Une analyse des : Physics Informed Neural Network (PINNs)

Lila Chakri[1] , **Oihan Cordelier**[2] ,
[1]Étudiante, INF8225
[2]Étudiant, INF8225

Abstract

Dans ce projet, nous explorons l'application des Physics-Informed Neural Networks (PINNs) pour la résolution d'équations différentielles partielles complexes, notamment les équations de Navier-Stokes et de Schrödinger. Afin d'améliorer la stabilité et la précision des approximations, nous avons intégré des blocs résiduels (ResBlocks) dans l'architecture des réseaux. Les résultats expérimentaux montrent que l'utilisation des ResBlocks permet une meilleure convergence et une plus grande fidélité dans la prédiction des dynamiques physiques, en particulier pour des systèmes non linéaires complexes. Toutefois, des défis persistent, notamment la sensibilité aux hyperparamètres et les coûts computationnels élevés. Ce travail met en évidence le potentiel des PINNs enrichis par des architectures profondes et ouvre la voie à de futures améliorations pour la modélisation précise de phénomènes physiques variés.

1 Introduction

Les équations différentielles, qu'elles soient ordinaires (EDO) ou aux dérivées partielles (EDP), sont fondamentales pour modéliser des phénomènes physiques, biologiques et d'ingénierie. Leur résolution repose sur des méthodes numériques telles que les différences finies (FDM), les éléments finis (FEM) ou les volumes finis (FVM), qui nécessitent une discrétisation du domaine. Toutefois, ces méthodes deviennent coûteuses et complexes en haute dimension ou pour des géométries irrégulières.

Avec l'évolution de l'apprentissage profond, de nouvelles approches sans maillage ont émergé. Parmi les premières tentatives, le *Deep Ritz Method* [E and Yu, 2017] a proposé d'utiliser un réseau de neurones pour approximer la solution d'un problème variationnel, en minimisant une énergie associée à l'équation différentielle. Le *Deep Galerkin Method* (DGM) [Sirignano and Spiliopoulos, 2018], quant à lui, utilise une stratégie similaire, mais s'appuie directement sur la formulation forte de l'équation, en échantillonnant des points de collocation et en imposant l'annulation du résidu de l'équation différentielle.

C'est dans ce contexte que s'inscrivent les *Physics-Informed Neural Networks* (PINNs), introduits par Raissi, Perdikaris et Karniadakis [Raissi *et al.*, 2019]. Les PINNs intègrent explicitement les lois physiques — exprimées sous forme d'EDO ou d'EDP — dans la fonction de perte du réseau de neurones. Grâce à la différentiation automatique, ils minimisent à la fois l'erreur sur les données observées et le résidu physique, sans nécessiter de maillage spatial.

Les PINNs offrent une flexibilité remarquable : ils peuvent résoudre des problèmes directs (forward problems), inverses (inverse problems) ou identifier des paramètres inconnus à partir de données expérimentales. Ils ont montré leur efficacité dans des domaines variés, tels que la mécanique des fluides, la diffusion thermique et la dynamique des structures. Dans ce projet, nous explorerons les fondements théoriques des PINNs, leur mise en œuvre pratique et analyserons leurs performances et leurs limitations.

2 Approche théorique

2.1 Principe fondamental des PINNs

Les Physics-Informed Neural Networks (PINNs) sont une approche innovante qui combine l'apprentissage profond avec les connaissances physiques a priori pour résoudre des équations différentielles. Contrairement aux méthodes classiques de résolution numérique, qui nécessitent une discrétisation explicite du domaine et l'utilisation de maillages, les PINNs exploitent la capacité des réseaux de neurones à approximer des fonctions continues et différentiables dans des espaces complexes.

Un PINN se compose d'un réseau de neurones qui apprend une fonction $u_\theta(x, t)$ représentant la solution d'une équation différentielle, tout en respectant les contraintes physiques imposées par cette équation. La principale innovation des PINNs réside dans l'intégration directe des équations différentielles dans la fonction de perte utilisée lors de l'entraînement du réseau.

Fonction de perte

La fonction de perte des PINNs se compose de deux parties principales :

- Erreur sur les données observées ($\mathcal{L}_{data}$)
- Erreur sur les lois physiques ($\mathcal{L}_{physique}$)

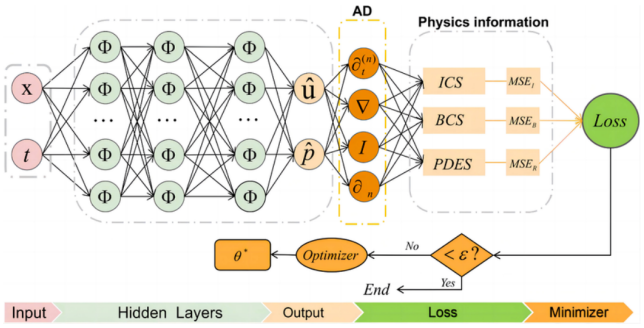

Figure 1: architecture d'un PINN classique [Wang *et al.*, 2024]

La fonction de perte globale est donc :

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \mathcal{L}_{physique}$$

où λ est un facteur de pondération.

2.2 Architecture des PINNs

Les PINNs sont généralement implémentés avec des réseaux de neurones multicouches (MLP) sans convolutions. Chaque réseau $u_\theta(x,t)$ est constitué de plusieurs couches entièrement connectées avec des fonctions d'activation non linéaires, typiquement $\tanh$ ou ReLU.

Auto-différentiation

L'auto-différentiation permet de calculer automatiquement les dérivées nécessaires à la résolution des équations différentielles, directement durant l'entraînement.

2.3 La fonction de perte dans les PINNs

Erreur sur les données ($\mathcal{L}_{data}$)

$$\mathcal{L}_{data} = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} |u_\theta(x_i,t_i) - u_{obs}(x_i,t_i)|^2$$

Erreur sur les lois physiques ($\mathcal{L}_{physique}$)

$$\mathcal{L}_{physique} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \frac{\partial u_\theta}{\partial t}(x_i,t_i) - \alpha \frac{\partial^2 u_\theta}{\partial x^2}(x_i,t_i) \right|^2$$

2.4 Extensions modernes des PINNs

- **PINNs Bayésiens** : Estimation de l'incertitude
- **cPINNs et XPINNs** : Décomposition du domaine
- **Fourier Neural Operators (FNO)** : Alternatives pour haute dimension
- **PINNs inverses** : Identification de paramètres ou lois physiques

2.5 Res-PINN

Une manière d'augmenter la capacité d'un réseau à interpoler des données plus complexes est d'augmenter sa profondeur, cependant un réseau profond peut souffrir du problème de disparation du gradient. Pour remédier à ce problème, il est possible d'utiliser des connections résiduelles [He *et al.*, 2015]. La figure 2 illustre un ResBlock qui consiste de 2

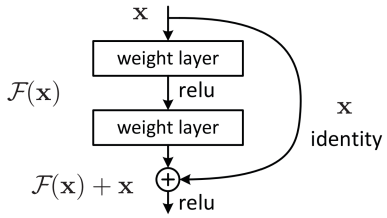

Figure 2: Schématisation d'une connexion résiduelle [He *et al.*, 2015]

couches cachées suivies par une connexion résiduelle. L'idée d'utiliser des connections résiduelles avec des PINNs a été proposée par [Cheng and Zhang, 2021] pour aider avec la propagation du gradient et donc de réduire l'erreur sur l'interpolation. À noter que dans l'implémentation proposée à la section 3, la fonction $\tanh$ est utilisée comme couche non linéaire à la place de la fonction ReLU.

3 Expériences

Afin de démontrer l'efficacité des PINNS ainsi que d'étudier l'influence de leurs hyper paramètres, 2 expériences du papier original [Raissi *et al.*, 2019] sont reproduites. La première porte sur l'équation de Schrödinger et vise à trouver la solution d'une EDP à partir de conditions frontières. La deuxième porte sur l'équation de Navier-Stokes autour d'un cylindre et vise à trouver les paramètres d'une EDP à partir d'une solution. Pour chacune des ces expériences, l'utilisation de connections résiduelles est étudiée [Cheng and Zhang, 2021]. Les reproductions de ces 2 expériences sont disponibles sur un GitHub public disponible à l'annexe A. Les figures associées avec les hyper paramètres testés y sont aussi présents. Ces implémentations sont fortement basées la librairie *pinns-torch* [Bafghi and Raissi, 2023].

3.1 Équation de Schrödinger

Pour cette expérience, les conditions initiales ainsi que les conditions frontières sont fournies au réseau. Une des fonctions de pertes vérifie si l'équation de Schrödinger est respectée. À partir de ces informations, le réseau interpole les valeurs à l'intérieur du domaine. Aucune valeur à l'intérieur du domaine n'est fournie au réseau pour son entraînement. La fonction non linéaire de Schrödinger, son domaine, ainsi que les conditions initiales et frontières sont :

$$f := ih_t + 0.5h_{xx} + |h|^2 h = 0, \qquad (1)$$
$$x \in [-5,5], \quad t \in [0, 0.5\pi]$$

$$h(0,x) = 2\text{sech}(x),$$
$$h(t,-5) = h(t,5),$$
$$h_x(t,-5) = h_x(t,5),$$

La fonction de perte utilise l'erreur quadratique moyenne (EQM) sur les conditions initiales et frontières en plus de la EQM sur l'EDP. Contrairement au papier original [Raissi *et al.*, 2019], l'EQM sur les conditions périodiques n'a pas était nécessaire pour obtenir des résultats aussi bon que l'état de

Table 1: Résultats d'entraînement sur Schrödinger

n_layers	hidden_dim	use_res	valid_loss
6	20	No	7.897E-02
6	**50**	**No**	**5.985E-04**
6	100	No	6.213E-04
6	50	Yes	1.495E-03
8	50	No	2.309E-03
8	50	Yes	7.408E-03
10	50	No	3.111E-03
10	50	Yes	4.598E-03

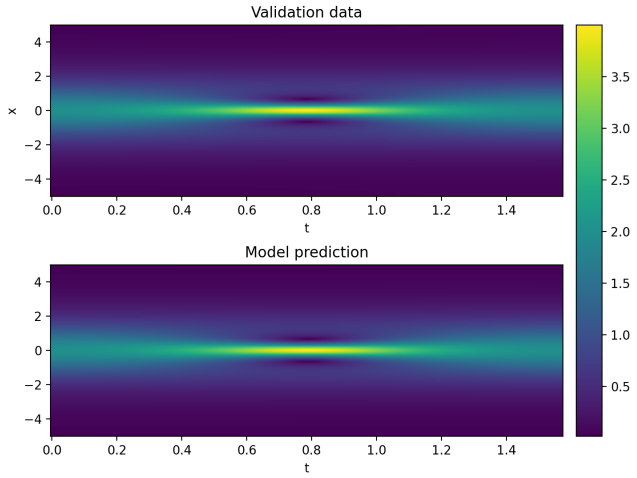

Figure 3: Validation de l'entraînement sur le problème de Schrödinger

l'art. Les conditions initiales sont vérifiées en utilisant 50 points ($N_0 = 50$) et l'équation de Schrödinger est vérifiée en utilisant 20 000 points ($N_f = 20000$).

$$MSE = MSE_0 + MSE_f \tag{2}$$

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2 \tag{3}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(_f^i, x_f^i)|^2 \tag{4}$$

Le tableau 1 énonce les différents hyper paramètres étudiés pour reproduire les résultats de l'état de l'art. Cette expérience illustre qu'utiliser une fonction de perte sur une EDP permet d'interpoler précisément une solution à partir d'un faible nombre de conditions initiales et frontières.

Les meilleurs résultats sur les données de validation ont été obtenu avec un réseaux de 6 couches sans connexions résiduelles. La valeur de la fonction perte obtenue avec ce réseau était de **5.9853E-4**, comparativement à l'état de l'art qui est de **1.97E-3**. La figure 3 compare les données de validation avec la prédiction du meilleur réseau obtenu. L'approfondissement du réseau ainsi que l'utilisation de couches résiduelles n'ont pas amélioré l'erreur de validation. Cela s'expliquerait par le fait que la complexité de cette tâche

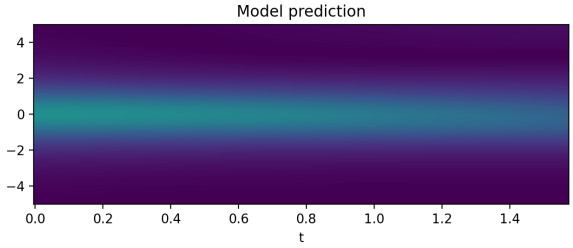

Figure 5: Validation de l'entraînement sans utilisation d'une fonction perte basée sur l'EDP

d'interpolation est assez simple pour qu'un réseau peu profond y performe bien. Les connexions résiduelles sont plus appropriées pour des réseaux profonds, nécessitant donc plus d'entraînement. La figure 4 illustre les courbes des fonctions pertes en fonction du nombre d'*epoch*. La *output loss*, la *validation loss* et la *PDE loss* sont la perte sur les conditions initiales, sur les données de validation et sur l'EDP de Schrödinger respectivement.

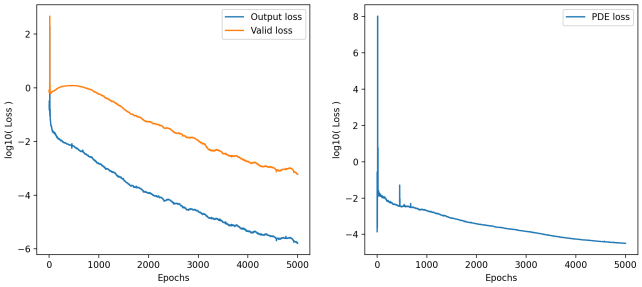

Figure 4: Courbe d'apprentissage sur le problème Schrödinger

Parmi les 2 méthodes d'optimisation étudiées, L-BFGS a obtenu de bien meilleurs résultats qu'ADAM présenté dans le tableau 2. Cela pourrait s'expliquer par la nature déterministe de ce problème puisque toutes les données d'entraînement étaient fournies au réseau lors de chaque passe avant. ADAM est plus approprié pour un problème stochastique où le réseau est entraîné par mini-batch.

Table 2: Résultats d'entraînement avec ADAM

lr	n_layers	hidden_dim	valid_loss
1e-5	6	50	6.614E-01
1e-3	6	50	1.166E+00

Comme illustrer par la figure 5, dans le cas où l'EDP n'est pas utilisée pour calculer la perte du réseau, celui-ci est incapable d'interpoler les données à l'intérieur du domaine. Cette expérience permet de valider l'utilisation d'une fonction perte basée sur une EDP définissant un phénomène physique permet d'obtenir une interpolation valide à partir de très peu de données.

3.2 Équation de Navier-Stokes

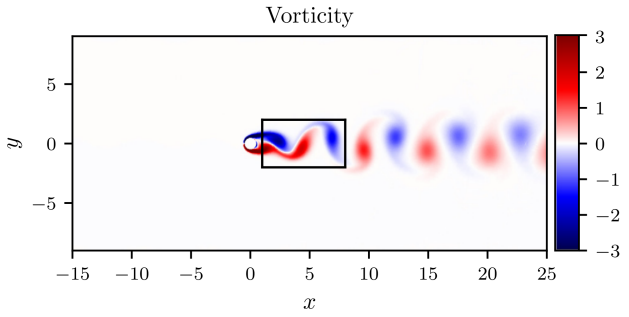

Figure 6: Vue d'ensemble du cas test Navier-Stokes [Raissi *et al.*, 2019]

Pour cette expérience, un écoulement turbulent incompressible est étudié dans la zone de sillage d'un cylindre de diamètre $D = 1$ tel qu'illustré par la figure 6. Le réseau est entraîné à prédire les composantes de vitesse du fluide $u(t, x, y)$ et $v(t, x, y)$ en plus de la pression $p(t, x, y)$. Contrairement à l'expérience sur l'équation de Schrödinger, le réseau doit aussi prédire des coefficients (λ_1, λ_2) de l'EDP qui sert aussi de fonction de perte. En d'autres termes, l'EDP est fournie de manière incomplète et le réseau doit la compléter tout en l'utilisant pour son entraînement. Le problème à résoudre par le réseau prend la forme :

$$\begin{aligned} u_t + \lambda_1(uu_x + vu_y) &= -p_x + \lambda_2(u_{xx} + u_{yy}) \\ v_t + \lambda_1(uv_x + vv_y) &= -p_y + \lambda_2(v_{xx} + v_{yy}) \\ u_x + v_y &= 0 \end{aligned} \tag{5}$$

La fonction de perte est la somme de l'EQM sur les composantes u, v en plus de l'EQM sur l'équation de Navier-Stokes. Les données de pression ne sont pas utilisées pour entraîner le modèle, elles se retrouvent dans l'équation de Navier-Stokes. Les fonctions de pertes se présentent comme :

$$\begin{aligned} f &:= u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy}) \\ g &:= v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy}) \\ MSE &:= \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right) \\ &+ \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i|^2 + |g(t^i, x^i, y^i|^2\right) \end{aligned}$$

Les paramètres λ se retrouvent à être des poids du modèle qui doivent être optimisés. La figure 7 illustre échantillonnage de 5000 points de collocation sur l'ensemble du domaine spatio-temporel utilisés pour l'entraînement du modèle. Ils représentent 5% des données totales disponibles. Le but de l'expérience est d'illustrer qu'avec un faible nombre données d'entraînement, le modèle peut correctement prédire la solution de l'EDP en plus de déterminer ses coefficients.

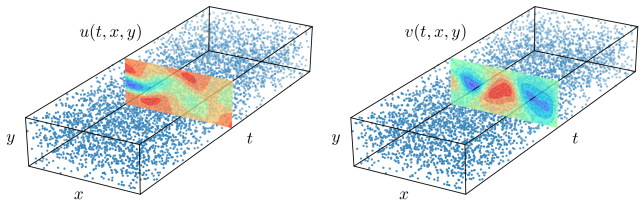

Figure 7: Illustration de l'échantillonnage sur le domaine spatio-temporel [Raissi *et al.*, 2019]

Table 3: Résultats des entraînements sur Navier-Stokes

lr	n_lay	hid_d	res	λ_1	λ_2	valid_loss
1.0E-03	12	20	No	0.83	0.0114	2.41E-02
1.0E-03	12	50	No	0.9295	0.0144	8.63E-03
1.0E-03	12	100	No	0.9813	0.0113	1.83E-03
1.0E-03	12	100	Yes	0.9977	0.0107	2.59E-04
1.0E-04	12	100	Yes	0.9355	0.0126	1.54E-03
1.0E-03	14	100	Yes	0.9972	0.0105	2.67E-04
1.0E-03	**16**	**100**	**Yes**	**0.9975**	**0.0104**	**2.40E-04**
1.0E-04	16	100	Yes	0.933	0.0115	6.17E-04

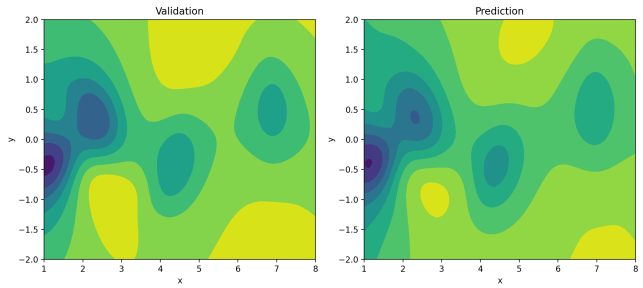

Figure 8: Comparaison entre la pression des données de validation et celle prédite par le modèle après entraînement au temps $t = 0$

La figure 8 compare le champ de pression à $t = 0$ entre les données de validation et la prédiction du modèle. Après 10 000 epochs, les champs de pression sont très similaires. À noter que leurs valeurs numériques ne sont pas présentées puisque le problème tel que posé ne permet pas de garanti que la pression va garder le bon ordre de grandeur. La figure 8 sert plutôt à vérifier que champs de pression conserve les mêmes variations, ce qui est le cas dans cette expérience. Pour valider numériquement la qualité de la prédiction, il est préférable d'utiliser l'EQM sur les champs de vitesse u et v ainsi que les coefficients $\lambda = (\lambda_1, \lambda_2)$. Le tableau 3 résume les différentes configurations du réseau testées pour reproduire les résultats de l'état de l'art. En augmentant la profondeur du réseau en plus d'utiliser des connexions résiduelles, il est possible d'obtenir des résultats proches de l'état de l'art en utilisant 10 000 epochs au lieu de 250 000 [Bafghi and Raissi, 2023].

La figure 9 présente la courbe de convergence avec un taux d'apprentissage de 1E-3. Celle-ci se caractérise par beaucoup d'oscillation qui sont souvent signe d'un taux d'apprentissage trop important. C'est pour cela que quelques essais ont été faits avec un taux d'apprentissage plus faible de 1E-4. Sa

courbe d'apprentissage est illustrée par la figure 10. Avec un taux d'apprentissage plus faible, les oscillations sont fortement atténuées, par contre, après 10 000 epochs, l'erreur sur les données de validation ainsi que sur les coefficients λ est plus élevée. La pente de la courbe n'ayant pas encore atteint un plateau, il est raisonnable de penser qu'avec un nombre d'epochs plus élevés de meilleurs résultats auraient été obtenus.

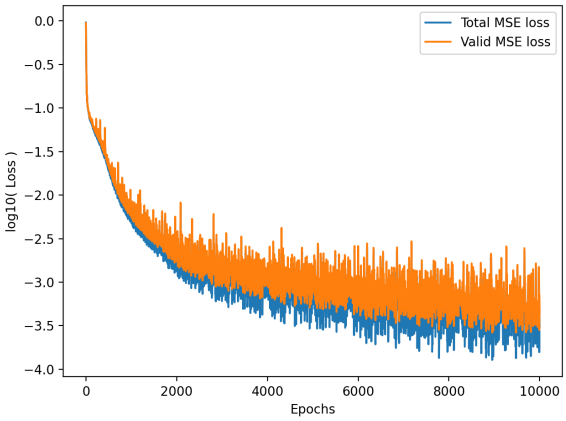

Figure 9: Courbe de la fonction perte sur les données d'entraînement et de validation avec un taux d'apprentissage de lr=1E-3

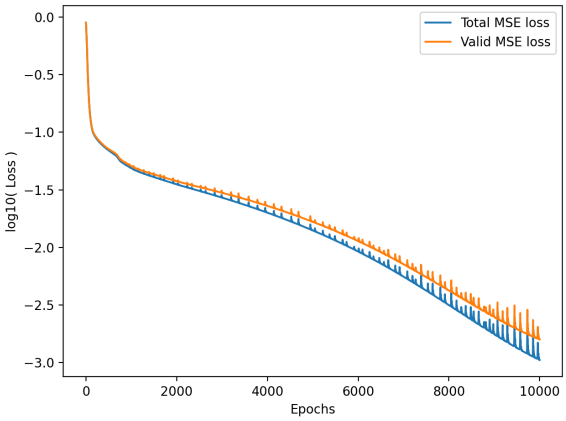

Figure 10: Courbe de la fonction perte sur les données d'entraînement et de validation avec un taux d'apprentissage de lr=1E-4

4 Analyse de l'approche

L'approche utilisée pour nous familiariser avec le sujet des PINNs a d'abord été de lire l'article original [Raissi *et al.*, 2019] pour comprendre le sujet ainsi qu'identifier les travaux qui les précèdes et leurs champs d'application. Par la suite nous avons identifié quelques articles sur leur évolution [Cheng and Zhang, 2021]. Il est possible de constater qu'en règle générale, les PINNs utilisent des architectures assez simples qui permettent notamment de les reproduire assez facilement. Nous avons jugé qu'il était réalisable de faire notre propre implémentation surtout qu'il existe plus librairies libres d'accès, notamment [Bafghi and Raissi, 2023]. Reproduire une implémentation nous a permis de valider notre compréhension du sujet, mais aussi d'étudier l'ajout de caractéristique qui ne sont pas présentées dans le papier original (les connexions résiduelles).

- Revue de la littérature, des travaux antérieurs et de ceux qui ont suivis
- Recherche d'implémentations
- Reproduction d'expérience avec une implémentation identifiée préalablement
- Reproduction de l'implémentation, en autre pour y apporter des modifications
- Validation de la reproduction (validation avec des résultats similaires à l'état de l'art)
- Étude des hyper paramètres (dimensions, optimiseur, taux d'apprentissage) et des modifications apportées (connections résiduelles)

5 Conclusion

La revue de la littérature ainsi que les expériences faites dans le cadre de ce travail démontre que les PINNs sont des outils efficaces pour interpoler des données à partir de conditions initiales et frontières si les EDP qui gouvernent le phénomène a observé sont connues comme démontré par l'expérience sur l'équation de Scrödinger. Sinon, à partir de données réparties dans le domaine spatio-temporel, il est aussi possible de compléter des EDP qui ne sont pas entièrement connues comme démontrer par l'expérience sur les équations de Navier-Stokes. Toutefois, les expériences ont démontré que les PINNs sont sensibles aux hyper paramètres choisis. Notamment, la méthode d'optimisation ainsi que le taux d'apprentissage a une influence importante sur les performances des PINNs. De plus, l'expérience de Schrödinger a démontré qu'approfondir un réseau ou bien utiliser des connexions résiduelles n'apporte pas toujours un avantage pour des problèmes simples. Comme mentionner par [Raissi *et al.*, 2019], les PINNs n'ont pas objectif pour de remplacer des méthodes déjà bien ancrer dans les domaines physiques, par exemple les analyses par éléments finies (FEM), mais plutôt de les utiliser conjointement.

A GitHub

Notre implémentation des PINNs et des Res-PINNs sur l'expérience de Schrödinger et Navier-Stokes est disponible sur le GitHub :
https://github.com/oihanc/INF8225_TD4.git

B Présentation

Une courte présentation résumant notre recherche sur les PINNs ainsi que nos expériences est disponible au :
lien suivant.

References

[Bafghi and Raissi, 2023] Reza Akbarian Bafghi and Maziar Raissi. PINNs-torch: Enhancing speed and usability of physics-informed neural networks with PyTorch. 2023.

[Cheng and Zhang, 2021] Chen Cheng and Guang-Tao Zhang. Deep learning method based on physics informed neural network with resnet block for solving fluid flow problems. 13(4):423, 2021. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[E and Yu, 2017] Weinan E and Bing Yu. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, 2017.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[Raissi *et al.*, 2019] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. 378:686–707, 2019.

[Sirignano and Spiliopoulos, 2018] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. 375:1339–1364, 2018.

[Wang *et al.*, 2024] Jie Wang, Xufeng Xiao, Xinlong Feng, and Hui Xu. An improved physics-informed neural network with adaptive weighting and mixed differentiation for solving the incompressible navier–stokes equations. 112(18):16113–16134, 2024.