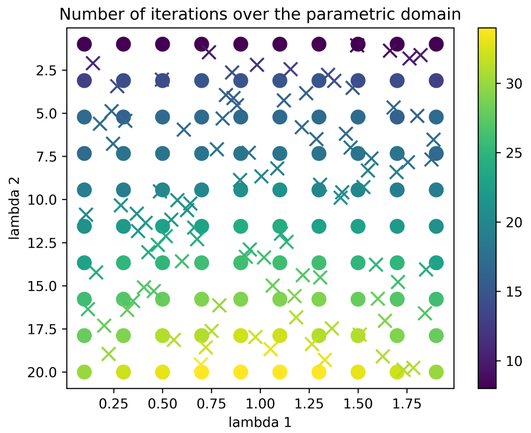
Test: 100 pairs via Latin Hypercube Sampling in the same intervals

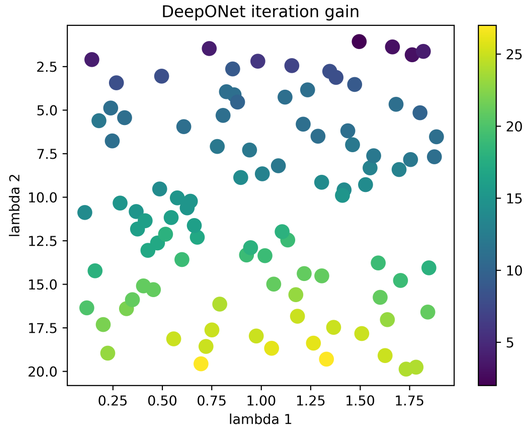
1D toy example

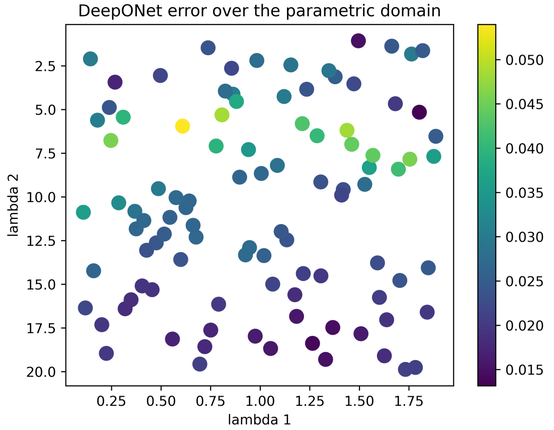
Varying parameters : λ_1, λ_2.

DeepONet

percentage gain mean : 68.89% – median : 70.83%







Sampling

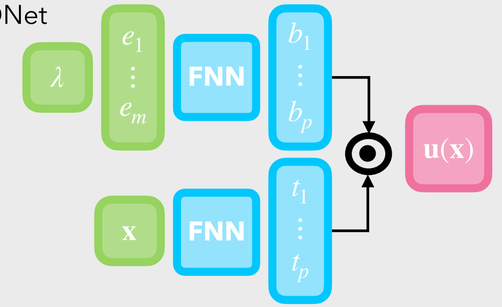
Train: 10×10 *regular grid* $\lambda_1 \in [0.1, 0.9]$, $\lambda_2 \in [1, 30]$

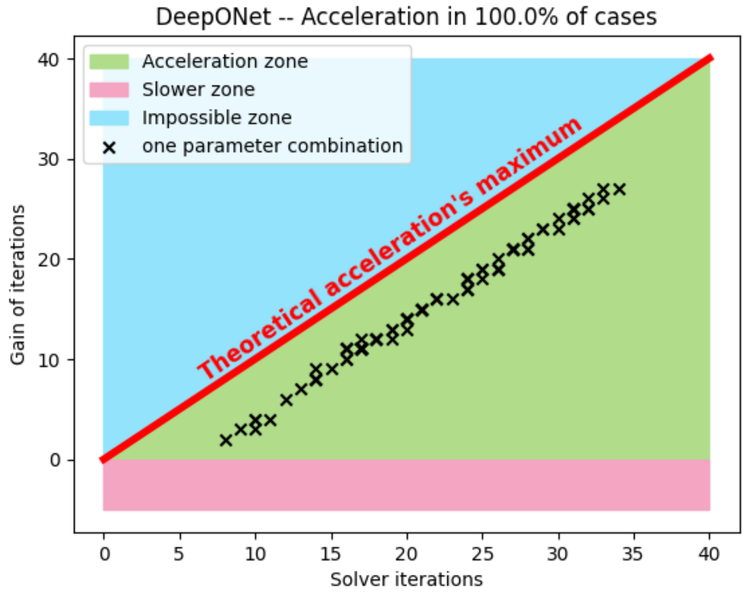
Test: 100 pairs via Latin Hypercube Sampling in the same intervals

Brain refresh

Nonlinear Poisson equation $-\frac{\partial}{\partial x}\left[q(u)\frac{\partial u}{\partial x}\right] = g(x,\lambda)$ in Ω

DeepONet

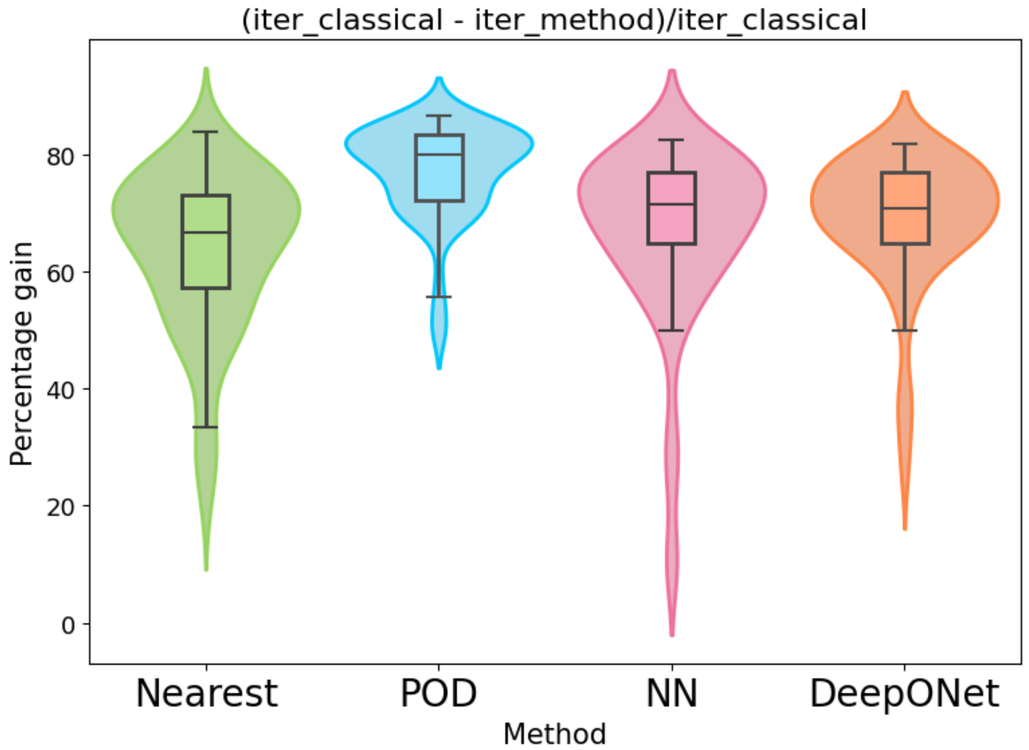




1D toy example

Varying parameters : λ_1, λ_2.

Method comparison



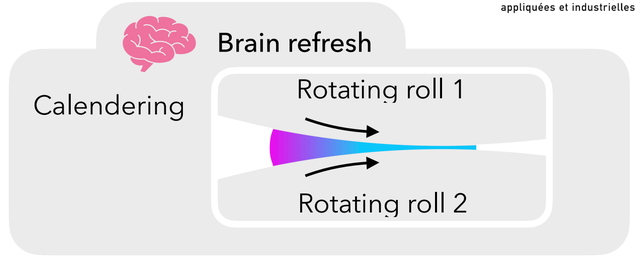
Brain refresh

Nonlinear Poisson equation $-\frac{\partial}{\partial x}\left[q(u)\frac{\partial u}{\partial x}\right] = g(x,\lambda)$ in Ω

Percentage gain recap

	Mean	Median
Nearest	63.47	66.67
POD	**77.13**	**80.00**
NN	66.58	71.43
DeepONet	68.89	70.83

n. obs = 100

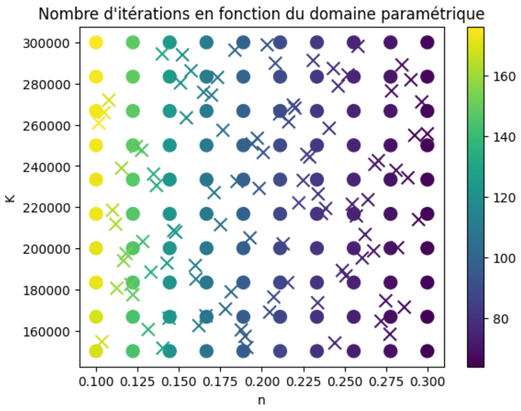


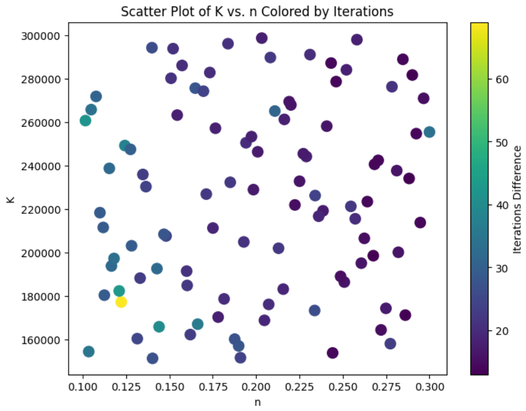
2D industrial example : calendering process

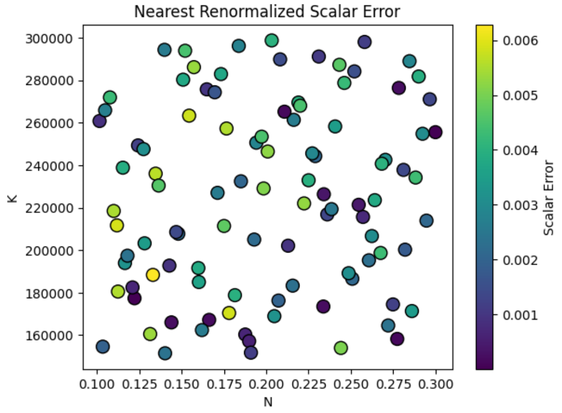
Varying parameters : $\mathbf{K}$, $\mathbf{n}$ in the truncated power-law viscosity model.

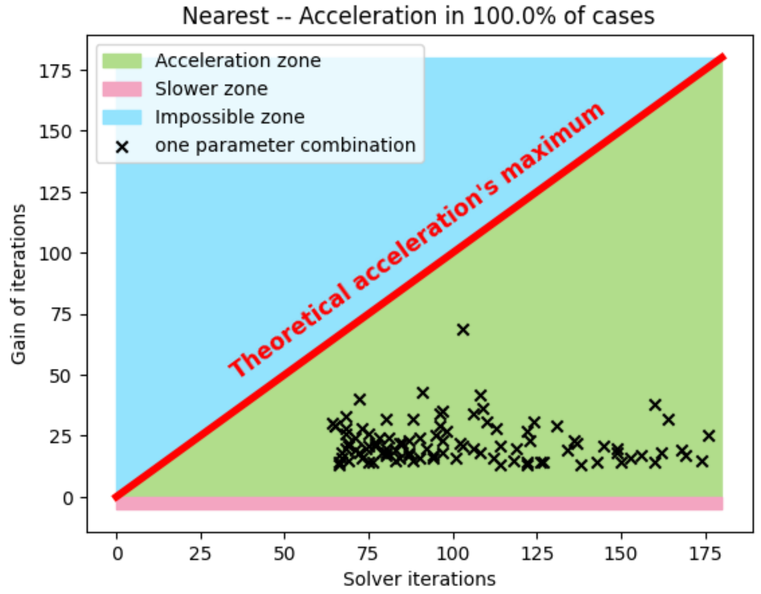
Baseline strategy : Nearest parameter solution

percentage gain mean : 22.16% – median : 20.54%









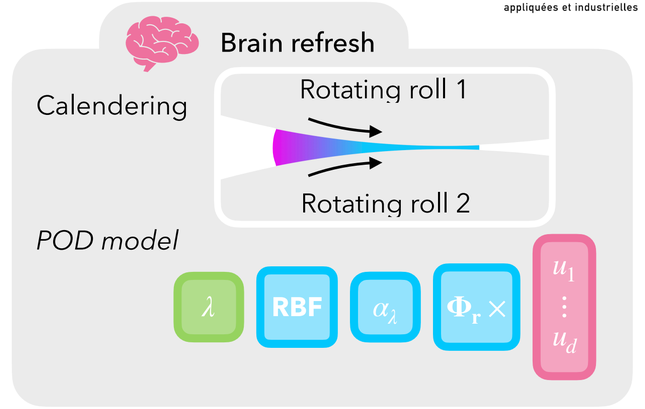
Sampling

Train: 10×10 *regular grid* $K \in [150e3, 300e3]$, $n \in [0.1, 0.3]$

Test: 100 pairs via Latin Hypercube Sampling in the same intervals

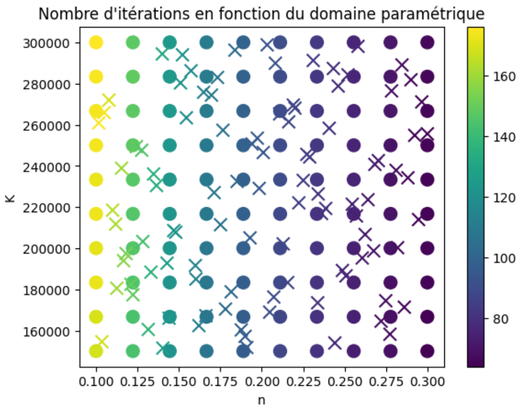
2D industrial example : calendering process

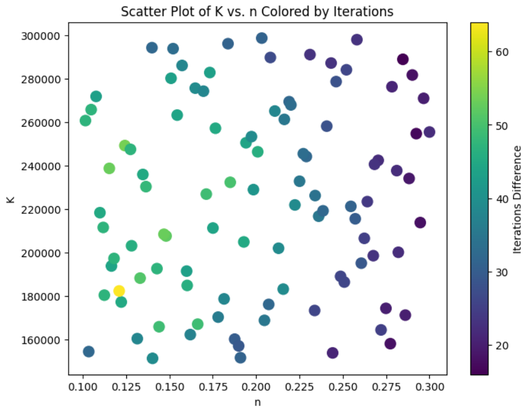
Varying parameters : $\mathbf{K}$, $\mathbf{n}$ in the truncated power-law viscosity model.

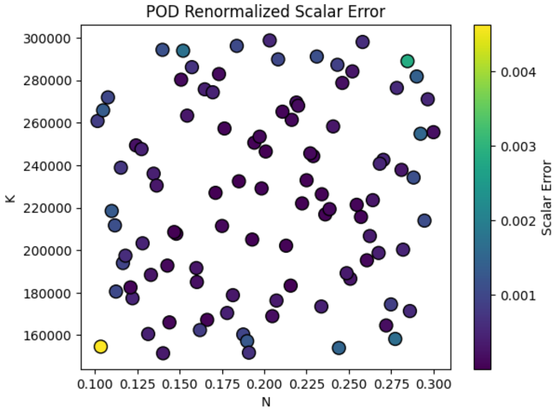


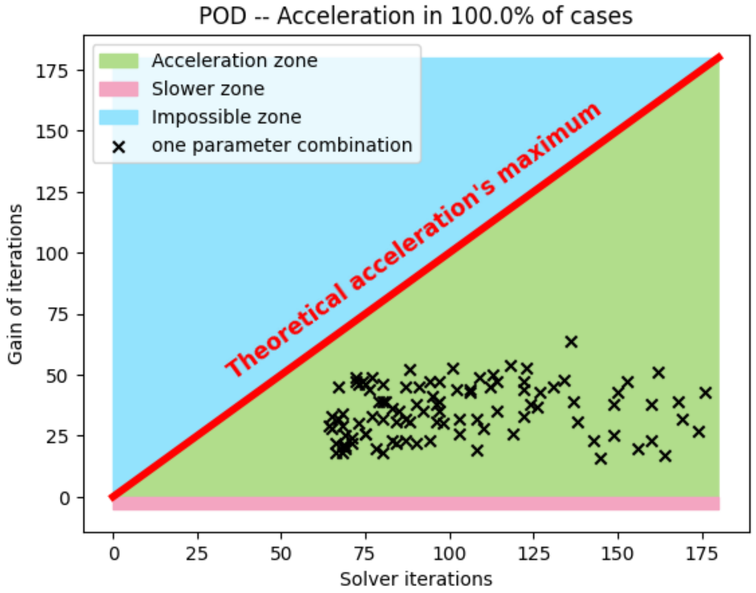
Proper Orthogonal Decomposition (POD)

percentage gain mean : 34.89% – median : 34.31%









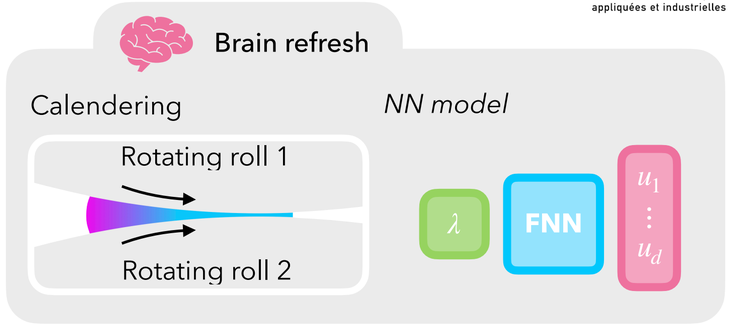
Sampling

Train: 10×10 *regular grid* $K \in [150e3, 300e3]$, $n \in [0.1, 0.3]$

Test: 100 pairs via Latin Hypercube Sampling in the same intervals

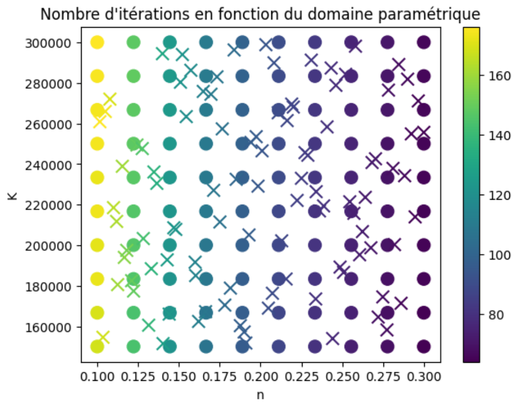
2D industrial example : calendering process

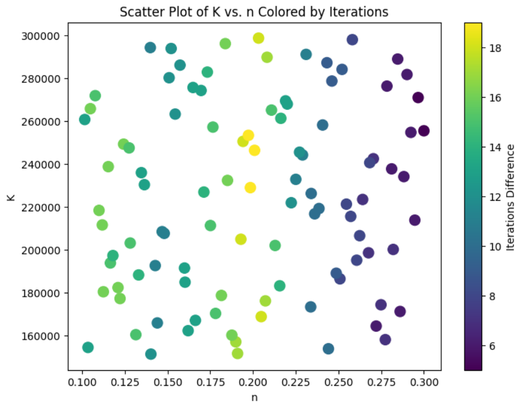
Varying parameters : $\mathbf{K}$, $\mathbf{n}$ in the truncated power-law viscosity model.

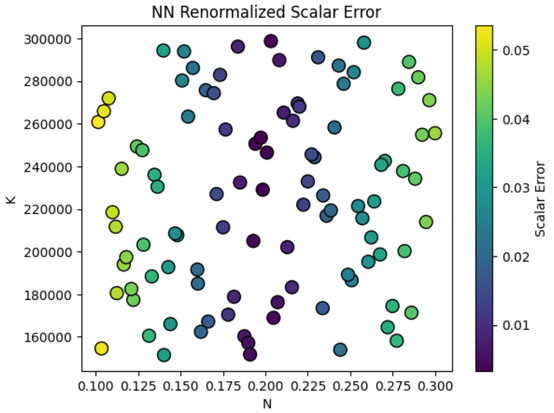


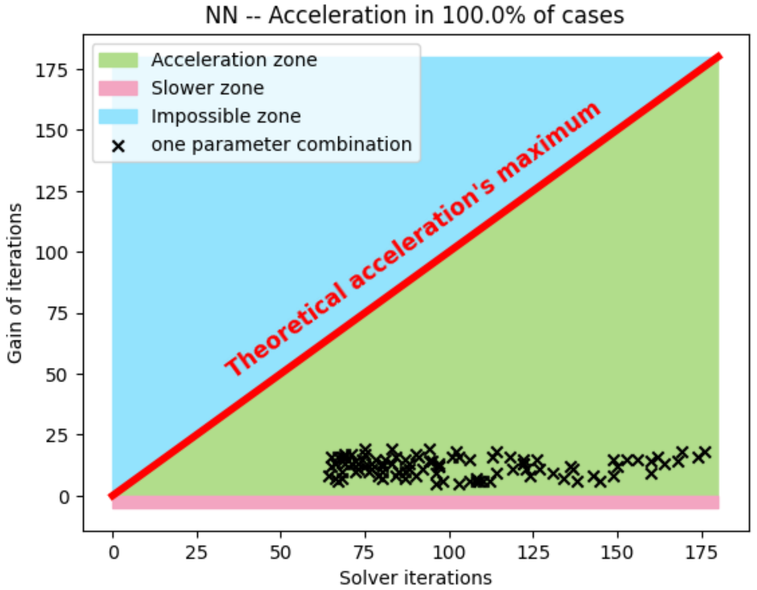
Neural Network (NN)

percentage gain mean : 12.06% – median : 10.91%









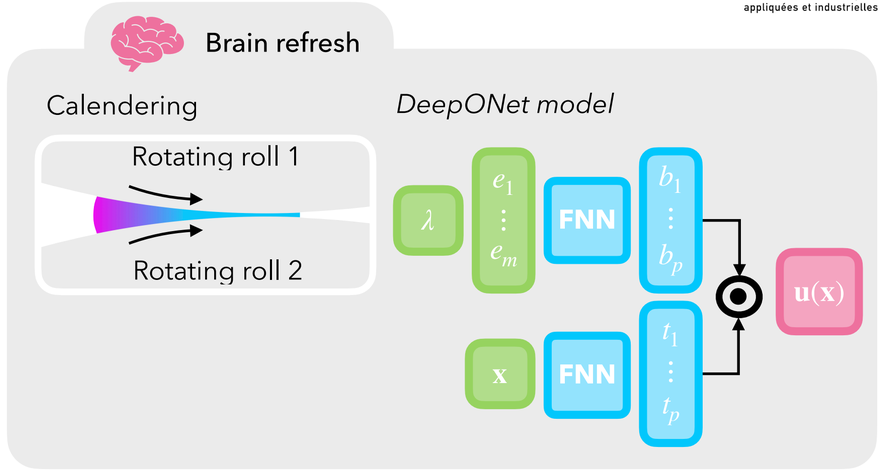
Sampling

Train: 10×10 *regular grid* $K \in [150e3, 300e3]$, $n \in [0.1, 0.3]$

Test: 100 pairs via Latin Hypercube Sampling in the same intervals

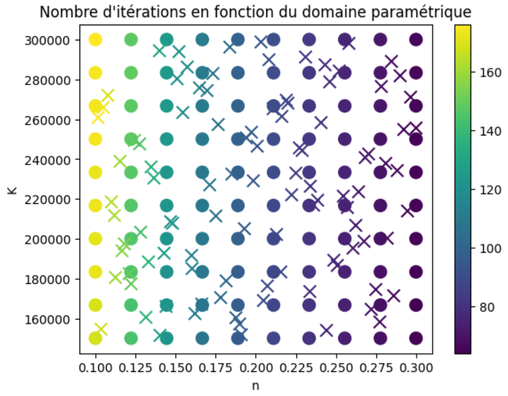
2D industrial example : calendering process

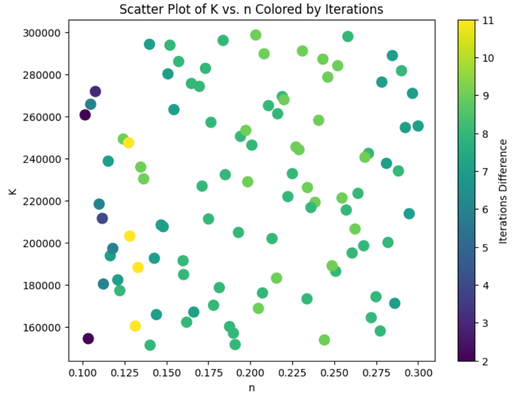
Varying parameters : $\mathbf{K}$, $\mathbf{n}$ in the truncated power-law viscosity model.

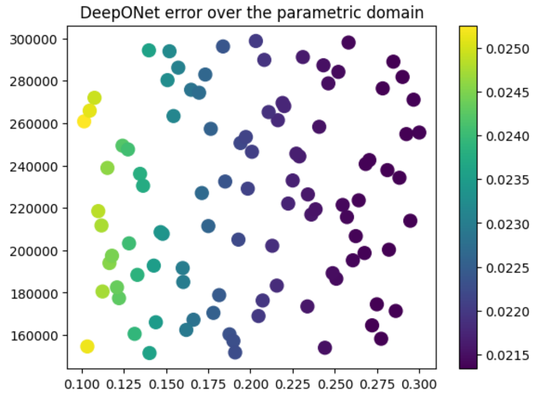


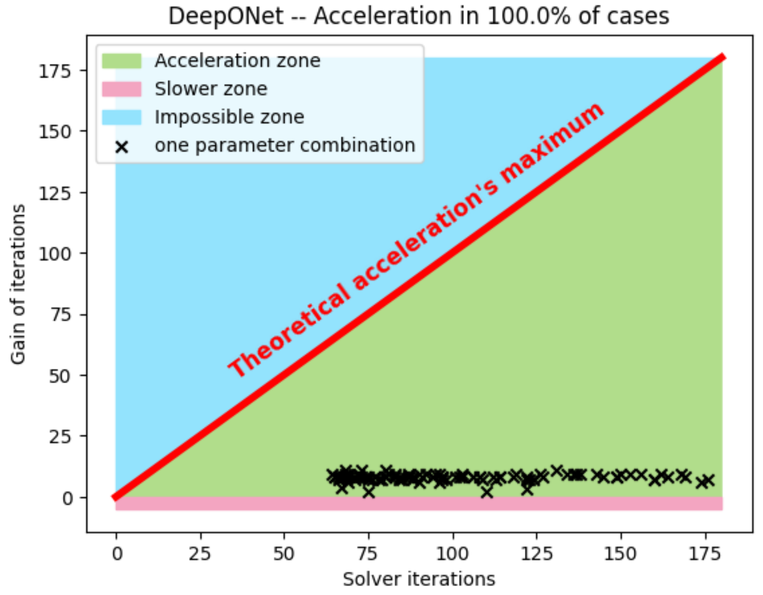
DeepONet

percentage gain mean : 8.49% – median : 8.89%









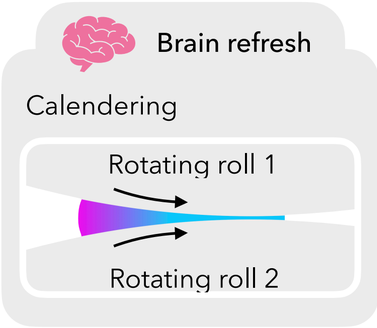
Sampling

Train: 10×10 *regular grid* $K \in [150e3, 300e3]$, $n \in [0.1, 0.3]$

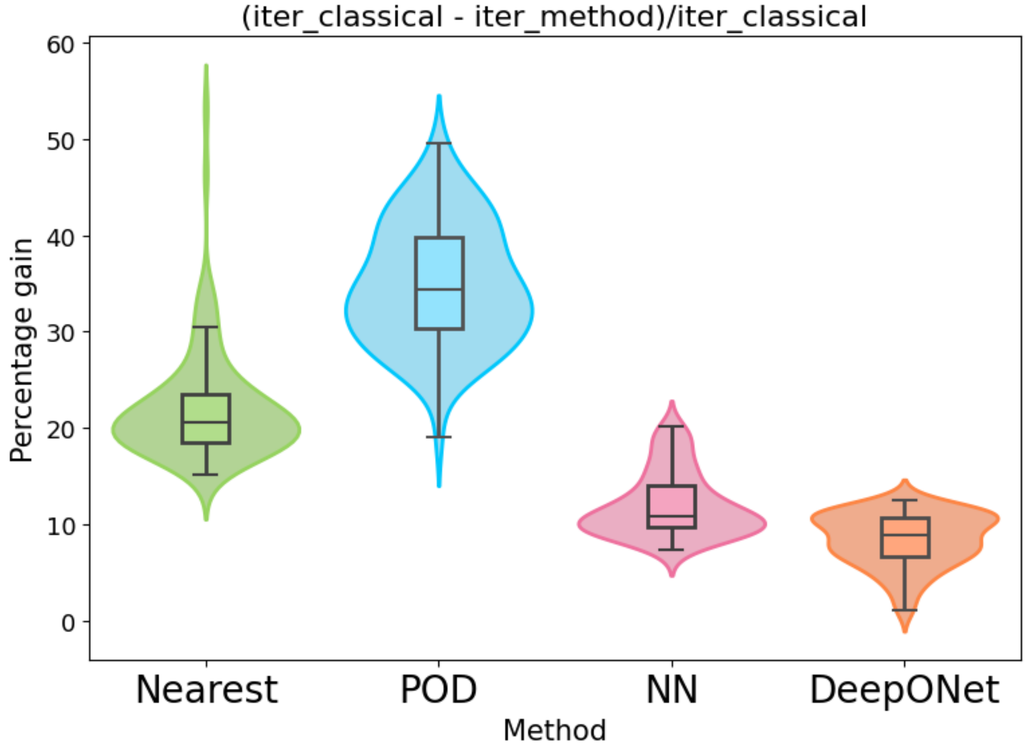
Test: 100 pairs via Latin Hypercube Sampling in the same intervals

2D industrial example : calendering process

Varying parameters : $\mathbf{K}$, $\mathbf{n}$ in the truncated power-law viscosity model.



Method comparison



Percentage gain recap

	Mean	Median
Nearest	22.16	20.54
POD	**34.89**	**34.31**
NN	12.06	10.91
DeepONet	8.49	8.89

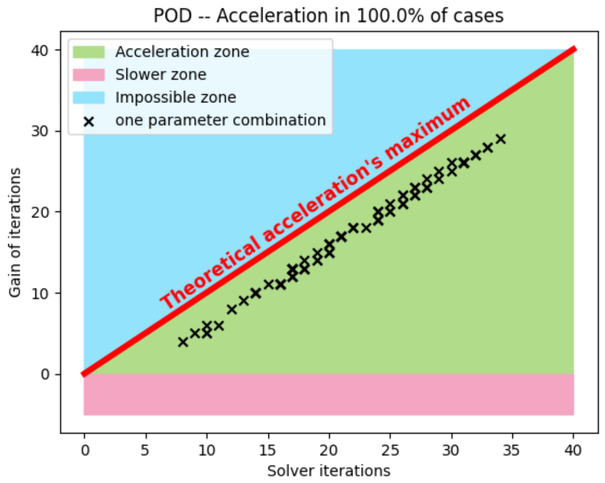
n. obs = 100

Preliminary conclusions

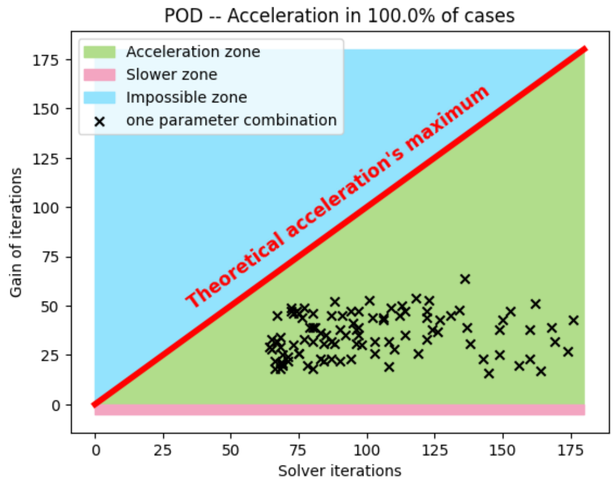
All initialization methods perform better than using a constant field.

POD consistently outperforms all other tested methods.

In 1D, the iteration gain grows with baseline cost



In 2D the iteration gain is constant



Why not PINNs ? Longer training time, inefficient for parametric studies

What's next ?

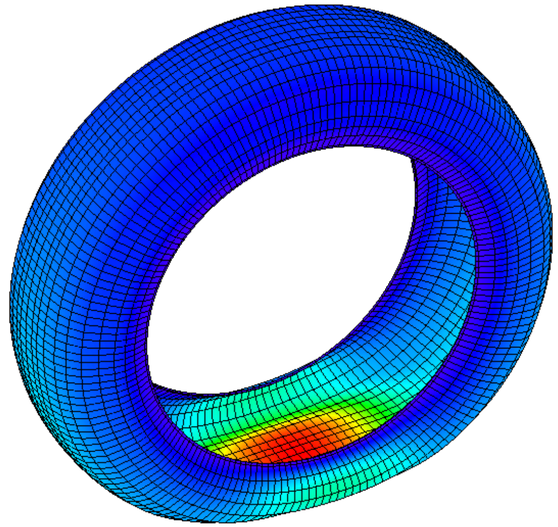
Extension to **industrial test cases**

Extension to 3D problems

Industrial application can be : tire loading under realistic conditions

Generalization to **geometrical variability**

Uncertainty quantification



How to assess confidence in surrogate-based initialization ?

Thank you !