

Analysis and improvement of the convergence of physics-informed neural networks

Nathan Ricard

Supervisor: Régis Duvigneau, Co-supervisor: Mickaël Binois

October 14, 2026

Table of contents

Generalities

- Problem Definition
- PINNs overview
- Example : 2D Heated Cavity
- Interests of PINNs

PINNs convergence analysis, their problems and their causes

- Neural network expressivity
- Comparison between PINN and data fitting
- Hessian's conditionning
- Conflict between PDE loss and BC loss

Min-max optimization method

Annexes

- Findings
 - Comparison between PINN and data fitting
 - Imposition of boundary conditions
 - Natural gradient descent
 - Research axis: Nash Games Training
 - Spectral co-clustering

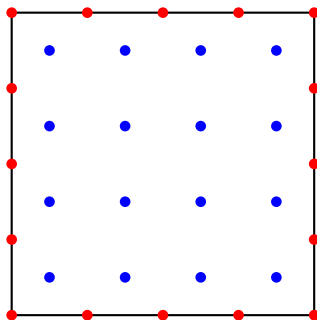
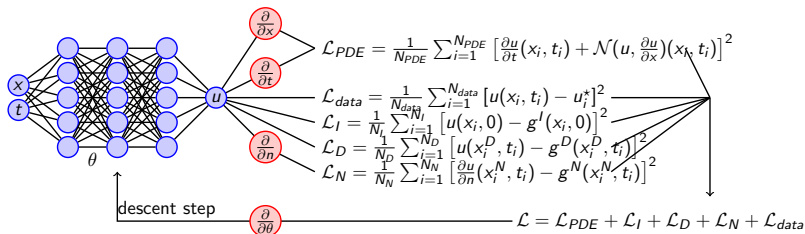
Problem Definition

- Generally, a PDE problem is:

$$\left\{ \begin{array}{l} \frac{\partial u}{\partial t} + \mathcal{N}[u] = 0, (t, x) \in [0, T] \times \Omega, \\ u(0, x) = g(x), x \in \Omega, \\ \mathcal{B}[u] = 0, (t, x) \in [0, T] \times \partial\Omega, \end{array} \right.$$

- Could be solved using classical methods (FEM, FD,...)
- But requires specific schemes and grid construction
- Objective: approximate u with a neural network u_θ , where θ denotes all tunable parameters

PINNs overview

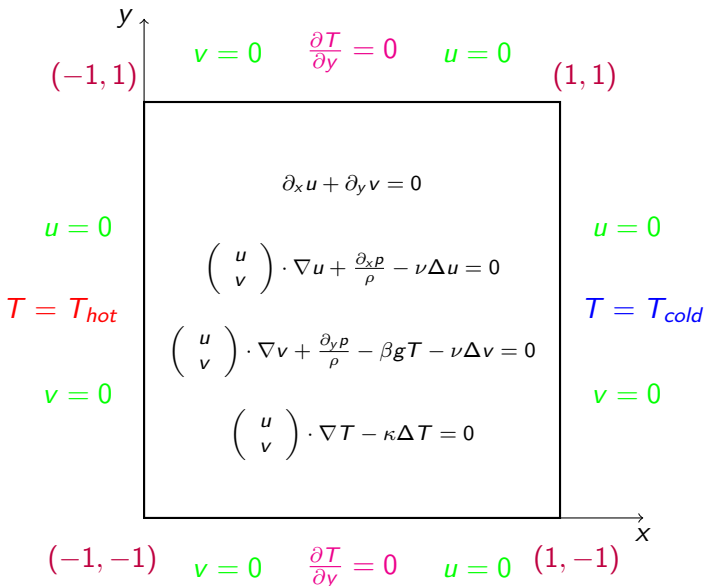


Interior points (PDE)

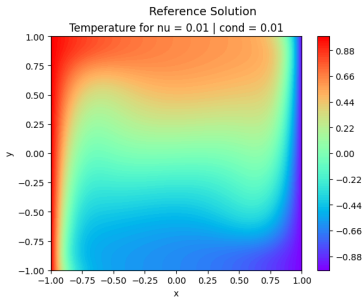
Boundary points (BC)

Figure: Squared domain with collocation points for PDE and boundary

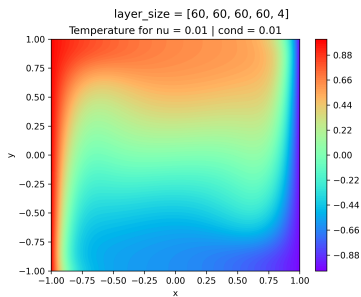
Example : 2D Heated Cavity



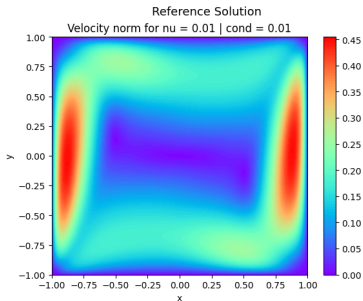
2D Heated Cavity : Problem solution for $\nu = \kappa = 0.01$



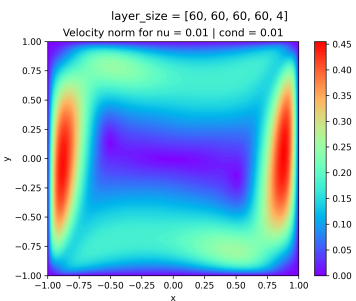
(a) Reference - Temperature



(b) PINN - Temperature



(c) Reference - Velocity Norm



(d) PINN - Velocity Norm

Interests and limitations of PINNs

- ▶ Limitations¹ :
 - ▶ Choice of hyperparameters : architecture, optimizer, activation, ...
 - ▶ Computational time
 - ▶ Convergence failures or difficulties
- ▶ The main interests of the PINN² are the following :
 - ▶ resolution of parametric problems (generalize predictions)
 - ▶ resolution of inverse problem (reconstruction of unknown parameters)
 - ▶ resolution of problem with data (physics + data)

Phd topic: analyse and improve PINN convergence

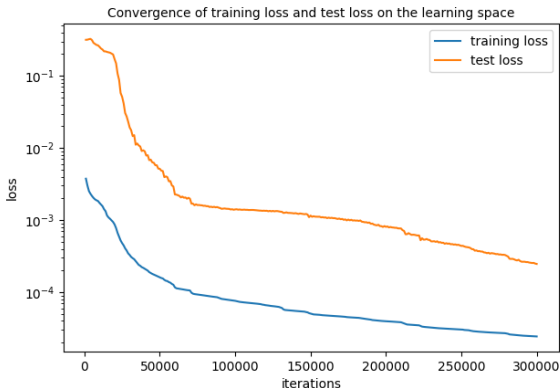
¹S. Krishnapriyan, A. Gholami, S. Zhe, R.M. Kirby, M.W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks", 2021.

²M. Raissi, P. Perdikaris, G.E. Karniadakis, "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, February 2019.

Neural network expressivity

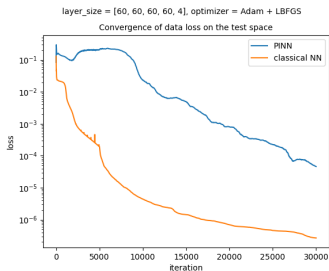
- We work to the 2D Heated Cavity problem.
- Objective of this study : see if a small neural network (physics informed or data fitting approach) can approximate correctly the reference solution.

layer_size = [20, 20, 20, 4], optimizer = L-BFGS, Classical method

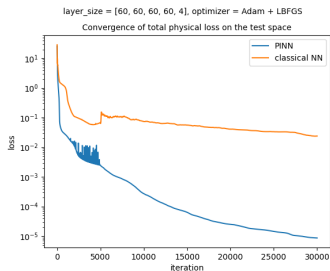


(a) Small NN : PINN

Comparison between PINN and data fitting



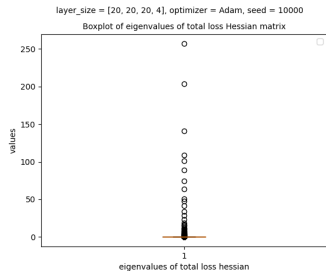
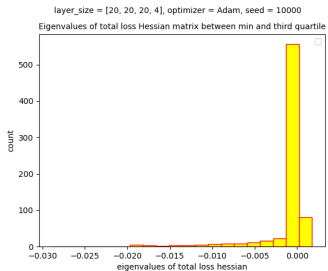
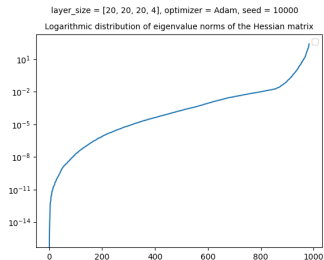
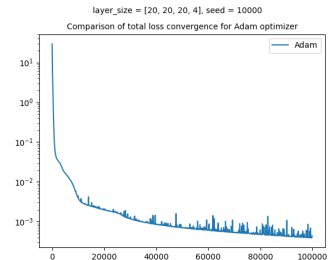
(a) Data loss



(b) Physical loss

- ▶ Good convergence of both physical and data loss for the PINN
- ▶ Close to the reference solution for the classical neural network but does not respect physics

Hessian's conditioning

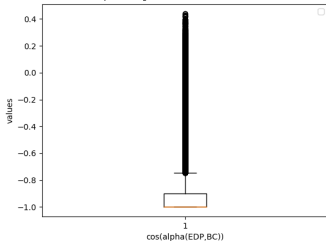


Anisotropic problem: low efficiency of first order method

Conflict between PDE loss and BC loss

layer_size = [20, 20, 20, 4], optimizer = Adam, seed = 1245

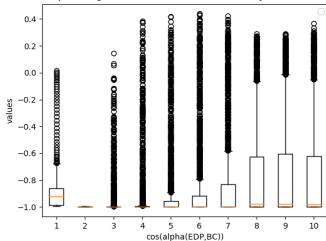
Boxplot of angles between PDE loss and BC loss



(a) Small NN

layer_size = [20, 20, 20, 4], optimizer = Adam, seed = 1245

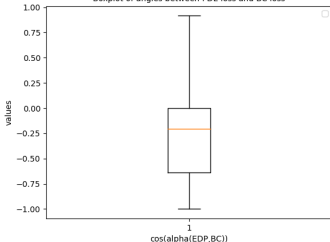
Boxplot of angles between total PDE and BC loss every 2000 iterations



(b) Small NN

layer_size = [60, 60, 60, 60, 4], optimizer = Adam, seed = 789

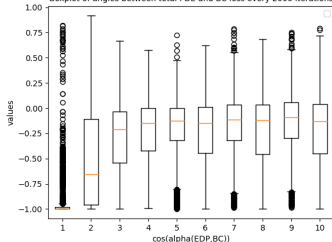
Boxplot of angles between PDE loss and BC loss



(c) Large NN

layer_size = [60, 60, 60, 60, 4], optimizer = Adam, seed = 789

Boxplot of angles between total PDE and BC loss every 2000 iterations



(d) Large NN

PINNs as a Multi-Objective Optimization Problem

Physics-Informed Neural Networks (PINNs) naturally lead to a multi-objective optimization formulation. We consider m objective functions:

$$f_1(\theta), f_2(\theta), \dots, f_m(\theta),$$

typically corresponding to the residual of the differential equation, boundary conditions, or additional data constraints.

Classical training minimizes a weighted sum:

$$F(\theta) = \sum_{i=1}^m w_i f_i(\theta),$$

which introduces several difficulties:

- ▶ imbalance in scale between the losses,
- ▶ conflicting objectives,
- ▶ delicate and problem-dependent choice of weights w_i .

Epigraphic Min–Max Formulation

To try to resolve these problems, we come back to multi-objective formulation consists in solving:

$$\min_{t, \alpha} t$$

subject to:

$$t \geq f_i - \alpha^\top W_i, \quad i = 1, \dots, m.$$

This epigraphic formulation corresponds to minimizing the **worst linearized objective**:

$$\min_{\alpha} \max_i \left(f_i - \alpha^\top W_i \right).$$

It yields a direction $d = \sum_i \alpha_i g_i$ that reduces the maximal objective value.

Geometric Interpretation

The method seeks a direction d such that:

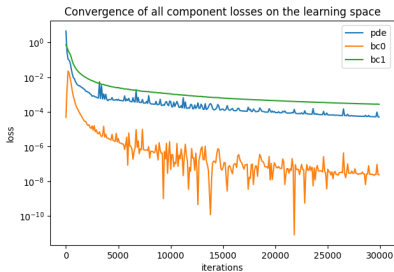
- ▶ the worst linearized objective is minimized;
- ▶ the objectives are brought closer to each other;
- ▶ conflicts between gradients are handled through W .

This can be interpreted as:

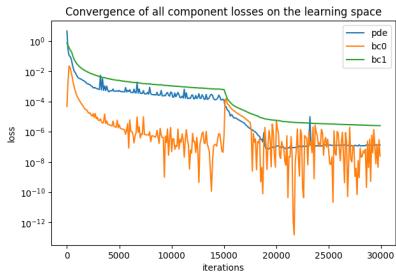
- ▶ a projection toward a region where objectives are equal,
- ▶ a search for a balanced point on the Pareto front,
- ▶ a reduction of the dominant component of the loss vector.

Min-max optimization method : results

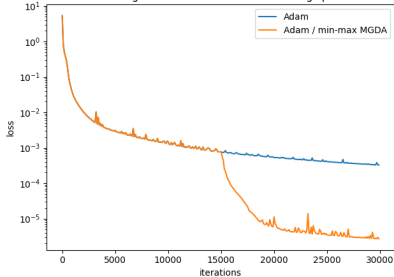
Adam



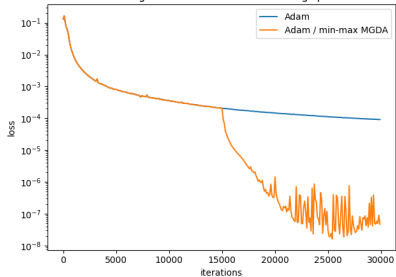
Adam then min-max MGDA



Convergence of total loss on the learning space

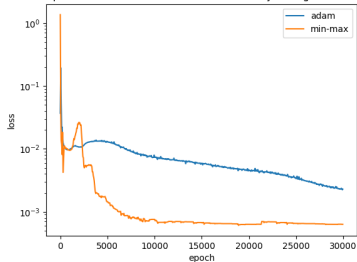


Convergence of L2 loss on the learning space

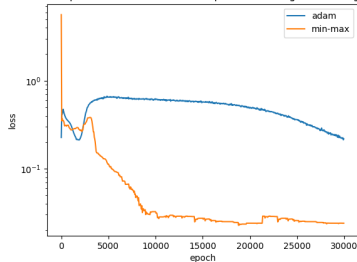


Min-max optimization method : results

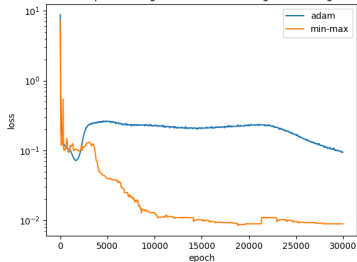
Comparison of L2 losses for horizontal velocity during the training



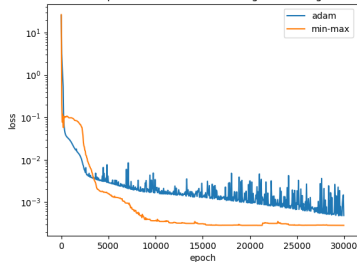
Comparison of L2 losses for temperature during the training



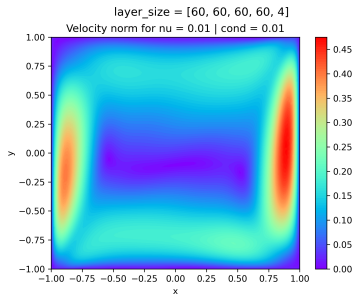
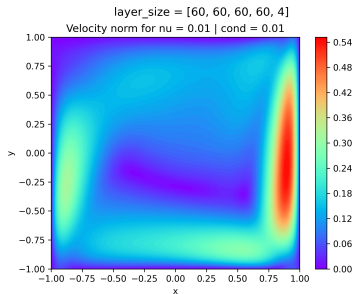
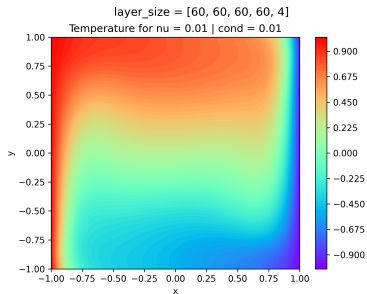
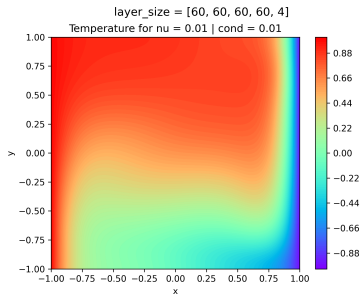
Comparison of global L2 losses during the training



Comparison of total losses during the training



Min-max optimization method : results



Advantages for PINNs

This MGDA variant offers several major advantages:

- ▶ **Automatic balancing** of PDE, boundary, and data losses;
- ▶ **No need for manual tuning** of loss weights;
- ▶ **Explicit handling of conflicts** between objectives;
- ▶ **More uniform convergence** toward a physically consistent solution;
- ▶ **Increased robustness** to ill-conditioning;
- ▶ **Natural adaptation** to the multi-constraint structure of PDEs.

This method is particularly well suited for PINNs, where the simultaneous satisfaction of several constraints is essential.

End of the presentation

Thanks for your attention!

Annexes

PINNs Definition

- For this, a PINN is a neural network which can be trained by minimizing the composite loss function:

$$\mathcal{L}(\theta) = w_i \mathcal{L}_{IC}(\theta) + w_b \mathcal{L}_{BC}(\theta) + w_r \mathcal{L}_{PDE}(\theta),$$

where $(w_i, w_b, w_r) \in \mathbb{R}_+^3$ and :

$$\mathcal{L}_{IC}(\theta) = \frac{1}{N_{IC}} \sum_{i=1}^{N_{IC}} |u_\theta(0, x_{IC}^i) - g(x_{IC}^i)|^2,$$

$$\mathcal{L}_{BC}(\theta) = \frac{1}{N_{BC}} \sum_{i=1}^{N_{BC}} |\mathcal{B}[u_\theta](t_{BC}^i, x_{BC}^i)|^2,$$

$$\mathcal{L}_{PDE}(\theta) = \frac{1}{N_{PDE}} \sum_{i=1}^{N_{PDE}} |\mathcal{R}_\theta(t_{PDE}^i, x_{PDE}^i)|^2,$$

with :

$$\mathcal{R}_\theta(t, x) = \frac{\partial u_\theta}{\partial t}(t, x) + \mathcal{N}[u_\theta](t, x)$$

PINNs Definition

- ▶ If we have data measurements on the points $\{t_{data}^i, x_{data}^i\}_{i=1}^{N_{data}}$ of the true solution u , a PINN can be trained by minimizing the following loss function:

$$\mathcal{L}(\theta) = w_i \mathcal{L}_{IC}(\theta) + w_b \mathcal{L}_{BC}(\theta) + w_r \mathcal{L}_{PDE}(\theta) + w_d \mathcal{L}_{data}(\theta),$$

where $w_d \in \mathbb{R}_+$ and:

$$\mathcal{L}_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u_{\theta}(t_{data}^i, x_{data}^i) - u(t_{data}^i, x_{data}^i)|^2,$$

- ▶ The points $\{t_{IC}^i, x_{IC}^i\}_{i=1}^{N_{IC}}, \{t_{BC}^i, x_{BC}^i\}_{i=1}^{N_{BC}}, \{t_{PDE}^i, x_{PDE}^i\}_{i=1}^{N_{PDE}}$ can be points are randomly sampled or the vertices of a mesh. To optimize the PINN loss function, we use, at each iteration, an optimizer like descend gradient algorithms (Adam, SGD, etc.) or Newton and Quasi-Newton methods (BFGS, L-BFGS, etc.).

Example: 2D Heated Cavity

We consider a steady flow in a squared domain of size 2×2 subject to differentially heated lateral walls. The temperature of the left wall is $T_L = 1$, whereas the one of the right wall is $T_R = -1$, adiabaticity being imposed for top and bottom walls. The governing equations are the incompressible Navier-Stokes equations with buoyancy and gravity, denoted as $\mathcal{N}_{NS}$:

$$\forall (x, y) \in [-1, 1]^2,$$

$$\partial_x u + \partial_y v = 0,$$

$$u \partial_x u + v \partial_y u + \frac{\partial_x p}{\rho} - g_x - \nu (\partial_{xx} u + \partial_{yy} u) + g_x \cdot \text{bulk} \cdot T = 0,$$

$$u \partial_x v + v \partial_y v + \frac{\partial_y p}{\rho} - g_y - \nu (\partial_{xx} v + \partial_{yy} v) + g_y \cdot \text{bulk} \cdot T = 0,$$

$$c_p (u \partial_x T + v \partial_y T) - \frac{\text{cond}}{\rho} (\partial_{xx} T + \partial_{yy} T) = 0,$$

where:

- ρ is the fluid density ($\rho = 1$)
- $g = (g_x, g_y)$ the gravity vector field ($g = (0.0, -10.0)$)
- ν the fluid dynamic viscosity ($\nu \in \{0.01, 0.0316, 0.1\}$)
- bulk the bulk coefficient ($\text{bulk} = 0.1$)
- c_p the thermic capacity ($c_p = 1$)
- cond the fluid thermic conductivity ($\text{cond} \in \{0.01, 0.0316, 0.1\}$)

Example: 2D Heated Cavity

In this case, the loss PINN to minimize is the following:

$$\mathcal{L}(\theta) = \mathcal{L}_{NS}(\theta) + \mathcal{L}_W(\theta) + \mathcal{L}_T(\theta) + \mathcal{L}_A(\theta),$$

with :

$$\mathcal{L}_W(\theta) = \frac{1}{N_W} \sum_{i=1}^{N_W} |u(x_i^W, y_i^W)|^2 + |v(x_i^W, y_i^W)|^2,$$

$$\mathcal{L}_T(\theta) = \frac{1}{N_{cold}} \sum_{i=1}^{N_{cold}} |T(x_i^{cold}, y_i^{cold}) - T_{cold}|^2 + \frac{1}{N_{hot}} \sum_{i=1}^{N_{hot}} |T(x_i^{hot}, y_i^{hot}) - T_{hot}|^2,$$

$$\mathcal{L}_A(\theta) = \frac{1}{N_A} \sum_{i=1}^{N_A} \left| \frac{\partial T}{\partial n}(x_i^A, y_i^A) \right|^2,$$

$$\mathcal{L}_{NS}(\theta) = \frac{1}{4} \sum_{j=1}^4 \frac{1}{N_{NS}} \sum_{i=1}^{N_{NS}} |\mathcal{R}_{\theta}^j(x_i^{NS}, y_i^{NS})|^2,$$

where $\mathcal{R}_{\theta}^j$ represents the residual associated to jth equation of Navier-Stokes incompressible equations.

Reminder: Gradient Descent method

- ▶ The Gradient Descent is an optimisation method based on following iteration:

$$\theta_{k+1} = \theta_k - \alpha \nabla f(\theta_k)$$

where :

- ▶ θ_k is the parameters vector at iteration k ,
- ▶ $\nabla f(\theta_k)$ is the gradient of the function f en θ_k ,
- ▶ α is the learning rate (i.e. descent step).

Reminder: Gradient Descent method - Adam algorithm

- ▶ **Adam** (*Adaptive Moment Estimation*) is a stochastic optimisation method.
- ▶ The parameters update is the following:

$$\theta_{k+1} = \theta_k - \alpha \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}$$

where:

- ▶ $\hat{m}_k$ is the gradient moving average (first order moment),
 - ▶ $\hat{v}_k$ is the moving average of the square of the gradient (second order moment),
 - ▶ α is the learning rate, and ϵ is a small regularization term to escape to divide by zero.
- ▶ Adam adjusts dynamically the learning rate in function of the first and second moments.
 - ▶ Converges faster than standard descent gradient for many problems.

Reminder: Newton method

- ▶ The Newton method uses information of the function curvature with the Hessian matrix:

$$\theta_{k+1} = \theta_k - H_f^{-1}(\theta_k) \nabla f(\theta_k)$$

where :

- ▶ $H_f(\theta_k)$ is the Hessian matrix of f at θ_k point,
- ▶ $\nabla f(\theta_k)$ is the gradient of f at θ_k point.
- ▶ Characteristic: fast convergence near to the optimum, but expensive calculation to obtain the Hessian.

Reminder: Quasi-Newton method

- ▶ The quasi-Newton methods approximates the Hessian matrix instead of calculating it directly:

$$\theta_{k+1} = \theta_k - B_k^{-1} \nabla f(\theta_k)$$

where B_k is an approximation of the Hessian updating at every iteration.

- ▶ Famous method: BFGS algorithm, which updates B_k without directly calculating H_f .
- ▶ Characteristic: better compromise between precision and computation cost than the Newton method.

Reminder: Quasi-Newton method - L-BFGS algorithm

- ▶ **L-BFGS** (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) is a variant of BFGS method.
- ▶ It economizes the memory saving only few vectors to approximate the Hessian matrix, without needing to calculate it completely.
- ▶ The parameters update follows the quasi-Newtonienne rule:

$$\theta_{k+1} = \theta_k - B_k^{-1} \nabla f(\theta_k)$$

where B_k is a weak memory approximation of the Hessian matrix.

- ▶ Efficient for big scale problems.
- ▶ Approach more robust than classical gradient descent for complex problems (like ill-conditioning or non-linear problems).
- ▶ Needs less memory than BFGS with a good Hessian approximation.

Reminder: Comparison between optimization methods

- ▶ **Gradient Descent:**
 - ▶ Easy to implement, but slow convergence.
- ▶ **Newton method:**
 - ▶ Fast convergence but expensive calculation of the Hessian.
- ▶ **Quasi-Newton method:**
 - ▶ Compromise between convergence speed and calculation cost.

State-of-the-art: Adaptive weights

- ▶ As seen above, the PINN loss is generally:

$$\mathcal{L}(\theta) = w_i \mathcal{L}_{IC}(\theta) + w_b \mathcal{L}_{BC}(\theta) + w_r \mathcal{L}_{PDE}(\theta),$$

where the weights are fixed during the training

- ▶ For adaptive weights methods, the PINN loss became:

$$\mathcal{L}(\theta) = w_i(t) \mathcal{L}_{IC}(\theta) + w_b(t) \mathcal{L}_{BC}(\theta) + w_r(t) \mathcal{L}_{PDE}(\theta),$$

where t is the iteration variable. So the weights change every iteration during the training.

- ▶ The objective of these techniques is to balance the contribution of each loss term to improve the convergence of the loss.

State-of-the-art: Preconditioning

- ▶ The preconditioning is used to improve the problem conditioning for ill-conditioned problems, like the PINN optimization.
- ▶ As seen previously, the Hessian matrix of the PINN loss is ill-conditioned for many problems. Due to this, the Gradient Descent methods like Adam are not efficient.
- ▶ On the other hand, the Newton and Quasi-Newton methods are very efficient for ill-conditioned problems because they contains preconditioning with the Hessian inverse in their line search:

$$\theta_{k+1} = \theta_k - H_f^{-1}(\theta_k) \nabla f(\theta_k)$$

State-of-the-art: Gradient projection

- ▶ The gradient projection is used to resolve or attenuate the gradient conflict by changing gradient directions for a more efficient optimization.
- ▶ For example, if $\nabla_{\theta}\mathcal{L}_{BC}$ and $\nabla_{\theta}\mathcal{L}_{PDE}$ are in conflicting directions, the orthogonal projection of $\nabla_{\theta}\mathcal{L}_{BC}$ on $\nabla_{\theta}\mathcal{L}_{PDE}$ is:

$$\nabla_{\theta,proj}\mathcal{L}_{BC} = \nabla_{\theta}\mathcal{L}_{BC} - \frac{\nabla_{\theta}\mathcal{L}_{PDE} \cdot \nabla_{\theta}\mathcal{L}_{BC}}{\|\nabla_{\theta}\mathcal{L}_{PDE}\|^2} \nabla_{\theta}\mathcal{L}_{PDE}$$

Comparison between PINN and data fitting

- ▶ Comparison between PINN (physics) and classical neural network (data)
- ▶ Finite elements method to create a reference solution
- ▶ Train set: 3000 points, test set: 9000 points
- ▶ Training: Adam 5000 iterations + L-BFGS 25000 iterations
- ▶ We use a model of 11404 parameters (4 hidden layers of 60 neurons themselves)

Summary of first and second year studies

We did the following studies:

- ▶ Analyze the PINN convergence problems and the cause of them (Hessian's conditioning, directional conflict)
- ▶ Neural network expressivity
- ▶ Comparison between PINN and data fitting
- ▶ Labelization of each loss
- ▶ Imposition of boundary conditions
- ▶ Natural gradient descent
- ▶ Nash games
- ▶ Co-clustering

Imposition of boundary conditions³

- ▶ Transform the classical problem $\min_{\theta} \mathcal{L}_{PINN} = \mathcal{L}_{PDE} + \mathcal{L}_{BC}$ to the problem $\min_{\theta} \mathcal{L}_{PDE}$.
- ▶ For this, we develop a reformulated solution:

$$u = g + \phi u_{NN}$$

where:

- ▶ u : reformulated solution,
- ▶ g : boundary condition,
- ▶ ϕ : distance function,
- ▶ u_{NN} : PINN solution.
- ▶ Advantage:
 - ▶ Boundary conditions directly imposed $\Rightarrow$ No gradient conflict.
- ▶ Inconvenients:
 - ▶ Impossibility to respect the boundary conditions at the intersection of different borders due to discontinuities.
 - ▶ Very long computation time compared to the classical approach of PINNs.

³Sukumar, N., Srivastava, A. (2022).

Natural gradient descent⁴

Objective: Minimize $\mathcal{L}(\theta)$ taking into account the geometry of the parameter space.

1. Classical gradient

$$g = \nabla_{\theta} \mathcal{L}(\theta) \in \mathbb{R}^p$$

2. Gram matrix (General case)

$$G = D^2 E(u_{\theta})(\partial_{\theta_i} u_{\theta}, \partial_{\theta_j} u_{\theta}), E(u_{\theta}) = \mathcal{L}(\theta)$$

3. Gram matrix (for data fitting problem specifically)

$$G = J^{\top} J \in \mathbb{R}^{p \times p}, J = \frac{\partial u_{\theta}(X)}{\partial \theta} \in \mathbb{R}^{n \times p}$$

4. Natural gradient descent

$$\tilde{g} = G^+ g$$

where G^+ is the pseudo-inverse of G .

5. Update

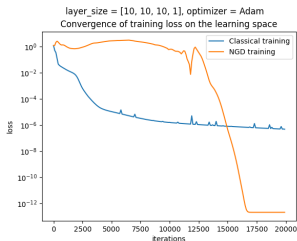
$$\theta \leftarrow \theta - \eta \tilde{g}$$

⁴Müller, J., Zeinhofer, M. (2023).

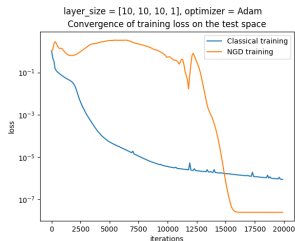
Natural gradient descent (NGD)

Problem: Minimize $\mathcal{L}(\theta) = \text{MSE}(u_\theta, u)$, with:

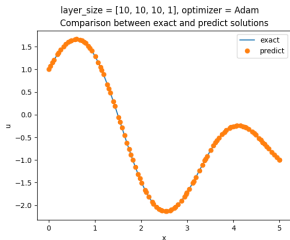
$$u(x) = \sin(2x) + \cos(x) - x(\sin(10) + \cos(5) + 1)/5.$$



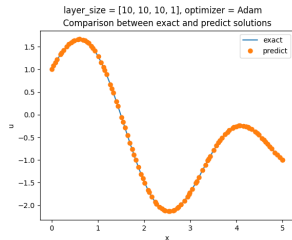
(a) Training loss



(b) Test loss



(c) Classical training



(d) NGD

Research axis: Extended Nash Games criteria

- We still work to the 2D Heated Cavity problem.
- Now, each component loss (for each PDE and boundary condition residuals) is considered as a player (9 players in this case).
- We define 3 different criteria for the territory split of the parameter space.
- We update the territory split at every iteration during training.

First criteria : Contribution of parameters for each loss

- ▶ Transform the classical problem : $\min_{\theta} \mathcal{L}_{PINN}$ to the following problem :

$$\begin{cases} \min_{\theta_c} \mathcal{L}_{PINN} \\ \min_{\theta_k} \mathcal{L}_k \end{cases}$$

- ▶ Splitting : $\theta = \theta_c \cup \bigcup_{k=1}^m \theta_k$ where :

- ▶ $\theta_c = \{\theta_i\}_i$ such as :

$$\frac{\partial \mathcal{L}_j}{\partial \theta_i} \cdot \frac{\partial \mathcal{L}_l}{\partial \theta_i} > 0, \forall j, l \in \{1, \dots, m\} \quad (1)$$

- ▶ $\theta_k = \{\theta_i\}_i$ such as (1) not satisfied and :

$$k = \arg \max_j \left| \frac{\partial \mathcal{L}_j}{\partial \theta_i} \right| \quad (2)$$

Second criteria : Relative contribution of parameters for each loss

- ▶ Transform the classical problem : $\min_{\theta} \mathcal{L}_{PINN}$ to the following problem :

$$\begin{cases} \min_{\theta_c} \mathcal{L}_{PINN} \\ \min_{\theta_k} \mathcal{L}_k \end{cases}$$

- ▶ Splitting : $\theta = \theta_c \cup \bigcup_{k=1}^m \theta_k$ where :
 - ▶ $\theta_c = \{\theta_i\}_i$ such as :

$$\frac{\partial \mathcal{L}_j}{\partial \theta_i} \cdot \frac{\partial \mathcal{L}_l}{\partial \theta_i} > 0, \forall j, l \in \{1, \dots, m\} \quad (3)$$

- ▶ $\theta_k = \{\theta_i\}_i$ such as (3) not satisfied and :

$$k = \arg \max_j \frac{\left| \frac{\partial \mathcal{L}_j}{\partial \theta_i} \right|}{\|\nabla_{\theta} \mathcal{L}_j\|} \quad (4)$$

Third criteria: Directional conflict between losses

- ▶ Transform the classical problem : $\min_{\theta} \mathcal{L}_{PINN}$ to the following problem :

$$\begin{cases} \min_{\theta_c} \mathcal{L}_{PINN} \\ \min_{\theta_k} \mathcal{L}_k \end{cases}$$

- ▶ Splitting : $\theta = \theta_c \cup \bigcup_{k=1}^m \theta_k$ where :

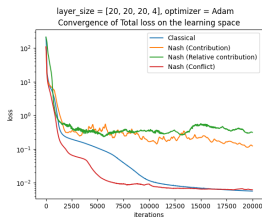
- ▶ $\theta_c = \{\theta_i\}_i$ such as :

$$\frac{\partial \mathcal{L}_j}{\partial \theta_i} \cdot \frac{\partial \mathcal{L}_l}{\partial \theta_i} > 0, \forall j, l \in \{1, \dots, m\} \quad (5)$$

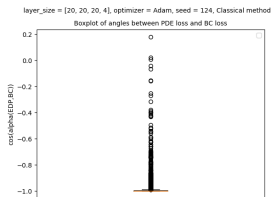
- ▶ $\theta_k = \{\theta_i\}_i$ such as (5) not satisfied and :

$$k = \arg \max_j \sum_{d=1}^m \frac{\partial \mathcal{L}_j}{\partial \theta_i} \cdot \frac{\partial \mathcal{L}_d}{\partial \theta_i} \quad (6)$$

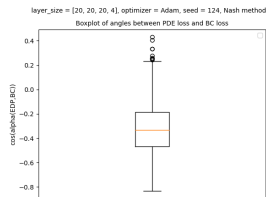
Results



Observation: Nash games with conflict criteria improve the initial convergence of Adam optimizer



(a) Angles cosine between $\nabla \mathcal{L}_{PDE}$ and $\nabla \mathcal{L}_{BC}$
(Classical method)



(b) Angles cosine between $\nabla \mathcal{L}_{PDE}$ and $\nabla \mathcal{L}_{BC}$
(Nash method)

Spectral co-clustering: Transformations

Directional-conflict-oriented transformations: Let $X \in \mathbb{R}^{M \times P}$ be the matrix (residuals $\times$ parameters). We apply one of the transformations $T : \mathbb{R}^{M \times P} \rightarrow \mathbb{R}^{M \times P}$:

$$T_{\text{colL2}} : X_{ij} \leftarrow \frac{X_{ij}}{\|X_{\cdot j}\|_2},$$

$$T_{\text{rowC+colL2}} : X \leftarrow X - \mu_{\text{row}}(X), \quad \text{then colL2},$$

$$T_{\text{sign}} : X_{ij} \leftarrow \text{sign}(X_{ij}),$$

$$T_{\text{zscore+L2}} : X \leftarrow \frac{X - \mu_{\text{col}}(X)}{\sigma_{\text{col}}(X)}, \quad \text{then colL2}.$$

Here, $\mu_{\text{row}}(X)$ is the vector of row means (one mean per row), $\mu_{\text{col}}(X)$ is the vector of column means (one mean per column), and $\sigma_{\text{col}}(X)$ is the vector of column standard deviations.

Spectral co-clustering

Spectral co-clustering:

1. *Input matrix and transformation:* Let $X_T = T(X) \in \mathbb{R}^{M \times P}$ be the transformed data matrix (rows = U , columns = V).
2. *Row/column degree matrices:*

$$D_r = \text{diag}(X_T \mathbf{1}_P), \quad D_c = \text{diag}(X_T^\top \mathbf{1}_M)$$

where D_r and D_c contain row and column sums.

3. *Bipartite normalization:*

$$\hat{X} = D_r^{-1/2} X_T D_c^{-1/2}$$

4. *Spectral analysis via SVD:* Compute the singular value decomposition

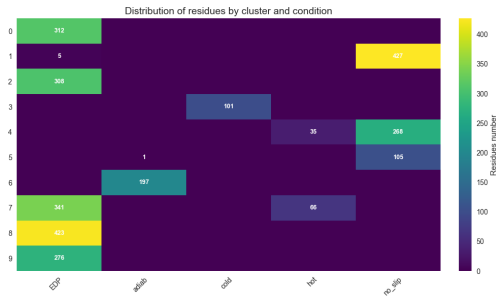
$$\hat{X} = U \Sigma V^\top$$

and keep the top- K singular vectors (largest singular values), excluding the trivial first component.

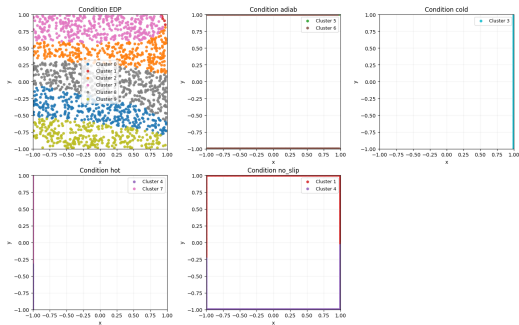
5. *Simultaneous clustering:* Apply k -means to the K -dimensional row embeddings (rows of U_K) and column embeddings (rows of V_K) to obtain:

$$\ell_{\text{rows}} \in \{1, \dots, K\}^M, \quad \ell_{\text{cols}} \in \{1, \dots, K\}^P.$$

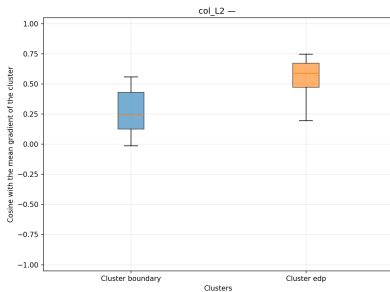
Spectral co-clustering : Results



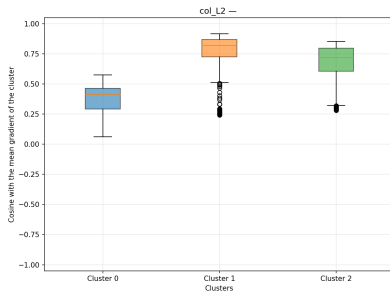
Heated Cavity 2D — colored conditions by cluster



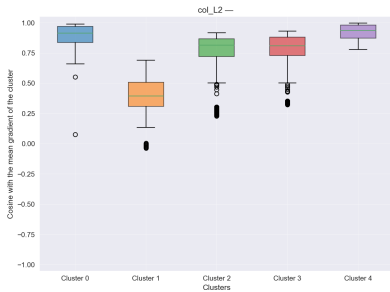
Spectral co-clustering : Results



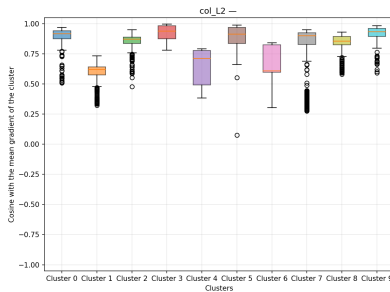
(a) Classical training



(b) 3 clusters



(c) 5 clusters



(d) 10 clusters

Research axis: Nash Games Training

- ▶ Now, we interest us at Nash games to train the PINN
- ▶ Laplace 1D :

$$\frac{d^2 u}{dx^2} = -(4 \sin(2x) + \cos(x)), x \in [0, 5] \quad (7)$$

$$u(0) = 1, u(5) = -1 \quad (8)$$

- ▶ Analytical solution :
$$u(x) = \sin(2x) + \cos(x) - \frac{\sin(10) + \cos(5) + 1}{5} x$$
- ▶ 3 Nash players: $\mathcal{L}_{PDE}$, $\mathcal{L}_{BC}$, $\mathcal{L}_{PINN}$

Research axis: Nash Games Training

- Transform the classical problem : $\min_{\theta} \mathcal{L}_{PINN}$ to the following problem :

$$\begin{cases} \min_{\theta_c} \mathcal{L}_{PINN} \\ \min_{\theta_p} \mathcal{L}_{PDE} \\ \min_{\theta_b} \mathcal{L}_{BC} \end{cases}$$

- Splitting : $\theta = (\theta_c, \theta_p, \theta_b)$ where :

- $\theta_c = \{\theta_i\}_i$ such as :

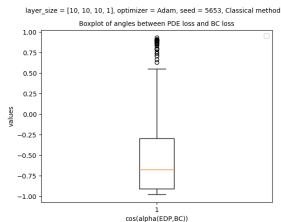
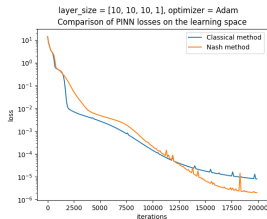
$$\frac{\partial \mathcal{L}_{PDE}}{\partial \theta_i} \cdot \frac{\partial \mathcal{L}_{BC}}{\partial \theta_i} > 0 \quad (9)$$

- $\theta_p = \{\theta_i\}_i$ such as (9) not satisfied and :

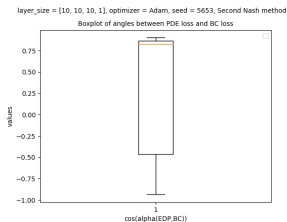
$$\frac{\left| \frac{\partial \mathcal{L}_{PDE}}{\partial \theta_i} \right|}{\|\nabla_{\theta} \mathcal{L}_{PDE}\|} > \frac{\left| \frac{\partial \mathcal{L}_{BC}}{\partial \theta_i} \right|}{\|\nabla_{\theta} \mathcal{L}_{BC}\|} \quad (10)$$

- $\theta_b = \{\theta_i\}_i$ such as (9) and (10) not satisfied

Research axis: Nash Games Training



(a) Angles cosine between $\nabla \mathcal{L}_{PDE}$ and $\nabla \mathcal{L}_{BC}$
(Classical method)



(b) Angles cosine between $\nabla \mathcal{L}_{PDE}$ and $\nabla \mathcal{L}_{BC}$
(Nash method)

Other future researches: MGDA

Inputs : $x_0, \epsilon, T_{\max}, \alpha_t$

Output : x^*

1. Initialization : $x_t \leftarrow x_0$
2. For each iteration $t = 0, 1, \dots, T_{\max}$:
 - 2.1 Calculate $\nabla L_i(x_t)$ for all i
 - 2.2 Resolve :

$$\min_{d_t} \|d_t\|^2 \quad \text{under constraint} \quad \nabla L_i(x_t)^T d_t \leq 0, \forall i$$

- 2.3 If $\|d_t\| \leq \epsilon$, end and return x_t
 - 2.4 Else, updating the solution :

$$x_{t+1} \leftarrow x_t + \alpha_t d_t$$

- 2.5 Update $t \leftarrow t + 1$
3. End if the convergence is reached.

General Principle of MGDA

The central idea of MGDA is to search for a **common descent direction** for all objectives.

For each objective f_i , we compute its gradient:

$$g_i = \nabla_{\theta} f_i(\theta).$$

We seek a direction of the form:

$$d = \sum_{i=1}^m \alpha_i g_i,$$

where the coefficients α_i are chosen so that d is a descent direction for all objectives.

Geometrically, MGDA selects the **minimum-norm element** of the convex hull of the gradients $\{g_i\}$.

Limitations of Classical MGDA

Although MGDA provides a multi-objective descent direction, several limitations arise in the context of PINNs:

- ▶ the direction is dominated by the gradient with the smallest norm;
- ▶ objectives with large errors may have small gradients;
- ▶ in high dimension, the MGDA direction norm decreases significantly;
- ▶ conflicts between objectives are not explicitly handled.

These issues lead to slow or unstable convergence.

Idea: Equalizing the Objectives

An alternative approach is to search for a direction d such that the objectives tend to become equal after the update.

Using a first-order approximation:

$$f_i(\theta - pd) \approx f_i(\theta) - p\langle g_i, d \rangle.$$

We impose linearized equality:

$$f_1 - p\langle g_1, d \rangle = f_2 - p\langle g_2, d \rangle = \cdots = f_m - p\langle g_m, d \rangle.$$

Substituting $d = \sum_j \alpha_j g_j$ yields a linear system in the coefficients α_j .

Matrix Formulation

Define the gradient correlation matrix:

$$W_{ij} = \langle g_i, g_j \rangle, \quad W = Jf Jf^\top.$$

The equalization condition becomes:

$$f_i - f_k = \alpha^\top (W_i - W_k), \quad \forall i, k.$$

An equivalent formulation is to solve:

$$\min_{\alpha} f_1 - \alpha^\top W_1$$

subject to the linear constraints:

$$f_i - f_{i+1} = \alpha^\top (W_i - W_{i+1}), \quad i = 1, \dots, m-1.$$

This approach explicitly aims at **equalizing the objectives**.