CentraleSupélec

Hybrid AI - Rapport

Auteurs:

Marc-César Garcia-Grenet
marc-cesar.garcia-grenet@student-cs.fr

Thomas Bodart
thomas.bodart@student-cs.fr

Projet GitHub:

lien vers le projet GitHub

April 2025

1 Introduction

Dans ce rapport, nous étudions différentes méthodes numériques pour résoudre l'équation de Poisson stationnaire à deux dimensions sur le domaine unité $\Omega = [0,1]^2$, avec des conditions aux limites de Dirichlet homogènes. L'objectif est d'évaluer et de comparer deux approches :

- un modèle d'apprentissage supervisé basé sur un perceptron multicouche (MLP),
- un réseau de neurones informé par la physique (PINN).

Le rapport détaille tout d'abord la discrétisation par différences finies, qui sert de méthode de référence. Ensuite, les méthodes MLP et PINN sont décrites en précisant leurs architectures, stratégies d'entraînement et formulations mathématiques. Enfin, une évaluation comparative est menée sur la base de plusieurs critères : précision numérique, coût d'entraînement, et temps d'inférence.

Ce travail permet de mieux comprendre les forces et faiblesses des méthodes d'apprentissage profond appliquées à la résolution d'équations aux dérivées partielles (EDP), dans le contexte de données simulées.

2 Discrétisation de l'équation de Poisson par différences finies

2.1 Problème continu

Nous considérons l'équation de Poisson sur le domaine carré $\Omega = [0,1]^2$ avec des conditions aux limites de Dirichlet homogènes :

$$\begin{aligned} -\Delta u(x,y) &= f(x,y), \quad \text{pour } (x,y) \in \Omega, \\ u(x,y) &= 0, \quad \text{pour } (x,y) \in \partial\Omega. \end{aligned}$$

The source term is defined by:

$$f(x,y) = x\sin(a\pi y) + y\sin(b\pi x),$$

where a and b are parameters controlling the frequency of the solution.

2.2 Maillage du domaine

Le domaine Ω est discrétisé en un maillage uniforme de $N \times N$ points intérieurs, avec un pas $h = \frac{1}{N+1}$. Les points du maillage sont donnés par :

$$x_i = ih, \quad y_j = jh, \quad \text{pour } i,j = 1,2,\ldots,N.$$

2.3 Approximation du Laplacien

La méthode des différences finies centrées d'ordre 2 est utilisée pour approximer le Laplacien. Pour un point intérieur (x_i, y_j), l'approximation est :

$$-\Delta u(x_i, y_j) \approx -\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} - \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2}.$$

En combinant les termes, on obtient :

$$\frac{1}{h^2}\left(-u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} + 4u_{i,j}\right) = f_{i,j}.$$

2.4 Système linéaire résultant

En réécrivant cette équation pour tous les points intérieurs, on obtient un système linéaire de la forme :

$$A\mathbf{u} = \mathbf{f},$$

où :

- $\mathbf{u}$ est le vecteur des valeurs inconnues $u_{i,j}$ au sein du domaine discretisé,
- $\mathbf{f}$ est le vecteur des valeurs de $f_{i,j}$,
- A est une matrice creuse de taille $N^2 \times N^2$, représentant l'opérateur Laplacien discretisé.

La matrice A est construite à l'aide du produit de Kronecker :

$$A = \frac{1}{h^2}(I_N \otimes T + T \otimes I_N),$$

où :

- I_N est la matrice identité de taille N,
- T est une matrice tridiagonale définie par :

$$T = \begin{bmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{bmatrix}.$$

2.5 Conditions aux limites

Les conditions de Dirichlet homogènes ($u = 0$ sur $\partial\Omega$) sont incorporées en fixant les valeurs de $u_{i,j}$ à zéro pour les indices correspondant aux bords du domaine. Ainsi, seules les valeurs intérieures sont inconnues et sont résolues via le système linéaire ci-dessus.

2.6 Résolution numérique

Le système linéaire obtenu est résolu numériquement à l'aide de solveurs adaptés aux matrices creuses, tels que ceux disponibles dans la bibliothèque SciPy en Python. Cette approche permet une résolution efficace même pour des maillages de grande taille.

2.7 Génération de données

Pour entraîner et évaluer les modèles, un jeu de données synthétique a été généré en résolvant l'équation de Poisson pour différentes valeurs des paramètres a et b. Le domaine Ω est discrétisé en une grille uniforme de 50×50 points (soit 2500 points intérieurs). Pour chaque couple (a, b), avec $a, b \in \{0, 1, \ldots, 9\}$, les étapes suivantes sont réalisées :

1. Calcul du second membre $f(x, y)$ à l'aide de l'expression analytique.
2. Discrétisation de l'opérateur Laplacien à l'aide d'un schéma en différences finies d'ordre deux, menant à une matrice creuse A comme décrite précédemment.
3. Résolution du système linéaire $A\mathbf{u} = \mathbf{f}$ à l'aide d'un solveur direct pour matrices creuses de la bibliothèque `scipy`.
4. Sauvegarde de f et de la solution u obtenue pour chaque configuration.

Cette procédure permet d'obtenir un ensemble de 100 instances de problème, enregistré dans une archive NumPy compressée. La génération est entièrement automatisée en Python, à l'aide des bibliothèques scientifiques standards telles que `NumPy` et `SciPy`.

3 Méthodologie

3.1 Méthode des différences finies

La méthode FD (différences finies) sert de solution de référence. En utilisant le Laplacien discrétisé sur une grille régulière, on résout le système linéaire creux pour obtenir $u(x, y)$ pour chaque configuration du problème. Cette méthode est à la fois précise et efficace pour résoudre un seul cas.

L'objectif de cette méthode est de simuler des données réelles du problème étudié ; en l'absence de telles données, nous les générons numériquement.

3.2 Approche supervisée par réseau MLP

Le modèle d'apprentissage supervisé est un perceptron multicouche (MLP) entraîné pour approximer la correspondance entre les coordonnées (x, y) et la solution $u(x, y)$ de l'équation de Poisson. Ce modèle est entraîné à partir de données annotées obtenues par la méthode des différences finies (FD).

Architecture du modèle :

- Entrée : coordonnées 2D (x, y) dans $[0, 1]^2$
- Sortie : valeur scalaire $u(x, y)$
- Réseau : 3 couches entièrement connectées

  Entrée (2) $\rightarrow$ Dense(1024, ReLU) $\rightarrow$ Dense(512, ReLU) $\rightarrow$ Dense(1)

Stratégie d'entraînement :

- Données : paires (x_i, y_i) avec cibles $u_{\text{FD}}(x_i, y_i)$ issues de la méthode FD pour un couple (a, b) donné.
- Fonction de perte :

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{BC}}, \quad \mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_i |u_{\text{MLP}}(x_i, y_i) - u_{\text{FD}}(x_i, y_i)|^2$$

- Les points sur le bord $(x, y) \in \partial\Omega$ sont inclus pour forcer la condition aux limites $u(x, y) \approx 0$ via :

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{bd}}} \sum_j |u_{\text{MLP}}(x_j, y_j)|^2$$

- Optimiseur : Adam, taux d'apprentissage 10^{-3}, 1000 époques, taille de batch 1024.

Le modèle MLP est entraîné indépendamment pour chaque couple (a, b) à partir du jeu de données préalablement calculé. Une fois entraîné, le modèle offre une inférence rapide sans nécessiter de recalcul du terme source ou de dérivation.

3.3 Réseaux de neurones informés par la physique (PINNs)

Le modèle PINN intègre directement l'équation de Poisson dans son processus d'entraînement, sans recourir à des données annotées. La fonction de perte est construite à partir de la différentiation automatique pour imposer les contraintes physiques de l'équation aux dérivées partielles.

Architecture du modèle :

- Entrée : coordonnées spatiales (x, y)
- Sortie : valeur scalaire $u(x, y)$
- Réseau :

  Entrée (2) $\rightarrow$ Dense(256) $\rightarrow$ Dense(1024) $\rightarrow$ Dense(1024) $\rightarrow$ Dense(256) $\rightarrow$ Sortie (1)

  avec activation `tanh` à chaque couche cachée.

Stratégie d'entraînement :

- Les points d'entraînement sont échantillonnés dans le domaine Ω et sur sa frontière $\partial\Omega$.
- Le Laplacien Δu est calculé par différentiation automatique d'ordre deux du réseau par rapport aux entrées (x, y).
- Fonction de perte :

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{BC}}$$

avec :

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{int}}} \sum_i |\Delta u(x_i, y_i) + f(x_i, y_i)|^2$$

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N_{\text{int}}} \sum_i |u(x_i, y_i) - u_{\text{FD}}(x_i, y_i)|^2$$

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{\text{bd}}} \sum_j |u(x_j, y_j)|^2$$

- Entraînement : 500 époques, batch de 1024, optimiseur Adam, taux d'apprentissage 10^{-3}.

Cette formulation permet au PINN de résoudre l'équation de Poisson même sans supervision explicite, tout en autorisant une comparaison facultative avec les solutions FD pour stabiliser l'apprentissage.

4 Résultats et Analyse

Nous comparons les performances des deux approches (MLP supervisé et PINN) selon trois axes : la précision numérique, le temps d'inférence et le temps d'entrainement.

4.1 Précision numérique

Nous évaluons les méthodes apprises (MLP et PINN) par rapport à la solution de référence FD, en utilisant deux métriques classiques :

- **Erreur L^2 :**

$$\|u_{\text{ML}} - u_{\text{FD}}\|_2 = \sqrt{\frac{1}{N^2} \sum_{i,j} \left(u_{\text{ML}}(x_i, y_j) - u_{\text{FD}}(x_i, y_j)\right)^2}$$

- **Erreur maximale :**

$$\|u_{\text{ML}} - u_{\text{FD}}\|_\infty = \max_{i,j} |u_{\text{ML}}(x_i, y_j) - u_{\text{FD}}(x_i, y_j)|$$

Résultats sur plusieurs cas :

Cas (a, b)	**Erreur** L^2 **(MLP / PINN)**	**Erreur Max (MLP / PINN)**
(3,3)	6.24×10^{-4} / 1.18×10^{-2}	2.48×10^{-3} / 1.99×10^{-2}
(4,5)	1.25×10^{-3} / 5.65×10^{-3}	4.04×10^{-3} / 2.00×10^{-2}
(8,4)	9.23×10^{-4} / 6.12×10^{-2}	2.78×10^{-3} / 9.24×10^{-2}
(9,9)	6.39×10^{-4} / 2.72×10^{-2}	1.99×10^{-3} / 5.33×10^{-2}

Analyse :

- Le modèle MLP obtient systématiquement de meilleures précisions que le PINN sur tous les cas testés.
- Les erreurs augmentent pour les cas à haute fréquence (ex. $a = 9$, $b = 9$), en particulier pour le PINN, ce qui traduit une plus grande difficulté à approximer des solutions fortement oscillantes.
- Le PINN reste cependant raisonnablement précis et peut-être utilisé sans données supervisées.

4.2 Temps d'entraînement

Les temps d'entraînement (mesurés sur CPU/GPU selon disponibilité) sont indiqués ci-dessous pour quatre cas typiques :

Cas (a, b)	**MLP (s)**	**PINN (s)**
(3,3)	67.75	100.04
(4,5)	75.00	104.67
(8,4)	81.00	100.33
(9,9)	66.26	116.60

Analyse :

- L'entraînement du PINN est systématiquement plus long, en raison du coût d'auto-différentiation pour le calcul du Laplacien.
- Le coût du MLP est relativement stable mais dépend du contenu de l'ensemble d'apprentissage (ici fixe à un cas (a, b) à la fois).

4.3 Temps d'inférence

Le tableau suivant compare les temps d'inférence (évaluation de $u(x, y)$ sur une grille 52×52) :

Cas (a, b)	**MLP (s)**	**PINN (s)**
(3,3)	0.065	0.009
(4,5)	0.003	0.008
(8,4)	0.004	0.008
(9,9)	0.003	0.008

Analyse :

- Le PINN offre le temps d'inférence le plus faible dans le cas a=3 et b=3 légèrement devant le MLP mais est pour le reste plus lent que le MLP.

4.4 Visualisation 2D des résultats

Les figures suivantes illustrent les solutions obtenues par les trois méthodes (FD, MLP, PINN) pour deux cas représentatifs : un cas "simple" avec faible fréquence ($a = b = 3$) et un cas "complexe" avec forte fréquence ($a = b = 9$).

Cas à faible fréquence ($a = 3$, $b = 3$)

Ce cas correspond à une source $f(x, y)$ peu oscillante. Les réseaux de neurones, qu'ils soient supervisés (MLP) ou non supervisés (PINN), parviennent à bien approximer la solution de référence fournie par la méthode des différences finies. Les écarts sont faibles, tant visuellement que numériquement.

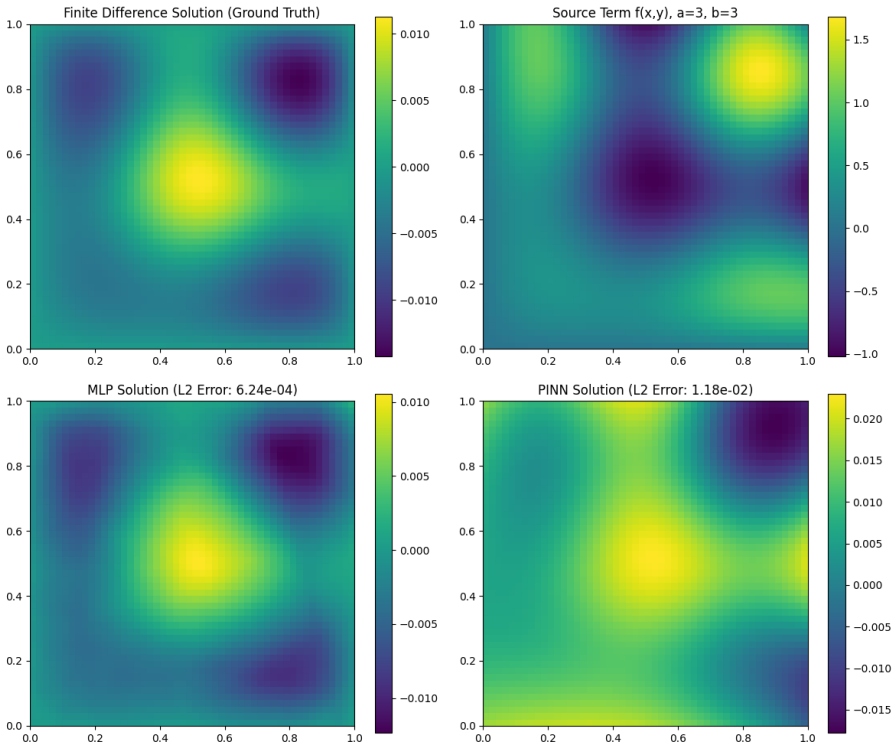

Figure 1: Comparaison des solutions obtenues pour $a = 3$, $b = 3$ avec les méthodes FD (référence), MLP et PINN.

Cas à forte fréquence ($a = 9$, $b = 9$)

Dans ce cas, la source $f(x, y)$ présente de fortes oscillations, rendant la solution plus complexe à approximer. Le MLP parvient encore à capturer les grandes structures de la solution, bien qu'avec une légère perte de précision. Le PINN, en revanche, montre des difficultés plus marquées, avec des écarts significatifs par rapport à la solution de référence, notamment dans les zones fortement oscillantes.

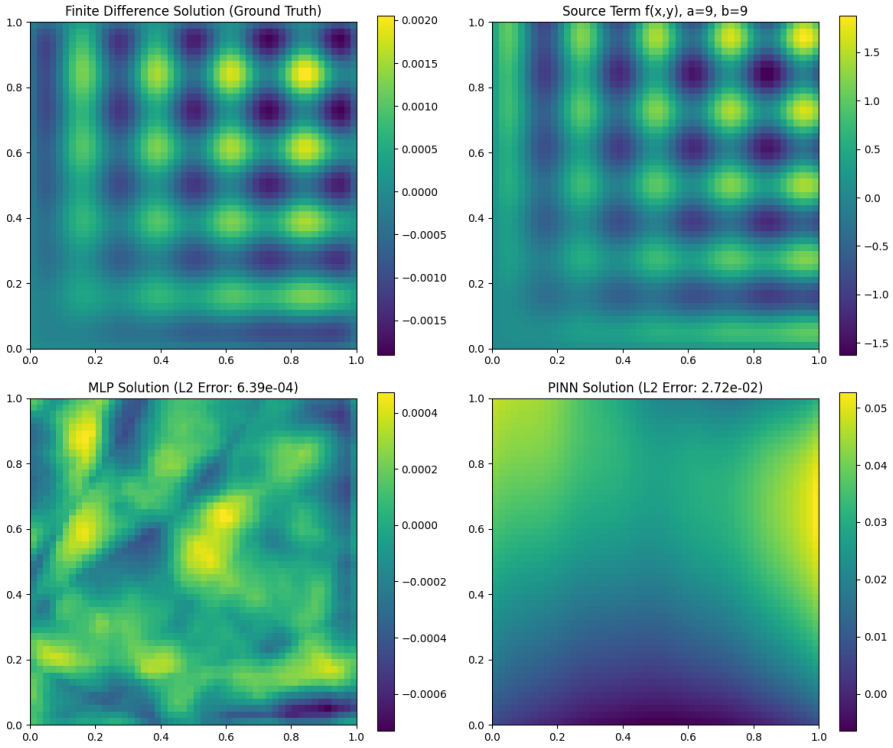

Figure 2: Comparaison des solutions obtenues pour $a = 9$, $b = 9$ avec les méthodes FD (référence), MLP et PINN.

5 Conclusion

Ce travail a permis de comparer trois approches pour résoudre numériquement l'équation de Poisson : une méthode classique (FD), un réseau de neurones supervisé (MLP) et une méthode basée sur les principes physiques (PINN).

Résumé des résultats

- Le MLP atteint systématiquement une précision supérieure à celle du PINN sur les cas testés, avec des erreurs L^2 inférieures à 10^{-3} contre 10^{-2} pour le PINN.
- Le coût d'entraînement du MLP est environ 30% plus faible que celui du PINN, principalement dû à l'absence de dérivées dans la fonction de coût.
- En inférence, le PINN est le plus rapide uniquement dans le cas a=3 et b=3, légèrement devant le MLP, mais les deux restent très efficaces.

Limites et critiques constructives

Malgré sa capacité à intégrer directement les équations physiques, le PINN montre des limites :

- L'approximation du Laplacien par différentiation automatique introduit des instabilités numériques.
- La convergence du PINN est plus lente et sensible aux hyperparamètres.
- Les erreurs du PINN augmentent fortement avec la fréquence du problème (grands a, b), traduisant une difficulté à capturer des comportements oscillants fins.

Le MLP, bien qu'efficace, dépend fortement de la qualité et de la diversité des données d'entraînement issues de la méthode FD. Il ne généralise pas à des valeurs de paramètres (a, b) non vues pendant l'apprentissage.

Propositions d'amélioration

Pour améliorer les performances, plusieurs pistes sont envisageables :

- **Pour le PINN :**
  - Employer des stratégies d'échantillonnage adaptatif pour mieux couvrir les zones complexes du domaine.
  - Introduire un équilibrage dynamique entre les différentes composantes de la fonction de coût (PDE / BC / MSE).
- **Pour le MLP :**
  - Utiliser des techniques de data augmentation sur les solutions FD.
  - Envisager un modèle conditionné par les paramètres (a, b) afin d'améliorer la généralisation.

Perspectives

Au-delà des approches testées ici, plusieurs méthodes pourraient être explorées :

- **Méthodes hybrides** combinant solveurs numériques traditionnels et réseaux profonds, pour réduire les erreurs tout en maintenant la structure physique.
- **Apprentissage multitâche** pour partager des représentations entre plusieurs cas (a, b), réduisant ainsi le besoin de réentraînement.